Error analysis

Marcin Kuta

Accuracy and precision

Accuracy refers to the error of an approximate quantity

Precision is the accuracy with which the basic arithmetic operations are performed

- Accuracy and precision are the same for scalar computation $c = a * b$
- Accuracy can be much worse than precision in the solution of a linear system of equations
- Accuracy is not limited by precision. Arithmetic of a given precision can be used to simulate arithmetic of arbitrarily high precision.

Absolute and relative error

Absolute error

$$|x - \hat{x}|$$

Relative error

$$\frac{|x - \hat{x}|}{|x|}$$

Significant digits

$$\underbrace{1.7320}_{\text{5 significant digits}} \qquad 0.0\,\underbrace{491}_{\text{3 significant digits}}$$

Well-posed problem

Problem f is well-posed or stable if it meets three Hadamard criteria:

- Existence: the solution to the problem exists
- Uniqueness: the solution to the problem is unique
- Stability: the solution depends continously on initial conditions

Otherwise, problem f is ill-posed or unstable.

Problem can be well-posed even if its condtion number is infinite.

Ill-posed problem

Examples of ill-posed problems:

- finding the number of real roots of a polynomial
- global optimization
- deconvolution

Most inverse problems are ill-posed:

- inverse heat equation
- image reconstruction in computer tomography
- inferring seismic properties of the Earth's interior from surface observation

Ill-posed problem has to be reformulated.
A frequent approach to making a problem well-posed is ℓ_2 regularization.

Conditioning of a problem

We further distinguish between well-posed problems.

Problem f is well-conditioned if all small perturbations of x lead to small changes in $f(x)$.

Problem f is ill-conditioned if some small perturbation of x leads to a large change in $f(x)$.

Conditioning of a problem

$$\delta f = f(x + \delta x) - f(x) \tag{1}$$

Absolute condition number

$$\kappa = \sup_{\delta x} \frac{|\delta f|}{|\delta x|} \tag{2}$$

Relative condition number

$$\kappa = \sup_{\delta x} \left(\frac{|\delta f|}{|f(x)|} \Big/ \frac{|\delta x|}{|x|} \right) \tag{3}$$

If f is differentiable, then

$$\kappa = \frac{||J_f(x)||}{||f(x)||/||x||} \tag{4}$$

Accurate algorithm

An algorithm $\tilde{f}$ for a problem f is accurate if for each $x \in X$

$$\frac{|\tilde{f}(x) - f(x)|}{|f(x)|} = O(\epsilon_{\text{machine}}) \tag{5}$$

Stable algorithm

An algorithm $\tilde{f}$ for a problem f is stable if for each $x \in X$

$$\frac{|\tilde{f}(x) - f(\tilde{x})|}{|f(\tilde{x})|} = O(\epsilon_{\text{machine}}) \tag{6}$$

for some $\tilde{x}$ with

$$\frac{|\tilde{x} - x|}{|x|} = O(\epsilon_{\text{machine}}) \tag{7}$$

A stable algorithm gives nearly the right answer to nearly the right question.

Backward stable algorithm

An algorithm $\tilde{f}$ for a problem f is backward stable if for each $x \in X$

$$\tilde{f}(x) = f(\tilde{x}) \tag{8}$$

for some $\tilde{x}$ with

$$\frac{|\tilde{x} - x|}{|x|} = O(\epsilon_{\text{machine}}) \tag{9}$$

Equation (8) arises from replacing $O(\epsilon_{\text{machine}})$ with 0 in (6).

A backward stable algorithm gives exactly the right answer to nearly the right question.

Computational error

Numerical error

- is due to finite precison of floating-point arithmetic
- Other names: rounding error, błąd numeryczny, szum numeryczny

Truncation error

- is due to approximation method used
- Would remain even if all arithmetic were exact
- Other names: discretization error, błąd metody

Computational error

- Computational error $=$ Numerical error $+$ Truncation error

Errors

$$f(x) = y$$

Forward error:

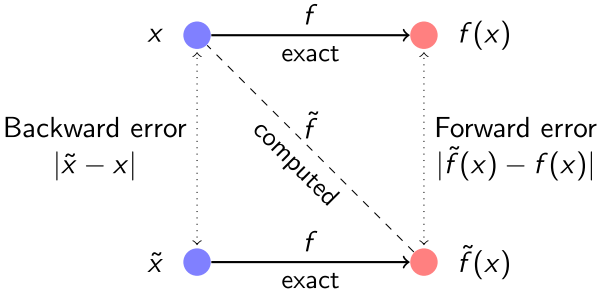
$$\text{Forward error } = \tilde{f}(x) - f(x) \tag{10}$$

Backward error:

$$\tilde{x} : \ f(\tilde{x}) = \tilde{f}(x)$$

$$\text{Backward error } = \tilde{x} - x \tag{11}$$

Errors



x
f
exact
$f(x)$
Backward error
$|\tilde{x} - x|$
$\tilde{f}$
computed
Forward error
$|\tilde{f}(x) - f(x)|$
$\tilde{x}$
f
exact
$\tilde{f}(x)$

Errors

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

- Forward error can be large due to poor algorithm $\tilde{f}$ but also due to bad conditioning of solved problem.
- Introducing backward error makes analysis of many algorithms easier.

Stability

- Forward stability

  Forward error and backward error are of similar magnitude
  Stabilność numeryczna
- Backward stability

  Specific to problems where numerical errors are the dominant form of errors
  Poprawność numeryczna

Errors in numerical derivatives

$$f(x+h) = f(x) + f'(x) \cdot h + f''(\theta) \cdot \frac{h^2}{2}, \quad \theta \in [x, x+h] \tag{12}$$

$$\frac{f(x+h) - f(x)}{h} = f'(x) + f''(\theta) \cdot \frac{h}{2}, \quad \theta \in [x, x+h] \tag{13}$$

$$\frac{f(x+h) - f(x)}{h} - f'(x) = f''(\theta) \cdot \frac{h}{2}, \quad \theta \in [x, x+h] \tag{14}$$

$$|f''(t)| \leq M \text{ for all } t \in [x, x+h] \tag{15}$$

Errors in numerical derivatives

$$\underbrace{E(h)}_{\substack{\text{computational}\\ \text{error}}} \leq \underbrace{\frac{Mh}{2}}_{\substack{\text{truncation}\\ \text{error}}} + \underbrace{\frac{2\epsilon}{h}}_{\substack{\text{rounding}\\ \text{error}}} \tag{16}$$

=błąd metody

=błąd numeryczny

$$h_{\text{min}} = 2\sqrt{\epsilon/M} \tag{17}$$

Errors in numerical derivatives

$$f(x+h) = f(x) + f'(x)\cdot h + f''(x)\cdot\frac{h^2}{2} + f'''(\theta_1)\cdot\frac{h^3}{6},\quad \theta_1 \in [x, x+h]$$

$$f(x-h) = f(x) - f'(x)\cdot h + f''(x)\cdot\frac{h^2}{2} - f'''(\theta_2)\cdot\frac{h^3}{6},\quad \theta_2 \in [x-h, x]$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{f'''(\theta_1) + f'''(\theta_2)}{2}\cdot\frac{h^2}{6} \tag{18}$$

$$|f'''(t)| \leq M \text{ for all } t \in [x-h, x+h] \tag{19}$$

Errors in numerical derivatives

$$\underbrace{E(h)}_{\substack{\text{computational}\\ \text{error}}} \leq \underbrace{\frac{Mh^2}{6}}_{\substack{\text{truncation}\\ \text{error}}} + \underbrace{\frac{\epsilon}{h}}_{\substack{\text{rounding}\\ \text{error}}} \tag{20}$$

=błąd metody

=błąd numeryczny

$$h_{\text{min}} = \sqrt[3]{3\epsilon/M} \tag{21}$$

Floating-point arithmetic

Floating-point arithmetic system $\mathbb{F}(\beta, p, L, U)$, where
β – base
p – precision, number of digits
L – smallest representable exponent
U – largest representable exponent

Relative numerical error ϵ introduced by replacing x with $\mathrm{fl}(x)$ is guaranteed to be no greater than machine epsilon, $\epsilon_{\text{machine}}$.

$$\epsilon = \left| \frac{x - \mathrm{fl}(x)}{x} \right| \leq \epsilon_{\text{machine}} \tag{22}$$

- Chopping: $\epsilon_{\text{machine}} = \beta^{1-p}$
- Rounding: $\epsilon_{\text{machine}} = \frac{1}{2}\beta^{1-p}$

IEEE Floating point representation

S	E	E	E	E	E	E	E	E	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

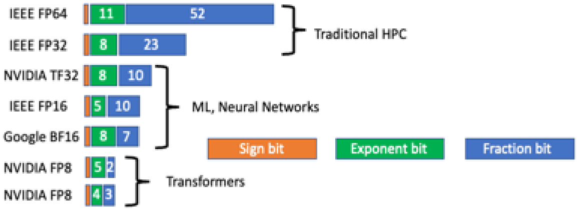
sign 8 exponent 23 mantissa bits

$$(-1)^{sign} * (2)^{exp-bias} * (1.mantissa)$$

0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$(-1)^{0} * (2)^{129-127} * (1.25) = 5$$

Floating point representations



Machine epsilon

Precision		
Half precision	9.77×10^{-04}	2^{-10}
Single precision	1.19×10^{-07}	2^{-23}
Double precision	2.22×10^{-16}	2^{-52}
Quadruple precision	1.93×10^{-34}	2^{-112}

Smallest repesentable number (dwarf number)

Precision		
Half precision		
Single precision	3.50×10^{-46}	2^{-151}
Double precision	4.94×10^{-324}	2^{-1074}
Quadruple precision		

Floating-point arithmetic

Let $x \in \mathbb{R}$, $y \in \mathbb{R}$.

Let $\bar{x} = \mathrm{fl}(x) \in \mathbb{F}$ and $\bar{y} = \mathrm{fl}(y) \in \mathbb{F}$ denote numerical representations of x and y.

Let ϵ denote relative error and δ denote absolute error introduced when replacing $x \in \mathbb{R}$ with $\bar{x} \in \mathbb{F}$:

$$\bar{x} = x(1 + \epsilon_x) = x + \delta_x$$
$$\bar{y} = y(1 + \epsilon_y) = y + \delta_y$$

Multiplication

Multiplication:

$$\bar{x}\bar{y} = xy(1+\epsilon)$$
$$\bar{x}\bar{y} = x(1+\epsilon_x)y(1+\epsilon_y) = xy(1+\epsilon_x+\epsilon_y+\epsilon_x\epsilon_y)$$

$$\bar{x}\bar{y} = xy+\delta$$
$$\bar{x}\bar{y} = (x+\delta_x)(y+\delta_y) = xy + x\delta_y + y\delta_x + \delta_x\delta_y$$

$$\epsilon \approx \epsilon_x + \epsilon_y \tag{23}$$

$$\delta \approx x\delta_y + y\delta_x \tag{24}$$

- relative error is small
- absolute error is large if x or y is large

Division

Division:

$$\bar{x}/\bar{y} = x/y(1+\epsilon)$$
$$\bar{x}/\bar{y} = \frac{x(1+\epsilon_x)}{y(1+\epsilon_y)} = \frac{x(1+\epsilon_x)(1-\epsilon_y)}{y(1+\epsilon_y)(1-\epsilon_y)} = \frac{x(1+\epsilon_x-\epsilon_y-\epsilon_x\epsilon_y)}{y(1-\epsilon_y^2)} \approx \frac{x}{y}(1+\epsilon_x-\epsilon_y)$$

$$\bar{x}/\bar{y} = x/y+\delta$$
$$\bar{x}/\bar{y} = \frac{x+\delta_x}{y+\delta_y} = \frac{(x+\delta_x)(y-\delta_y)}{(y+\delta_y)(y-\delta_y)} = \frac{xy-x\delta_y+y\delta_x-\delta_x\delta_y}{y^2-\delta_y^2} \approx \frac{xy-x\delta_y+y\delta_x}{y^2} = \frac{x}{y}+\frac{y\delta_x-x\delta_y}{y^2}$$

$$\epsilon \approx \epsilon_x - \epsilon_y \tag{25}$$

$$\delta \approx \frac{1}{y^2}(y\delta_x - x\delta_y) \tag{26}$$

- relative error is small
- absolute error is large if x is large or y is small

Addition

Addition:

$\bar{x} + \bar{y} = (x + y)(1 + \epsilon)$
$\bar{x}+\bar{y} = x(1+\epsilon_x)+y(1+\epsilon_y) = x+y+x\epsilon_x+y\epsilon_y = (x+y)(1+\frac{x\epsilon_x+y\epsilon_y}{x+y})$

$\bar{x} + \bar{y} = (x + y) + \delta$
$\bar{x} + \bar{y} = (x + \delta_x) + (y + \delta_y) = x + y + \delta_x + \delta_y$

$$\epsilon = \frac{x\epsilon_x + y\epsilon_y}{x + y} \tag{27}$$

$$\delta = \delta_x + \delta_y \tag{28}$$

- relative error is large if $x \approx -y$
- absolute error is small

Substraction

Substraction:

$\bar{x} - \bar{y} = (x - y)(1 + \epsilon)$
$\bar{x} - \bar{y} = x(1 + \epsilon_x) - y(1 + \epsilon_y) = x - y + x\epsilon_x - y\epsilon_y = (x - y)(1 + \frac{x\epsilon_x - y\epsilon_y}{x - y})$

$\bar{x} - \bar{y} = (x - y) + \delta$
$\bar{x} - \bar{y} = (x + \delta_x) - (y + \delta_y) = x - y + \delta_x - \delta_y$

$$\epsilon = \frac{x\epsilon_x - y\epsilon_y}{x - y} \tag{29}$$

$$\delta = \delta_x - \delta_y \tag{30}$$

- relative error is large if $x \approx y$
- absolute error is small

References I

[1] Lloyd Trefethen, David Bau
Numerical Linear Algebra,
1997

[2] Michael T. Heath
Scientific Computing. An Introductory Survey, 2nd Edition,
Chapter 1: Scientific Computing
2002

[3] Michael T. Heath
Chapter 1: Scientific Computing
http://heath.cs.illinois.edu/scicomp/notes/cs450_chapt01.pdf

[4] Nicholas Higham
Accuracy and Stability of Numerical Algorithms
2002

References II

[5] Nicholas Higham
What Is Backward Error?

[6] https://pythonnumericalmethods.berkeley.edu/notebooks/chapter09.00-Representation-of-Numbers.html