Table of Contents I

notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

III conditioning

notes

Contents map

developed content units	**taxonomy levels**
ill conditioning	u1, e1
ill posedness	u1, e1

prerequisite content units	**taxonomy levels**
least squares regression	u1, e1

notes

Main ILO of sub-module "Ill conditioning"

Describe what ill conditioning and ill posed-
ness mean, in the context of system identification

Recognize when ill conditioning may happen in practice

notes

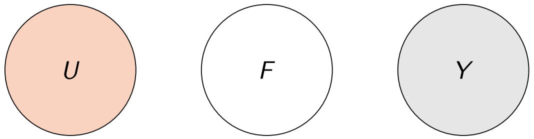
- By the end of this module, I want you to be able to tell whether a system identification problem might be hard to solvenot because the algorithm is wrong, but because the problem itself is "tricky" in a mathematical sense. We'll break down what it means for a problem to be ill-posed or ill-conditioned, and most importantly, how to spot it in real data.

Starting point: system identification

starting from

$$y[k] = f\Big(u[k], u[k-1], \dots\Big) + d[k] \qquad \mathcal{D} = \{u[k], y[k]\}_{k \in \mathcal{K}}$$

identify the model $f(\cdot)$

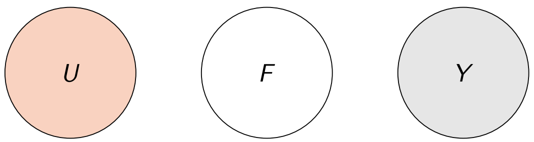


notes

- Let's start from the general setting: we have an inputoutput dataset, and our goal is to find a model that maps inputs to outputs. In other words, we're doing system identification. Sounds simple, right? But whether or not this is a well-posed problem depends on things like the structure of the data, how noisy it is, and how we model $f(\cdot)$. This will be our playground for understanding ill-posedness and ill-conditioning.

Definition of ill-posedness and ill-conditioning

$$y[k] = f\Big(u[k], u[k-1], \dots\Big) + d[k] \qquad \mathcal{D} = \{u[k], y[k]\}_{k \in \mathcal{K}}$$



- *ill-posed problem (in the Hadamard sense):* solution is either not unique or does not depend continuously on the data
- *ill-conditioned problem:* solution is very sensitive to the data

notes

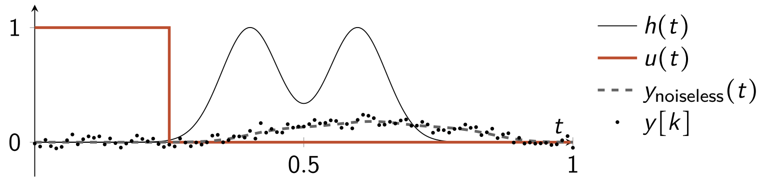
- These definitions might sound abstract, but here's the intuition. A problem is ill-posed if there's no unique solution, or if small changes in the data can cause huge changes in the result. Ill-conditioning is a bit more subtle: the solution exists and is unique, but it's extremely sensitive to small errorslike noise in your measurements. Both are bad news when you're trying to learn a model from data.

Example: the Hunt reconstruction problem

(continuous-time LTI with sampled output)

$$h(t) = \exp\left(-\left(\frac{t-0.4}{0.075}\right)^2\right) + \exp\left(-\left(\frac{t-0.6}{0.075}\right)^2\right) \qquad u(t) = \begin{cases} 1 & \text{if } 0 \le t \le 0.25 \\ 0 & \text{otherwise} \end{cases}$$

$$y_{\text{noiseless}}(t) = \int_0^{+\infty} h(\tau)u(t-\tau)d\tau \qquad y[k] = y_{\text{noiseless}}[k] + v(k)$$



notes

- This is a classic reconstruction problem: were trying to recover the impulse response $h(t)$ from noisy output samples. Even if this looks innocent, it turns out to be a perfect example of an ill-conditioned problem. We'll see why very soon. For now, just get familiar with the idea that even in simple-looking setups, things can go wrong when noise and sampling are involved.

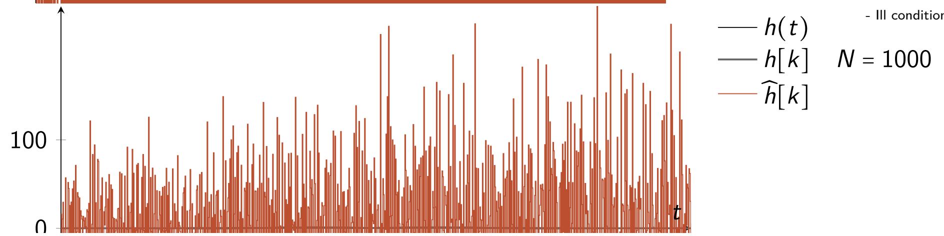
This problem can be solved with linear algebra!

$$\boldsymbol{y} = U\boldsymbol{h} + \boldsymbol{d} \quad \Rightarrow \quad \widehat{\boldsymbol{h}} = \left(U^T U\right)^{-1} U^T \boldsymbol{y}$$

notes

- So far, were just applying linear algebra: youve seen this in least squares regression. The cool part is that this integral problem becomes a linear system. But heres the catch: just because you can write down the inverse doesnt mean it behaves well. Whether or not this works depends heavily on the properties of the matrix U.

Is the Hunt reconstruction problem well defined?



notes

- Heres where the issue becomes visible: small changes in the measurements lead to wildly different reconstructions. Thats a clear sign of ill-conditioning. Notice how the reconstruction improves only when we throw a huge amount of data at the problem. So its not wrong, but its very fragile.

What is happening?

$$\boldsymbol{e} = \boldsymbol{h} - \widehat{\boldsymbol{h}} = U^{-1}\boldsymbol{d}$$

$$\frac{\|\boldsymbol{e}\|}{\|\boldsymbol{h}\|} \leq \frac{\sigma_{\max}(U)}{\sigma_{\min}(U)} \frac{\|\boldsymbol{d}\|}{\|U\boldsymbol{h}\|}$$

- the slower u the higher $\frac{\sigma_{\max}(U)}{\sigma_{\min}(U)}$
- the faster Δ the higher $\frac{\sigma_{\max}(U)}{\sigma_{\min}(U)}$

notes

- This equation quantifies the problem. The reconstruction error is amplified by the condition number of U. If that number is large, even tiny noise $\boldsymbol{d}$ becomes huge error $\boldsymbol{e}$. And what makes the condition number large? Slow input signals, or high-frequency systems. So now were connecting theory with practice: how you choose your input signal really matters.

how can we improve our estimates?

$\mapsto$ regularization

notes

- So what can we do about it? If the problem is ill-conditioned, regularization can help. Its like adding some "reasonable assumptions" to stabilize the solution. Thats what well look at next.

Summarizing

Describe what ill conditioning and ill posed-ness mean, in the context of system identification

Recognize when ill conditioning may happen in practice

- TODO

notes

- You should now be able to tell whether a system identification problem is ill-posed or ill-conditioned, and explain why that matters. Use the inputoutput data structure and ask: is the mapping well-behaved? Is it sensitive to noise? And remember, in practice, its not just about having a solutionits about having a reliable one.

Most important python code for this sub-module

notes

Linear algebra tools

- `numpy.linalg.solve`
- `numpy.linalg.inv`

notes

- when discussing ill-conditioning, we're typically dealing with numerical computation issues that arise in linear algebra, optimization, and other mathematical operations, so all the Python code that refer to that are related to this module

Self-assessment material

notes

notes

- see the associated solution(s), if compiled with that ones :)

Question 1

Which of the following best describes the difference between an ill-posed and an ill-conditioned problem in system identification?

Potential answers:

I: **(wrong)** Ill-conditioned problems have no solution, while ill-posed problems have too many.

II: **(correct)** Ill-posed problems may lack uniqueness or continuous dependence on the data, while ill-conditioned problems are extremely sensitive to small changes in data.

III: **(wrong)** Ill-posed problems always have unstable solutions, while ill-conditioned ones always diverge.

IV: **(wrong)** Ill-conditioning is due to randomness in the input, while ill-posedness is due to measurement noise.

V: **(wrong)** I do not know

Solution 1:

Question 2

Why does the Hunt reconstruction problem become ill-conditioned as the length of the input increases?

Potential answers:

I: **(wrong)** Because more data always makes the system overdetermined.
II: **(correct)** Because slow or non-diverse input signals lead to poor numerical conditioning of the matrix U.
III: **(wrong)** Because increasing the number of samples reduces the noise-to-signal ratio.
IV: **(wrong)** Because the model structure becomes nonlinear with large N.
V: **(wrong)** I do not know

Solution 1:

- Ill conditioning 3

When input signals change slowly or are not sufficiently rich, the matrix U formed by the convolution becomes poorly conditioned, meaning the singular values vary greatly. This increases the condition number, making the estimation highly sensitive to measurement noise.

notes

- see the associated solution(s), if compiled with that ones :)

Question 3

In the context of system identification, what does the condition number $\frac{\sigma_{\max}(U)}{\sigma_{\min}(U)}$ represent?

Potential answers:

I: **(correct)** The maximum amplification of relative errors in the data to the estimation error.
II: **(wrong)** The rate of convergence of the optimization algorithm used.
III: **(wrong)** The ratio between input and output power in the system.
IV: **(wrong)** The likelihood that a model is nonlinear.
V: **(wrong)** I do not know

Solution 1:

- Ill conditioning 4

The condition number quantifies how sensitive the output of a system (e.g., the estimated parameters) is to small changes in the input (e.g., the data). A high condition number indicates that even small noise in the data can lead to large errors in the solution.

notes

- see the associated solution(s), if compiled with that ones :)

Question 4

What is a practical way to reduce ill-conditioning in system identification?

Potential answers:

I: **(correct)** Use richer or faster-varying input signals during data collection.
II: **(wrong)** Use fewer data points to simplify the estimation problem.
III: **(wrong)** Reduce the noise artificially in the measurements after data collection.
IV: **(wrong)** Make the input signal constant over time to ensure stability.
V: **(wrong)** I do not know

Solution 1:

One way to improve the conditioning of the identification problem is to use an input signal that excites a wide range of system dynamics. This helps ensure the matrix U has more balanced singular values, reducing the condition number.

notes

- see the associated solution(s), if compiled with that ones :)

Question 5

Why is regularization used when solving ill-conditioned system identification problems?

Potential answers:

I: **(wrong)** To make the inverse of U exactly equal to zero.
II: **(correct)** To stabilize the solution by penalizing large parameter values or enforcing smoothness.
III: **(wrong)** To reduce the condition number by artificially shrinking the data.
IV: **(wrong)** To avoid computing the inverse of the matrix altogether.
V: **(wrong)** I do not know

Solution 1:

Regularization introduces additional constraints (e.g., on the norm of the parameter vector) to control the sensitivity of the solution to noise in the data. It does not remove ill-conditioning but mitigates its effects, often improving generalization.

notes

- see the associated solution(s), if compiled with that ones :)

Recap of sub-module "Ill conditioning"

- Ill-posed problems may lack a solution, have multiple solutions, or be highly sensitive to small changes in data
- Ill-conditioned problems have a solution, but it is numerically unstable and highly sensitive to input errors
- The condition number of a matrix quantifies the degree of ill-conditioning; a high condition number indicates poor numerical stability
- In system identification, slowly varying or insufficiently rich input signals can lead to ill-conditioning
- Regularization techniques can mitigate the effects of ill-conditioning by introducing stability through additional constraints
- Choosing appropriate input signals is critical to ensuring well-posed and well-conditioned identification problems
- Understanding the structure and properties of the data matrix (e.g., U in least squares problems) is essential to diagnose ill-conditioning

notes

- the most important remarks from this sub-module are these ones