Big Data Analysis
(MA60306)

Bibhas Adhikari

Spring 2022-23, IIT Kharagpur

Lecture 6
January 13, 2023

Computing with data

Note An alternative way of expressing y is

$$y = \pm\beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t} \right) = \pm \underbrace{.d_1 d_2 \ldots d_t}_{t\text{-digit fraction}} \times \beta^e$$

where $0 \leq d_i \leq \beta - 1$ and $d_1 \neq 0$ (for normalized numbers)

Computing with data

Note An alternative way of expressing y is

$$y = \pm\beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t}\right) = \pm \underbrace{.d_1 d_2 \ldots d_t}_{t\text{-digit fraction}} \times \beta^e$$

where $0 \leq d_i \leq \beta - 1$ and $d_1 \neq 0$ (for normalized numbers) In this representation, d_1 is called the *most significant digit* and d_t the *the least significant digit*

Computing with data

Note An alternative way of expressing y is

$$y = \pm\beta^e\left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t}\right) = \pm \underbrace{.d_1 d_2 \ldots d_t}_{t\text{-digit fraction}} \times\beta^e$$

where $0 \leq d_i \leq \beta - 1$ and $d_1 \neq 0$ (for normalized numbers) In this representation, d_1 is called the *most significant digit* and d_t the *the least significant digit*

Machine and arithmetic	β	t	$e_{\min}$	$e_{\max}$	u
Cray-1 single	2	48	−8192	8191	4×10^{-15}
Cray-1 double	2	96	−8192	8191	1×10^{-29}
DEC VAX G format, double	2	53	−1023	1023	1×10^{-16}
DEC VAX D format, double	2	56	−127	127	1×10^{-17}
HP 28 and 48G calculators	10	12	−499	499	5×10^{-12}
IBM 3090 single	16	6	−64	63	5×10^{-7}
IBM 3090 double	16	14	−64	63	1×10^{-16}
IBM 3090 extended	16	28	−64	63	2×10^{-33}
IEEE single	2	24	−125	128	6×10^{-8}
IEEE double	2	53	−1021	1024	1×10^{-16}
IEEE extended (typical)	2	64	−16381	16384	5×10^{-20}

Computing with data

The spacing between the floating point numbers is characterized in terms of *machine epsilon*, denotes as ϵ_M which is the distance from 1.0 to the next larger floating point number.

Computing with data

The spacing between the floating point numbers is characterized in terms of *machine epsilon*, denotes as ϵ_M which is the distance from 1.0 to the next larger floating point number.

Homework Show that

$$\epsilon_M = \beta^{1-t}$$

Computing with data

The spacing between the floating point numbers is characterized in terms of *machine epsilon*, denotes as ϵ_M which is the distance from 1.0 to the next larger floating point number.

Homework Show that

$$\epsilon_M = \beta^{1-t}$$

Lemma The spacing between a normalized floating point number x and an adjacent normalized floating point number is at least $\beta^{-1}\epsilon_M|x|$ and at most $\epsilon_M|x|$.

Computing with data

- Given $x \in \mathbb{R}$, $fl(x)$ denotes an element from F that is nearest to x (how to break the ties)
- $x \mapsto fl(x)$ is called *rounding* which is monotone: $x \geq y$ implies $fl(x) \geq fl(y)$

Computing with data

Given $x \in \mathbb{R}$, $fl(x)$ denotes an element from F that is nearest to x (how to break the ties)

$x \mapsto fl(x)$ is called *rounding* which is monotone: $x \geq y$ implies $fl(x) \geq fl(y)$

overflows if $fl(x) > \max\{|y| : y \in F\}$ and *underflows* if $0 < |fl(x)| \leq \min\{|y| : 0 \neq y \in F\}$

Computing with data

- Given $x \in \mathbb{R}$, $fl(x)$ denotes an element from F that is nearest to x (how to break the ties)
- $x \mapsto fl(x)$ is called *rounding* which is monotone: $x \geq y$ implies $fl(x) \geq fl(y)$
- *overflows* if $fl(x) > \max\{|y| : y \in F\}$ and *underflows* if $0 < |fl(x)| \leq \min\{|y| : 0 \neq y \in F\}$

Example

$\rightarrow$ Let $\beta = 10$, $t = 3$, $e_{\text{min}} = -3$, $e_{\text{max}} = 3$. Then setting $a = 0.111 \times 10^3$, $b = 0.120 \times 10^3$, $c = a \times b = 0.133 \times 10^5$ is overflow

Computing with data

Given $x \in \mathbb{R}$, $fl(x)$ denotes an element from F that is nearest to x (how to break the ties)

$x \mapsto fl(x)$ is called *rounding* which is monotone: $x \geq y$ implies $fl(x) \geq fl(y)$

overflows if $fl(x) > \max\{|y| : y \in F\}$ and *underflows* if $0 < |fl(x)| \leq \min\{|y| : 0 \neq y \in F\}$

Example

- → Let $\beta = 10$, $t = 3$, $e_{\min} = -3$, $e_{\max} = 3$. Then setting $a = 0.111 \times 10^3$, $b = 0.120 \times 10^3$, $c = a \times b = 0.133 \times 10^5$ is overflow
- → Let $\beta = 10, t = 3, e_{\min} = -2, e_{\max} = 3$. Then setting $a = 0.1 \times 10^{-1}$, $b = 0.2 \times 10^{-1}$, $c = a \times b = 2 \times 10^{-4}$ is underflow

Theorem If $x \in \mathbb{R}$ lies in the range of F then

$$fl(x) = x(1+\delta),\ |\delta| < u,$$

where $u = \frac{1}{2}\beta^{1-t}$ is called the *unit roundoff*.

Theorem If $x \in \mathbb{R}$ lies in the range of F then

$$fl(x) = x(1+\delta),\ |\delta| < u,$$

where $u = \frac{1}{2}\beta^{1-t}$ is called the *unit roundoff*.

Thus

$$\text{The relative error} = \frac{|fl(x) - x|}{|x|} \leq \frac{1}{2}\beta^{1-t}$$

Theorem If $x \in \mathbb{R}$ lies in the range of F then

$$fl(x) = x(1+\delta),\ |\delta| < u,$$

where $u = \frac{1}{2}\beta^{1-t}$ is called the *unit roundoff*.

Thus

$$\text{The relative error} = \frac{|fl(x) - x|}{|x|} \leq \frac{1}{2}\beta^{1-t}$$

IEEE standard arithmetic $\beta = 2$ and support two precisions.

Single precision: $t = 24$, $e_{\min} = -125$, $e_{\max} = 128$, $u = 2^{-24} \approx 5.96 \times 10^{-8}$

Theorem If $x \in \mathbb{R}$ lies in the range of F then

$$fl(x) = x(1+\delta),\ |\delta| < u,$$

where $u = \frac{1}{2}\beta^{1-t}$ is called the *unit roundoff*.

Thus

$$\text{The relative error} = \frac{|fl(x) - x|}{|x|} \leq \frac{1}{2}\beta^{1-t}$$

IEEE standard arithmetic $\beta = 2$ and support two precisions.

Single precision: $t = 24$, $e_{\text{min}} = -125$, $e_{\text{max}} = 128$,
$u = 2^{-24} \approx 5.96 \times 10^{-8}$

Double precision: $t = 53$, $e_{\text{min}} = -1021$, $e_{\text{max}} = 1024$,
$u = 2^{-53} \approx 1.11 \times 10^{-16}$

Computing with data

Floating point format

Type	Size	Significant	Exponent	Range
Single	32 bits	23+1 bits	8 bits	$10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$10^{\pm 308}$

Computing with data

Floating point format

Type	Size	Significant	Exponent	Range
Single	32 bits	23+1 bits	8 bits	$10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$10^{\pm 308}$

Standard model for floating point arithmetic

$$fl(x \circ y) = (x \circ y)(1+\delta), \ |\delta| \le u, \circ = +, -, *, /$$

Computing with data

Floating point format

Type	Size	Significant	Exponent	Range
Single	32 bits	23+1 bits	8 bits	$10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$10^{\pm 308}$

Standard model for floating point arithmetic

$$fl(x \circ y) = (x \circ y)(1+\delta),\ |\delta| \leq u,\ \circ = +, -, *, /$$

Flops The cost of a numerical algorithm is measured in flops. A flop is an elementary floating point operation $\circ$. When we say an algorithm requires $2n^3/3$ flops, we mean $2n^3/3 + O(n^2)$ flops

Computing with data

Sources of errors in the data

$\rightarrow$ errors from measurement or estimation of real world data

Computing with data

Sources of errors in the data

$\rightarrow$ errors from measurement or estimation of real world data

$\rightarrow$ error in storing data on a computer (tiny)

Computing with data

Sources of errors in the data

- $\rightarrow$ errors from measurement or estimation of real world data
- $\rightarrow$ error in storing data on a computer (tiny)
- $\rightarrow$ result of errors (big or small) in an earlier computation if the data is a solution of another problem

Computing with data

Sources of errors in the data

- $\rightarrow$ errors from measurement or estimation of real world data
- $\rightarrow$ error in storing data on a computer (tiny)
- $\rightarrow$ result of errors (big or small) in an earlier computation if the data is a solution of another problem
- $\rightarrow$ truncation errors

Computing with data

Sources of errors in the data

- $\rightarrow$ errors from measurement or estimation of real world data
- $\rightarrow$ error in storing data on a computer (tiny)
- $\rightarrow$ result of errors (big or small) in an earlier computation if the data is a solution of another problem
- $\rightarrow$ truncation errors

Precision and accuracy

- $\rightarrow$ Accuracy refers to the absolute or relative error of an approximate quantity

Computing with data

Sources of errors in the data

- → errors from measurement or estimation of real world data
- → error in storing data on a computer (tiny)
- → result of errors (big or small) in an earlier computation if the data is a solution of another problem
- → truncation errors

Precision and accuracy

- → Accuracy refers to the absolute or relative error of an approximate quantity
- → Precision is the accuracy with which the basic arithmetic operations are performed

Computing with data

Backward error - How to measure the quality of a solution?

Computing with data

Backward error - How to measure the quality of a solution?

Let f be a problem with input x and $\widehat{y}$ be a computed solution. Then we ask:

Computing with data

Backward error - How to measure the quality of a solution?

Let f be a problem with input x and $\widehat{y}$ be a computed solution. Then we ask:

For what set of data have we solved the problem?

Computing with data

Backward error - How to measure the quality of a solution?

Let f be a problem with input x and $\widehat{y}$ be a computed solution. Then we ask:

For what set of data have we solved the problem? i.e. for what $\triangle x$, do we have

$$\widetilde{y} = f(x + \triangle x)?$$

Computing with data

Backward error - How to measure the quality of a solution?

Let f be a problem with input x and $\widehat{y}$ be a computed solution. Then we ask:

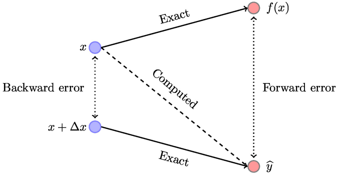
For what set of data have we solved the problem? i.e. for what $\triangle x$, do we have

$$\widetilde{y} = f(x + \triangle x)?$$

There can be many such $\triangle x$, we look for the smallest one. The

$$\min |\triangle x| \text{ such that } \widehat{y} = f(x + \triangle x)$$

is called the backward error of the solution.



Computing with data

Forward error The error of $\widehat{y}$ is called the forward error i.e.

$$|\widehat{y} - f(x)|$$

Computing with data

Forward error The error of $\widehat{y}$ is called the forward error i.e.

$$|\widehat{y} - f(x)|$$

Backward stability A method for computing $y = f(x)$ is called backward stable if, for any x, it produces a computed $\widehat{y}$ with a small backward error i.e. $\widehat{y} = f(x + \triangle x)$ for some small $\triangle x$

Computing with data

Forward error The error of $\widehat{y}$ is called the forward error i.e.

$$|\widehat{y} - f(x)|$$

Backward stability A method for computing $y = f(x)$ is called backward stable if, for any x, it produces a computed $\widehat{y}$ with a small backward error i.e. $\widehat{y} = f(x + \triangle x)$ for some small $\triangle x$ (how small is small?)

Computing with data

Forward error The error of $\widehat{y}$ is called the forward error i.e.

$$|\widehat{y} - f(x)|$$

Backward stability A method for computing $y = f(x)$ is called backward stable if, for any x, it produces a computed $\widehat{y}$ with a small backward error i.e. $\widehat{y} = f(x + \triangle x)$ for some small $\triangle x$ (how small is small?)

$\rightarrow$ the operation $x \pm y$ is the exact result for a perturbed data $x(1+\delta)$ and $y(1+\delta)$ with $|\delta| \leq u$, thus by definition addition and subtraction are are backward stable operations

Computing with data

Forward error The error of $\widehat{y}$ is called the forward error i.e.

$$|\widehat{y} - f(x)|$$

Backward stability A method for computing $y = f(x)$ is called backward stable if, for any x, it produces a computed $\widehat{y}$ with a small backward error i.e. $\widehat{y} = f(x + \triangle x)$ for some small $\triangle x$ (how small is small?)

$\rightarrow$ the operation $x \pm y$ is the exact result for a perturbed data $x(1+\delta)$ and $y(1+\delta)$ with $|\delta| \leq u$, thus by definition addition and subtraction are are backward stable operations

Example An algorithm for solving $Ax = b$ is called backward stable if the computed solution $\widehat{x}$ is such that

$$(A + \triangle A)\widehat{x} = b + \triangle b$$

with small $\triangle A$ and $\triangle b$ (in terms of norm of course)

Computing with data

Gaussian elimination without pivoting is unstable! Let

$$A = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Apply GE without pivoting:

$$[A|b] \mapsto \begin{bmatrix} 10^{-10} & 1 & 1 \\ 0 & 2 - 10^{10} & 3 - 10^{10} \end{bmatrix}$$

Computing with data

Gaussian elimination without pivoting is unstable! Let

$$A = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Apply GE without pivoting:

$$[A|b] \mapsto \begin{bmatrix} 10^{-10} & 1 & 1 \\ 0 & 2 - 10^{10} & 3 - 10^{10} \end{bmatrix}$$

Check in computer that it would give the solution $x_2 = 1, x_1 = 0$, whereas the solution is supposed to be $x_1 = 1 = x_2$?

Computing with data

Gaussian elimination without pivoting is unstable! Let

$$A = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Apply GE without pivoting:

$$[A|b] \mapsto \begin{bmatrix} 10^{-10} & 1 & 1 \\ 0 & 2 - 10^{10} & 3 - 10^{10} \end{bmatrix}$$

Check in computer that it would give the solution $x_2 = 1, x_1 = 0$, whereas the solution is supposed to be $x_1 = 1 = x_2$? What are the $\triangle A$ and $\triangle b$ here?

Computing with data

Gaussian elimination without pivoting is unstable! Let

$$A = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Apply GE without pivoting:

$$[A|b] \mapsto \begin{bmatrix} 10^{-10} & 1 & 1 \\ 0 & 2 - 10^{10} & 3 - 10^{10} \end{bmatrix}$$

Check in computer that it would give the solution $x_2 = 1, x_1 = 0$, whereas the solution is supposed to be $x_1 = 1 = x_2$? What are the $\triangle A$ and $\triangle b$ here? What happens if you use pivoting?

Computing with data

Question Are these forward error and backward error related?

Computing with data

Question Are these forward error and backward error related?

Suppose $y = f(x)$ is problem and the approximate solution is $\widehat{y} = f(x + \triangle x)$.

Computing with data

Question Are these forward error and backward error related?

Suppose $y = f(x)$ is problem and the approximate solution is $\widehat{y} = f(x + \triangle x)$. Suppose f is twice differentiable function. Then

$$\widehat{y} - y \;\; = \;\; f(x + \triangle x) - f(x)$$

Computing with data

Question Are these forward error and backward error related?
Suppose $y = f(x)$ is problem and the approximate solution is $\widehat{y} = f(x + \triangle x)$. Suppose f is twice differentiable function. Then

$$\begin{aligned} \widehat{y} - y &= f(x + \triangle x) - f(x) \\ &= f'(x)\triangle x + \frac{f''(x + \theta \triangle x)}{2!}(\triangle x)^2, \ \theta \in (0, 1) \qquad (1) \end{aligned}$$

Computing with data

Question Are these forward error and backward error related?
Suppose $y = f(x)$ is problem and the approximate solution is $\widehat{y} = f(x + \triangle x)$. Suppose f is twice differentiable function. Then

$$\begin{aligned} \widehat{y} - y &= f(x + \triangle x) - f(x) \\ &= f'(x)\triangle x + \frac{f''(x + \theta\triangle x)}{2!}(\triangle x)^2, \ \theta \in (0, 1) \end{aligned} \tag{1}$$

Then

$$\frac{\widehat{y} - y}{y} = \left(\frac{xf'(x)}{f(x)}\right)\frac{\triangle x}{x} + O((\triangle x)^2).$$

Computing with data

Question Are these forward error and backward error related?
Suppose $y = f(x)$ is problem and the approximate solution is $\widehat{y} = f(x + \triangle x)$. Suppose f is twice differentiable function. Then

$$\begin{aligned} \widehat{y} - y &= f(x + \triangle x) - f(x) \\ &= f'(x)\triangle x + \frac{f''(x + \theta \triangle x)}{2!}(\triangle x)^2, \ \theta \in (0,1) \end{aligned} \tag{1}$$

Then

$$\frac{\widehat{y} - y}{y} = \left(\frac{xf'(x)}{f(x)}\right)\frac{\triangle x}{x} + O((\triangle x)^2).$$

The quantity

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

measures, for small $\triangle x$, the relative change in the output for a relative change in the input. It is called (relative) condition number of the problem f.

Computing with data

Therefore

$$\text{forward error} \quad \leq \quad \text{condition number} \quad \times \quad \text{backward error}$$

Computing with data

Therefore

$$\text{forward error} \;\leq\; \text{condition number} \;\times\; \text{backward error}$$

We conclude

$\rightarrow$ For a well-condition problem, if the backward error is high then forward error is high

Computing with data

Therefore

$$\text{forward error} \quad \leq \quad \text{condition number} \quad \times \quad \text{backward error}$$

We conclude

- $\rightarrow$ For a well-condition problem, if the backward error is high then forward error is high
- $\rightarrow$ For an ill conditioned problem, if the backward error is small then the forward error is high

Computing with data

Therefore

$$\text{forward error} \quad \leq \quad \text{condition number} \quad \times \quad \text{backward error}$$

We conclude

→ For a well-condition problem, if the backward error is high then forward error is high

→ For an ill conditioned problem, if the backward error is small then the forward error is high

→ The forward error is small only when the backward error is small and the problem is well conditioned

Computing with data

Well/ill conditioned problems A problem is called ill-conditioned if a small deviation of the input data cause large relative error in the computed solution, regardless of the method for the solution. Otherwise, it is called well-conditioned

Computing with data

Well/ill conditioned problems A problem is called ill-conditioned if a small deviation of the input data cause large relative error in the computed solution, regardless of the method for the solution. Otherwise, it is called well-conditioned

Condition number of a problem If f is a problem with respect to the data x then the condition number of f is

$$\frac{\text{relative error in the solution}}{\text{relative perturbation in the data}} = \frac{\frac{|f(x)-f(y)|}{|f(x)|}}{\frac{|x-y|}{|x|}}$$

Computing with data

Well/ill conditioned problems A problem is called ill-conditioned if a small deviation of the input data cause large relative error in the computed solution, regardless of the method for the solution. Otherwise, it is called well-conditioned

Condition number of a problem If f is a problem with respect to the data x then the condition number of f is

$$\frac{\text{relative error in the solution}}{\text{relative perturbation in the data}} = \frac{\frac{|f(x)-f(y)|}{|f(x)|}}{\frac{|x-y|}{|x|}}$$

A mathematical definition Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a problem. Then the condition number of f is

$$\lim_{\epsilon \to 0} \sup_{\|\triangle x\| \leq \epsilon \|x\|} \frac{\|f(x+\triangle x) - f(x)\|}{\epsilon \|f(x)\|}$$

Computing with data

Example of an ill-conditioned linear system Let $Ax = b$ with

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4.0001 & 2.002 \\ 1 & 2.002 & 2.004 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 8.0021 \\ 5.006 \end{bmatrix}.$$

Computing with data

Example of an ill-conditioned linear system Let $Ax = b$ with

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4.0001 & 2.002 \\ 1 & 2.002 & 2.004 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 8.0021 \\ 5.006 \end{bmatrix}.$$

The exact solution is $x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Computing with data

Example of an ill-conditioned linear system Let $Ax = b$ with

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4.0001 & 2.002 \\ 1 & 2.002 & 2.004 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 8.0021 \\ 5.006 \end{bmatrix}.$$

The exact solution is $x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Change b to $b' = \begin{bmatrix} 4 \\ 8.0020 \\ 5.0061 \end{bmatrix}$. Then the relative change is:

$$\frac{\|b' - b\|}{\|b\|} = \frac{\|\triangle b\|}{\|b\|} = 1.3795 \times 10^{-5}.$$

Computing with data

Example of an ill-conditioned linear system Let $Ax = b$ with

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4.0001 & 2.002 \\ 1 & 2.002 & 2.004 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 8.0021 \\ 5.006 \end{bmatrix}.$$

The exact solution is $x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Change b to $b' = \begin{bmatrix} 4 \\ 8.0020 \\ 5.0061 \end{bmatrix}$. Then the relative change is:

$$\frac{\|b' - b\|}{\|b\|} = \frac{\|\triangle b\|}{\|b\|} = 1.3795 \times 10^{-5}.$$

Solving the system $Ax' = b'$ we have $x' = \begin{bmatrix} 3.0850 \\ -0.0436 \\ 1.0022 \end{bmatrix}$.

Computing with data

Condition number of a matrix - the most important notion dealing with matrix computations

Let $A = \begin{bmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{bmatrix}$ and $b = \begin{bmatrix} 4.1 \\ 9.7 \end{bmatrix}$. Then $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is the solution of $Ax = b$.

Computing with data

Condition number of a matrix - the most important notion dealing with matrix computations

Let $A = \begin{bmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{bmatrix}$ and $b = \begin{bmatrix} 4.1 \\ 9.7 \end{bmatrix}$. Then $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is the solution of $Ax = b.$

Now consider $b' = \begin{bmatrix} 4.11 \\ 9.7 \end{bmatrix}$. Then solving $Ax = b'$ in MATLAB gives $x = \begin{bmatrix} 0.3400 \\ 0.9700 \end{bmatrix}$!!

Computing with data

Condition number of a matrix - the most important notion dealing with matrix computations

Let $A = \begin{bmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{bmatrix}$ and $b = \begin{bmatrix} 4.1 \\ 9.7 \end{bmatrix}$. Then $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is the solution of $Ax = b$.

Now consider $b' = \begin{bmatrix} 4.11 \\ 9.7 \end{bmatrix}$. Then solving $Ax = b'$ in MATLAB gives $x = \begin{bmatrix} 0.3400 \\ 0.9700 \end{bmatrix}$!!

Question Why??