Linear Algebra: Conditioning & Stability

Nathan Kutz - Lecture Notes

November 30, 2023

Solving a problem numerically is heavily dependent on the how this problem is defined and the numerical algorithm used to solve it. Bad-construction of these elements will bring ill-behavior of the solution procedure, and hence false results. Conditioning and stability are quality measures for the construction of these elements. The former assess the construction of the problem definition, whereas the latter concerned with the construction of the algorithm. In this script we will explore the meaning of conditioning and stability more rigorously and their applications.

Motivation

Claiming the reliability of the results obtained from a numerical procedure is a concern of crucial importance in numerical analysis. To claim reliability, one needs to quantify the robustness of the inner components of the numerical procedure (i.e. problem definition, numerical algorithm) followed to compute the solution. Conditioning and stability are metrics that describe the robustness of these inner components. Both concepts try to answer the following question "How reliable is my numerical procedure?".

Introduction

To illustrate the meaning of conditioning and stability lets list the steps of numerical procedure for an arbitrary problem.

1. Problem definition: In general sense a numerical problem can be defined in the following generic form[1], with f as mapping model.
$$f : \text{Inputs} \mapsto \text{Outputs}$$

2. Numerical algorithm: Steps to develop the numerical solution based on the problem definition.

[1] i.e analytical expression, matrix, algorithm. . .

Conditioning and stability address the robustness of the model f and the numerical algorithm respectively by measuring their sensitivity to perturbations.

Conditioning of a Problem

Definition 1 *A problem definition $f(x)$ is said to be well-conditioned if small perturbations (δ) in the input ($x + \delta x$) will result small perturbations in the output $f(x + \delta x)$.*

Definition 2 *A problem definition $f(x)$ is said to be ill-conditioned if small perturbations (δ) in the input ($x + \delta x$) will result very large perturbations in the output $f(x + \delta x)$.*

As mentioned conditioning address how the problem is formulated or the nature of the problem itself, it is a property of the problem. Conditioning is measured through the condition number "*kappa*" which quantify the sensitivity of the problem. The condition number can be computed in two ways

Absolute condition number ($\hat{\kappa}$)

The absolute condition number of a function $f : R^m \to R^n$ at a point $\mathbf{x} \in R^m$ is defined by

$$\hat{\kappa}(\mathbf{x}) = \lim_{\delta \to 0} \sup_{\|\delta \mathbf{x}\| \le \delta} \frac{\|f(\mathbf{x} + \delta \mathbf{x}) - f(\mathbf{x})\|}{\|\delta \mathbf{x}\|}$$

The absolute condition number of f is the limit of the change in output over the change of input, which very similar to the definition of the derivative. This makes sense since both are concerned to quantify rate of change. Hence as $\hat{\kappa} \to \infty$ or get very large the problem become ill-conditioned. The exact cutoff between well- and ill-conditioned depends on the context of the problem and the uses of the results.

Relative condition number (κ)

The relative condition number is introduced to rule out the dependance on the nature of data of f, since that could bring false judgments such the following

data values are small $\to$ small $\hat{\kappa}$ $\to$ well-conditioned

Hence to detour such problem $\hat{\kappa}$ is normalized by data. Which makes the relative condition number reads

$$\kappa(\mathbf{x}) = \lim_{\delta \to 0} \sup_{\|\delta \mathbf{x}\| \le \delta} \frac{\|f(\mathbf{x} + \delta \mathbf{x}) - f(\mathbf{x})\| \Big/ \|f(\mathbf{x})\|}{\|\delta \mathbf{x}\| \Big/ \|\mathbf{x}\|}$$

Application of Conditioning Analysis

Example 1 *fsdf*

dfsdf ■

Stability of an Algorithm

Designing algorithms[2] to solve numerical problems in computing devices comes with a challenge that exact computations are not feasible. Approximations of mathematical objects[3] introduce errors that are unavoidable. Algorithm designer has investigate how these errors transcend through the algorithm. Bad design of an algorithm, lead to disastrous consequences such as error amplification [4]. Stability address the behavior of the algorithm under these errors. Let $\hat{f}(x)$ to be an algorithm to solve problem $f(x)$.

[2] i.e Any process to solve the problem definition. Could simply be direct evaluation of the problem definition, collection of steps, or numerical method

[3] i.e numbers, sequences, ...

[4] Hence false results

Definition 3 *An algorithm $\hat{f}(x)$ to solve problem $f(x)$ is said to be stable if small perturbations (typically due to roundoff error) in the input will result small perturbations in its output.*

Definition 4 *An algorithm is said to be unstable if small perturbations in the input will result large perturbations in its output.*

The consequence of stability can be traced by computing the either the absolute or relative errors[5]. A stable algorithm will make these errors small.

[5] As mentioned, it is better to consider relative error. To eliminate data dependency

Absolute error

$$\|\hat{f}(\mathbf{x}) - f(\mathbf{x})\| \sim \varepsilon$$

Relative error

$$\frac{\|\hat{f}(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|} \sim \varepsilon$$

Where ε is machine epsilon or machine precision. This means an optimal stable algorithm will produce results that is accurate to machine precision[6].

[6] This is the lower bound in machine computing

Application of Stability

Example 2 *fsdf*

In this section we will illustrate the stability analysis of some algorithm classes, noting that stability analysis could performed to other classes

- Algorithm that evaluate the problem definition directly.
- Iterative algorithms[7].
- Recursive algorithm[8].

[7] The error get summed through the algorithm

[8] The error get multiplied through the algorithm

Tylor expansion

$$y(t+\Delta t) = y(t) + \Delta \frac{dy}{dt} + \frac{\Delta t^2}{2} \frac{d^2 y(c)}{dt^2}$$

Given this differential problem

$$\frac{dy}{dt} = f(y)$$

With Forward Eular approximation, the ODE reads

$$\frac{y(t_n) - y(t_n + \Delta t)}{\Delta t} + \mathcal{O}(\Delta t^2) = f(y(t_n))$$

References