
Scientific Computing 1
Handout 7
November 26, 2014

Error Concepts, Stability and Conditioning

Consider $y = f(x)$, $x \in D \subset \mathbb{R}^n$, $y \in V \subset \mathbb{R}^m$ and the computed result $\hat{y} = y + \Delta y = f(x + \Delta x)$

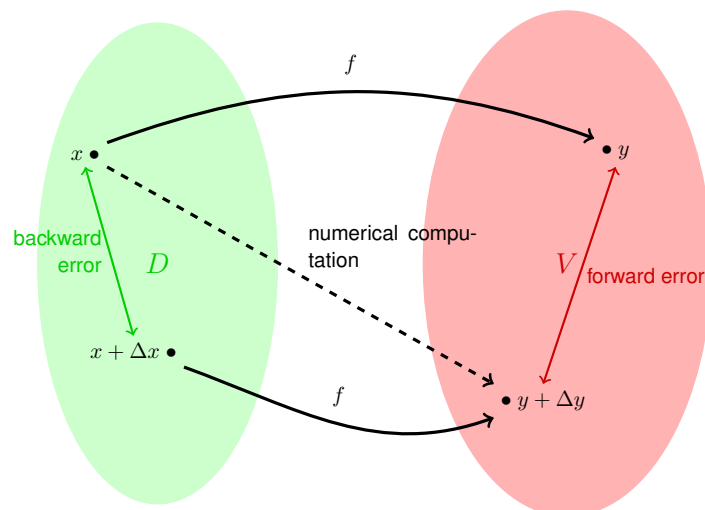


Figure 1: relation of forward and backward errors and numerical computation

- **forward error:**

- *absolute:* $\|y - \hat{y}\|$,
- *relative:* $\|y - \hat{y}\| / \|y\|$.

Estimated in a *forward error analysis*.

- **backward error:**

- *absolute:* η ,
- *relative:* $\eta / \|x\|$,

where $\eta := \inf\{\|\Delta x\| \mid \hat{y} = f(x + \Delta x)\}$. Estimated in a *backward error analysis*.

- **numerical stability** (of an algorithm):

- An algorithm producing a relative backward error of the magnitude of the relative data errors $\frac{\|\Delta x\|}{\|x\|}$ is called *(numerically) backward stable*.
- If an algorithm produces relative forward errors of the magnitude a backward stable algorithm would produce, then it is denoted *(numerically) forward stable*.

• **backward stable** \Rightarrow **forward stable**

• **(relative) condition number:**

$$c(f, x) := \frac{\|x\|}{\|f(x)\|} \|f'(x)\|$$

Depends on problem *and* the data. Mathematical problems are often not generally badly conditioned but conditionally badly conditioned depending on the data.

- $c(f, x) \approx 1 \Rightarrow$ well conditioned
- $c(f, x) \gg 1 \Rightarrow$ badly conditioned
- $c(f, x) \ll 1 \Rightarrow$ possibly bad due to “loss of information” *leftrightarrow* large backward error.

An algorithm can already become unstable if a (well conditioned) problem is subdivided in several steps and only one step is ill conditioned.

• **rule of thumb:**

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}$$

• **justification of computational results:**

- good conditioning and stable algorithm \Rightarrow reliable results
- bad conditioning or unstable algorithm \Rightarrow unsure results