

Preliminaries

Lec 11: Numbers, Problems, and Algorithms

Floating - point
number.

Conditioning

stability

Floating-Point Numbers

Absolute and Relative Errors

In numerical analysis, we use an **algorithm** to *approximate* some quantity of interest.

- We estimate of the accuracy of the computed value via an **absolute error** or a **relative error**:

$$\frac{0.01 \cancel{\text{ft}}}{50 \cancel{\text{ft}}} = \frac{-0.01}{50}$$
$$e_{\text{abs}} = A_{\text{approx}} - A_{\text{exact}}$$
$$e_{\text{rel}} = \frac{A_{\text{approx}} - A_{\text{exact}}}{A_{\text{exact}}} = \frac{A_{\text{approx}}}{A_{\text{exact}}} - 1,$$

(absolute error)
(relative error)

where A_{exact} is the exact, analytical answer and A_{approx} is the approximate, numerical answer.

- If e_{abs} or e_{rel} is small, we say that the approximate answer is **accurate**.

- {
- "percentage" error ← (1 means 100%, --)
 - dimensionless (units of measurement not important)

Example: Stirling's Formula

Stirling's formula provides a “good” approximation to $n!$ for large n :

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e} \right)^n. \quad (\star)$$

- Assume that the exact value of $n!$ is found by `factorial`.
- Estimate $n!$ using (\star) .
- Show the accuracy of this approximation for various values of n .

Try in MATLAB:

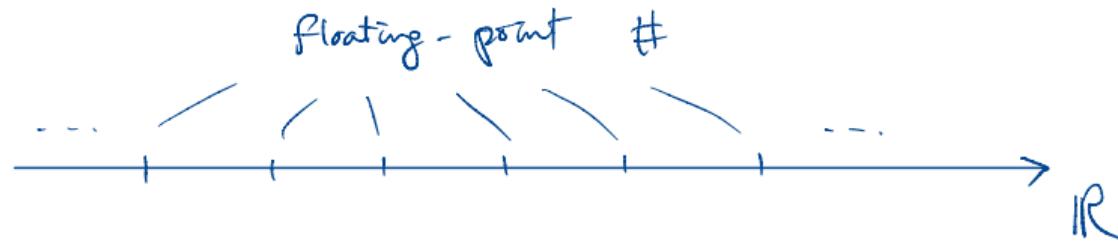
```
n = ...;
err_abs = sqrt(2*pi*n)*(n/exp(1))^n - factorial(n);
err_rel = err_abs/factorial(n);
disp(err_abs)
disp(err_rel)
```

Limitations of Digital Representations

A digital computer uses a finite number of bits to represent a real number and so it cannot represent all real numbers.

- The represented numbers cannot be arbitrarily large or small;
- There must be gaps between them.

So for all operations involving real numbers, it uses a subset of \mathbb{R} called the floating-point numbers, \mathbb{F} .



Floating-Point Numbers

base-2 scientific notation w/ given # of digits.

A floating-point number is written in the form $\pm(1 + F)2^E$ where

- E , the exponent, is an integer;
- F , the mantissa, is a number $F = \sum_{i=1}^d b_i 2^{-i}$, with $b_i = 0$ or $b_i = 1$.

Note that F can be rewritten as

$$F = 2^{-d} \sum_{k=0}^{d-1} b_{d-k} 2^k,$$

related to
gap size.

where M is an integer in $\mathbb{N}[0, 2^d - 1]$.

Consequently, there are 2^d evenly-spaced numbers between 2^E and 2^{E+1} in the floating-point number system.

$$2^E \quad 2^{E+1}$$

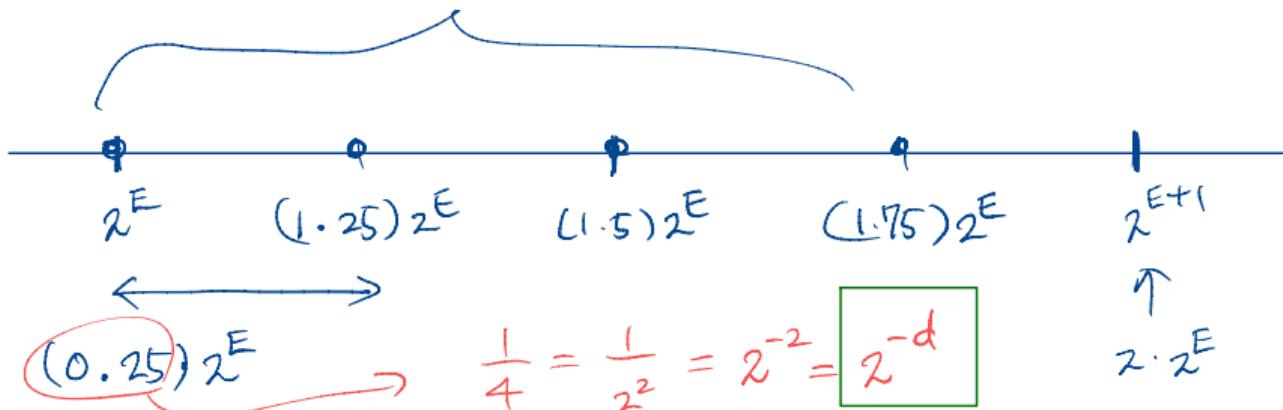
Number line ($d=2$)

$$(1+F)2^E \quad \text{where}$$

$$F = \sum_{i=1}^2 b_i 2^{-i} = \frac{b_1}{2} + \frac{b_2}{2^2}$$

$2^d = 2^2 = 4$ evenly spaced numbers

b_1	b_2	F
0	0	$\frac{0}{2} + \frac{0}{2^2} = 0$
0	1	$\frac{0}{2} + \frac{1}{2^2} = \frac{1}{4}$
1	0	$\frac{1}{2} + \frac{0}{2^2} = \frac{1}{2}$
1	1	$\frac{1}{2} + \frac{1}{2^2} = \frac{3}{4}$



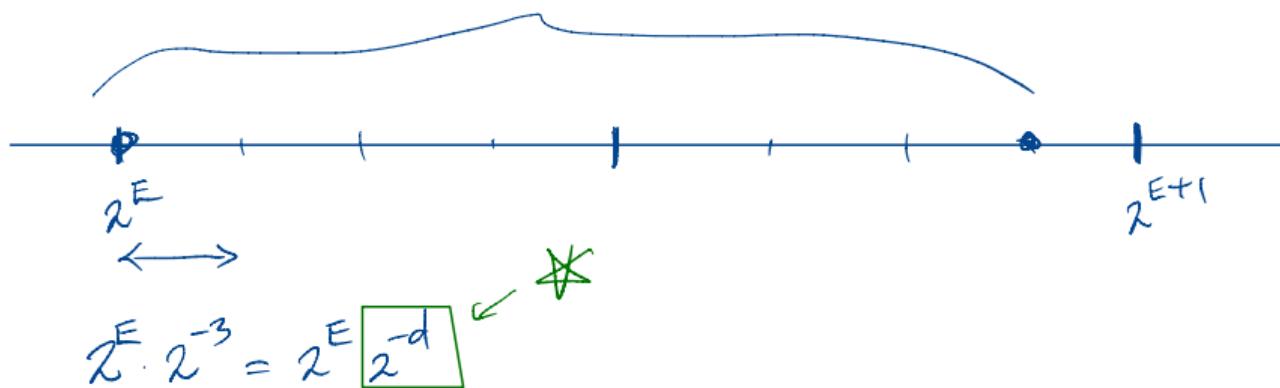
Number line ($d = 3$)

$(1+F)2^E$ where

$$F = \sum_{i=1}^2 b_i 2^{-i} = \frac{b_1}{2} + \frac{b_2}{2^2}$$

b_1	b_2	b_3		b_1	b_2	b_3	
0	0	0	0	1	0	0	0.5
0	0	1	0.125	1	0	1	0.625
0	1	0	0.25	1	1	0	0.750
0	1	1	0.375	1	1	1	0.875

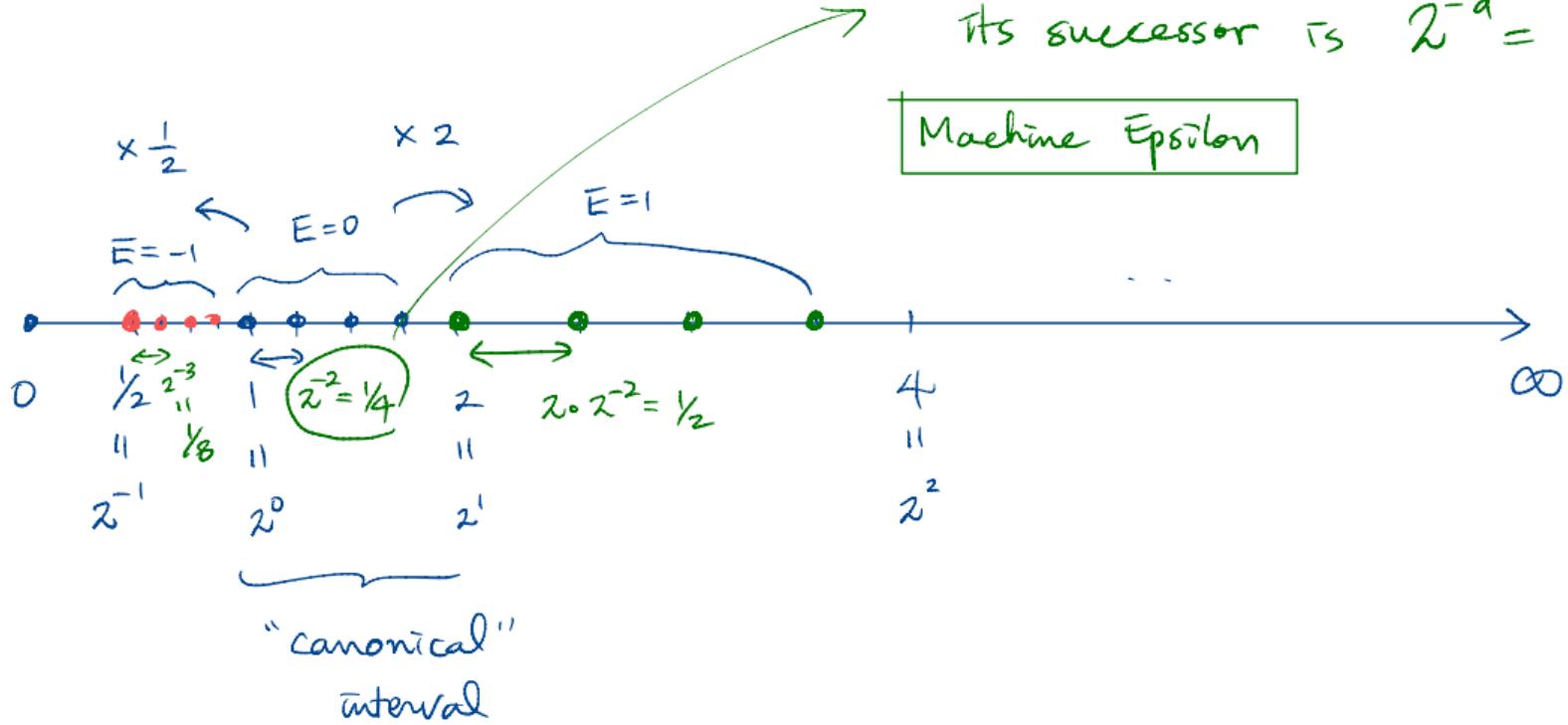
$2^d = 2^3 = 8$ evenly spaced points



Overview of F (w/ d=2)

$(1+F) \cdot 2^E$

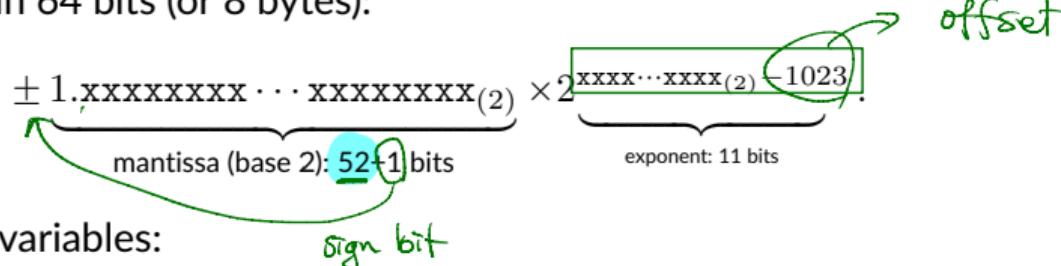
the gap b/w 1 and its successor is $2^{-d} = 1/4$.



Floating-Point Numbers – IEEE 754 Standard

- MATLAB, by default, uses double precision floating-point numbers, stored in memory in 64 bits (or 8 bytes):

$$d=52$$



- Predefined variables:

- eps = the distance from 1.0 to the next largest double-precision number:

"machine epsilon"

$$2^{-52}$$

$$\text{eps} = 2^{-52} \approx 2.2204 \times 10^{-16}.$$

$\sim 10^{-308}$

- realmin = the smallest positive floating-point number that is stored to full accuracy; the actual smallest is $\text{realmin}/2^{52}$. $\rightarrow 0$ anything smaller (underflow)
- realmax = the largest positive floating-point number

$$\sim 10^{308}$$

$\rightarrow \text{"Inf"}$ anything larger
(overflow)

Machine Epsilon and Relative Errors

The IEEE standard guarantees that the *relative representation error* and the *relative computational error* have sizes smaller than $\boxed{\text{eps}}$, the *machine epsilon*:

$$= x^{-d} \quad (\text{IEEE}, d=52)$$

- **Representation:** The floating-point representation, $\hat{x} \in \mathbb{F}$, of $x \in \mathbb{R}$ satisfies

$$\hat{x} = x(1 + \epsilon_1), \quad \text{for some } |\epsilon_1| \leq \frac{1}{2} \boxed{\text{eps}}.$$

$$\hat{x} = x + x\epsilon_1$$

- **Arithmetic:** The floating-point representation, $\hat{x} \oplus \hat{y}$, of the result of $\hat{x} + \hat{y}$ with $\hat{x}, \hat{y} \in \mathbb{F}$ satisfies

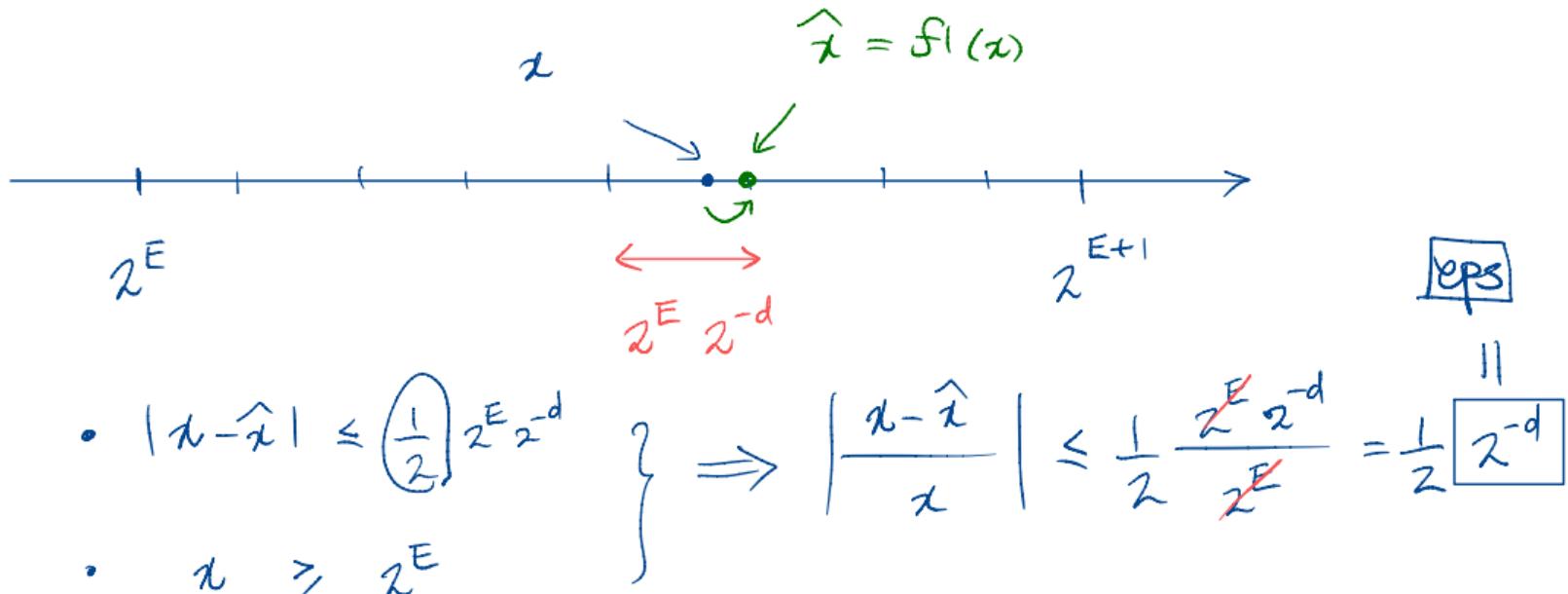
$$\hat{x} \oplus \hat{y} = (\hat{x} + \hat{y})(1 + \epsilon_2), \quad \text{for some } |\epsilon_2| \leq \frac{1}{2} \boxed{\text{eps}}.$$

$$\epsilon_1 = \frac{\hat{x} - x}{x}$$

Similarly with \ominus, \otimes, \div corresponding to $-$, \times , \div , respectively.

Representation Error ($d=3$)

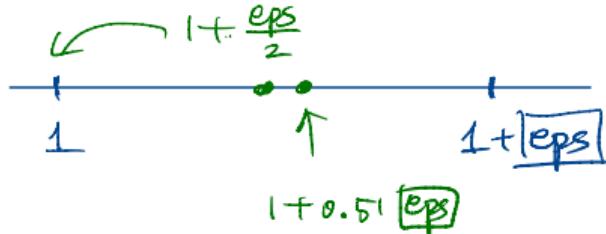
Let $x \in [x^E, x^{E+1})$.



Round-Off Errors

Computers CANNOT usually

- represent a number correctly;
- add, subtract, multiply, or divide correctly!!



Run the following and examine the answers:

```
format long  
1.2345678901234567890  
12345678901234567890  
*(1 + eps) - 1 : eps  
(1 + .5*eps) - 1 : 0  
(1 + .51*eps) - 1 : eps  
n = input(' n = '); ( n^(1/3) )^3 - n
```

1

1+eps

Catastrophic Cancellation

In finite precision storage, two numbers that are close to each other are indistinguishable. So subtraction of two nearly equal numbers on a computer can result in loss of many significant digits.

Catastrophic Cancellation

Consider two real numbers stored with 10 digits of precision:

$$e = 2.7182818284,$$

$$b = 2.7182818272.$$

- Suppose the actual numbers e and b have additional digits that are not stored.
- The stored numbers are good approximations of the true values.
- However, if we compute $e - b$ based on the stored numbers, we obtain $0.0000000012 = 1.2 \times 10^{-9}$, a number with only two significant digits.

Example 1: Cancellation for Large Values of x

Question

Compute $f(x) = e^x(\cosh x - \sinh x)$ at $x = 1, 10, 100$, and 1000 .

Numerically:

```
format long
x = input(' x = ');
y = exp(x) * (cosh(x) - sinh(x));
disp([x, y])
```

Example 2: Cancellation for Small Values of x

Question

Compute $f(x) = \frac{\sqrt{1+x} - 1}{x}$ at $x = 10^{-12}$.

Numerically:

```
x = 1e-12;  
fx = (sqrt(1+x) - 1)/x;  
disp( fx )
```

To Avoid Such Cancellations ...

- Unfortunately, there is no universal way to avoid loss of precision.
- One way to avoid catastrophic cancellation is to remove the source of cancellation by simplifying the given expression before computing numerically.
- For Example 1, rewrite the given expression recalling that

$$\cosh x = (e^x + e^{-x})/2 \quad \sinh x = (e^x - e^{-x})/2.$$

- For Example 2, try again after rewriting $f(x)$ as

$$f(x) = \frac{\sqrt{1+x} - 1}{x} \cdot \frac{\sqrt{1+x} + 1}{\sqrt{1+x} + 1} = \frac{1}{\sqrt{1+x} + 1}.$$

- Do you now have an improved accuracy?

Problems and Conditioning

Problems and Conditioning

- A mathematical *problem* can be viewed as a function $f : X \rightarrow Y$ from a data/input space X to a solution/output space Y .
- We are interested in changes in $f(x)$ caused by small perturbations of x .
- A *well-conditioned* problem is one with the property that all small perturbations of x lead to only small changes in $f(x)$

Condition Number

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and $\hat{x} = x(1 + \epsilon)$ be the representation of $x \in \mathbb{R}$.

- The ratio of the relative error in f due to the change in x to the relative error in x simplifies to

$$\frac{|f(x) - f(x(1 + \epsilon))|}{|\epsilon f(x)|}.$$

- In the limit of small error (ideal computer), we obtain

$$\begin{aligned}\kappa(x) &:= \lim_{\epsilon \rightarrow 0} \frac{|f(x) - f(x(1 + \epsilon))|}{|\epsilon f(x)|} \\ &= \left| \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon x) - f(x)}{\epsilon x} \cdot \frac{x}{f(x)} \right| = \left| \frac{xf'(x)}{f(x)} \right|, \quad (\star)\end{aligned}$$

which is called the **(relative) condition number**.

$$\left| \frac{f(\hat{x}) - f(x)}{f(x)} \right| \leq K_{f(x)} \left| \frac{\hat{x} - x}{x} \right| \leq K_{f(x)} \boxed{\text{eps}}$$

Example: Conditioning of Subtraction

Consider $f(x) = x - c$ where c is some constant. Using the formula (\star) , we find that the associated condition number is

$$\kappa(x) = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x}{x - c} \right|.$$

- It is large when $x \approx c$.

Example: Conditioning of Multiplication

The condition number of $f(x) = cx$ is

$$\kappa(x) = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x \cdot c}{cx} \right| = 1.$$

- No magnification of error.

Example: Conditioning of Function Evaluation

The condition number of $f(x) = \cos(x)$ is

$$\kappa(x) = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{-x \sin x}{\cos x} \right| = |x \tan x|.$$

- The condition number is large when $x = (n + 1/2)\pi$, where $n \in \mathbb{Z}$.

Example: Conditioning of Root-Finding

Let $r = f(a; b, c)$ be a root of $ax^2 + bx + c = 0$. Instead of direct differentiation, use implicit differentiation

$$r^2 + 2ar \frac{dr}{da} + b \frac{dr}{da} = 0.$$

Solve for the derivative,

$$f'(a) = \frac{dr}{da} = -\frac{r^2}{2ar + b} = -\frac{r^2}{\pm\sqrt{b^2 - 4ac}},$$

then compute the condition number using the formula (\star) to get

$$\kappa(a) = \left| \frac{af'(a)}{f(a)} \right| = \left| \frac{ar^2}{\pm r\sqrt{b^2 - 4ac}} \right| = \left| \frac{ar}{\sqrt{b^2 - 4ac}} \right|.$$

- Conditioning is poor for small discriminant, i.e., near repeated roots.

Stability of Algorithms

Algorithms

- Recall that we defined a *problem* as a function $f : X \rightarrow Y$.
- An *algorithm* can be viewed as another map $\tilde{f} : X \rightarrow Y$ between the same two spaces, which involves errors arising in
 - representing the actual input x as \hat{x} ;
 - implementing the function f numerically on a computer.

Analysis – General Framework

The relative error of our interest is

$$\begin{aligned} \left| \frac{\tilde{f}(\hat{x}) - f(x)}{f(x)} \right| &\leqslant \left| \frac{\tilde{f}(\hat{x}) - f(\hat{x})}{f(x)} \right| + \left| \frac{f(\hat{x}) - f(x)}{f(x)} \right| \\ &\approx \underbrace{\left| \frac{\tilde{f}(\hat{x}) - f(\hat{x})}{f(\hat{x})} \right|}_{\text{numerical error}} + \underbrace{\left| \frac{f(\hat{x}) - f(x)}{f(x)} \right|}_{\text{perturbation error}} \leqslant (\hat{\kappa}_{\text{num}} + \kappa_f) \boxed{\text{eps}}. \end{aligned}$$

where $\kappa = \kappa_f$ be the (relative) condition number of the exact problem f and

$$\hat{\kappa}_{\text{num}} = \max \left| \frac{\tilde{f}(\hat{x}) - f(\hat{x})}{f(\hat{x})} \right| \Bigg/ \left| \frac{\hat{x} - x}{x} \right|.$$

Example: Root-Finding Revisited

Consider again solving the quadratic problem $ar^2 + br + c = 0$.

- Taking $a = c = 1$ and $b = -(10^6 + 10^{-6})$, the roots can be computed exactly by hand: $r_1 = 10^6$ and $r_2 = 10^{-6}$.
- If numerically computed in MATLAB using the quadratic equation formula, r_1 is correct but r_2 has only 5 correct digits.
- Fix it using $r_2 = (c/a)/r_1$.