

Preliminaries to Numerical Analysis

- Numerical linear algebra
 - square linear systems: $A\vec{x} = \vec{b}$
 - overdetermined linear systems: $A\vec{x} = \vec{b}$
 - spectral theory: $A\vec{x} = \lambda\vec{x}$
(eigenvalues, singular values)
- Nonlinear
 - root finding
 - piecewise polynomial interp.
 - num. calculus (diff, integ. ODE theory)

Contents

Main Sources of errors.

- ① Floating-Point Numbers (how #'s are represented / stored on computer)
- ② Conditioning (inherent property of a problem)
- ③ Stability (algorithm)

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**:

$$e_{\text{abs}} = A_{\text{approx}} - A_{\text{exact}}$$

(absolute error)

$$e_{\text{rel}} = \frac{A_{\text{approx}} - A_{\text{exact}}}{A_{\text{exact}}} = \frac{A_{\text{approx}}}{A_{\text{exact}}} - 1,$$

(relative error)

"percentage"
"dimensionless"

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**.

Example: Stirling's Formula

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

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

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)
```

When $n=10$:

Abs. err. = -30104....

Rel. err. = -0.008296 (0.8%)

When $n=100$:

Abs. err. = $-7.77... \times 10^{154}$

Rel. err. = -0.00083298 (0.08%)

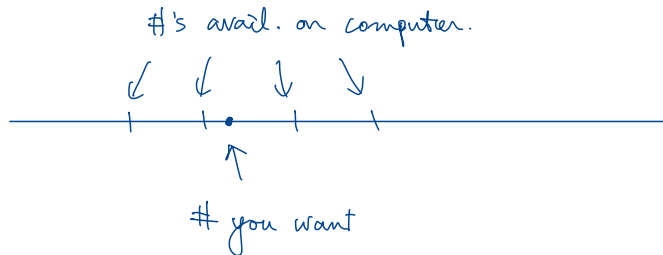
Floating-Point Numbers

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} .



Scientific notation (base 10)

one digit before dec. pt. ; no zero here.

$$\boxed{3}.14 \times 10^2 = 314$$

$$3.14 \times 10^{-3} = 0.00314$$

$$3.14 \times 10^2$$

$$= (3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}) \times 10^2$$

$$= 3 \times 10^2 + 1 \times 10^1 + 4 \times 10^0$$

$$= 314.$$

Scientific notation (base 2)

one bit here ; not zero.

$$\boxed{1}.101_{(2)} \times 2^{\boxed{10_{(2)}}}$$

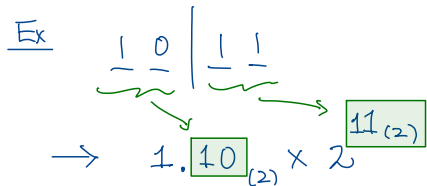
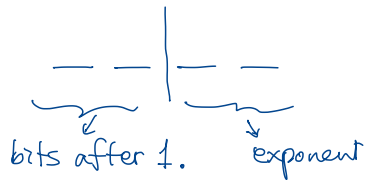
$$= (1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) \times 2^2$$

$$= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$$

$$\downarrow \quad (= 110.1_{(2)})$$

$$= 4 + 2 + 0 + \frac{1}{2} = 6.5$$

Rough idea (4 bits)



* $00|00 : 1.00_{(2)} \times 2^{0_{(2)}} = 1$

$01|00 : 1.01_{(2)} \times 2^{0_{(2)}} = 1 + \frac{1}{4}$

$10|00 : 1.10_{(2)} \times 2^{0_{(2)}} = 1 + \frac{1}{2}$

$11|00 : 1.11_{(2)} \times 2^{0_{(2)}} = 1 + \frac{1}{2} + \frac{1}{4}$

$= 1 + \frac{3}{4}.$

* $00|11 : 1.00_{(2)} \times 2^{11_{(2)}}$

$= 1 \times 2^3 = 8.$

$11|11 : 1.11_{(2)} \times 2^{11_{(2)}}$

$= (1 + \frac{3}{4}) \times 8 = 8 + 6$

$= 14$

Floating-Point Numbers

base-2 scientific notation.

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 $b_i = 1$.

fractional part

d : # of bits
after 1.

Note that F can be rewritten as

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

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

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

Floating-Point Numbers – IEEE 754 Standard

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

$$\pm \underbrace{1.\text{xxxxxxxx} \cdots \text{xxxxxxxx}}_{\text{mantissa (base 2): } 52+1 \text{ bits}} \times 2^{\underbrace{\text{xxxx} \cdots \text{xxxx}}_{\text{exponent: 11 bits}} - 1023}$$

offset

$$(-1)^s$$

- Predefined variables:

Sign bit (s)

$S=0$ for (+) ; $S=1$ for (-)

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

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

- realmin = the smallest positive floating-point number that is stroed to full accuracy; the actual smallest is $\text{realmin}/2^{52}$.
- realmax = the largest positive floating-point number

$$\begin{aligned}
 \bullet \quad \underbrace{111 \dots 1}_{11}_{(2)} &= 2^{10} + 2^9 + \dots + 2^1 + 2^0 \\
 &= \frac{2^{11} - 1}{2 - 1} = 2048 - 1 = 2047
 \end{aligned}$$

$$2047 - 1023 = \underline{1024} \leftarrow \text{"max." exp.}$$

$$\bullet \quad \underbrace{00 \dots 0}_{11}_{(2)} = 0$$

$$0 - 1023 = \underline{-1023} \leftarrow \text{"min." exp.}$$

Machine Epsilon and Relative Errors

$$2^{-52} \approx 2.2 \times 10^{-16}$$

of mantissa bits

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

gap btw 1 and its successor.

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

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

percentage error = relative error

- **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

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

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

Algebraic

$$|x - \hat{x}| \leq \frac{1}{2} 2^E \boxed{\text{eps}}$$

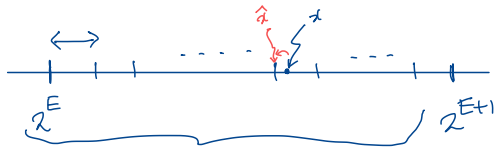
$$x \geq 2^E$$

$$\left| \frac{x - \hat{x}}{x} \right| \leq \frac{\frac{1}{2} 2^E \boxed{\text{eps}}}{2^E}$$
$$= \frac{1}{2} \boxed{\text{eps}}$$

Visual

$$2^E \leq x < 2^{E+1}$$

$$-1022 \leq E \leq 1023$$



$$\# \text{ of f-p \#s} = 2^E$$

$$\text{Gap} = 2^E \boxed{\text{eps}}$$

$$= \frac{\text{Width}}{\# \text{ of pts}} = \frac{2^{E+1} - 2^E}{2^{E+1}}$$

$$= 2^E \cdot 2^{-E+1} = 2^E \boxed{\text{eps}}$$

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
(1 + .5*eps) - 1
(1 + .51*eps) - 1
n = input(' n = '); ( n^(1/3) )^3 - n
```

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:

$$= e^x \left(\frac{e^{-x}}{2} + \frac{e^{-x}}{2} \right)$$
$$= 1$$

NaN (not a number)

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

$$\cosh x = \frac{e^x + e^{-x}}{2}$$

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

For large x ,

$$\cosh x \approx \sinh x$$

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?

Errors A_{exact} , A_{approx} .

$$\left\{ \begin{array}{l} \text{Abs. err.} = A_{\text{exact}} - A_{\text{approx}} \end{array} \right.$$

$$\left\{ \begin{array}{l} \text{Rel. err.} = \frac{A_{\text{exact}} - A_{\text{approx}}}{A_{\text{exact}}} \end{array} \right.$$

(percentage error; dimensionless)

Floating-point numbers (F)

- base-2 scientific notation

$$\underbrace{\pm 1}_{1} \cdot \underbrace{XX \dots X}_{52}_{(2)} \times 2^{\underbrace{X \dots X}_{11}_{(2)} - 1023} \quad \left. \vphantom{\pm 1} \right\} 64 \text{ bits}$$

$$-1022 \leq E \leq 1023$$

abstractly written as $\pm (1 + \textcircled{F}) 2^{\textcircled{E}}$

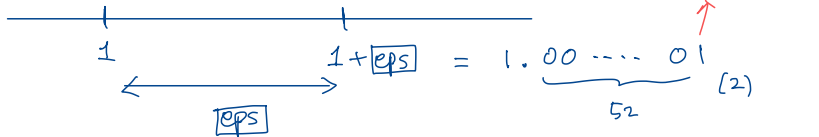
$$0 \leq F < 1$$

- machine epsilon (code / writing)

$$\bullet \quad \boxed{\text{eps}} = 2^{-52} \approx 2.22 \times 10^{-16}$$

• Significance?

1. Gap btw 1 and its successor.



2.
$$\left| \frac{x - \hat{x}}{x} \right| \leq \frac{1}{2} \boxed{\text{eps}}$$

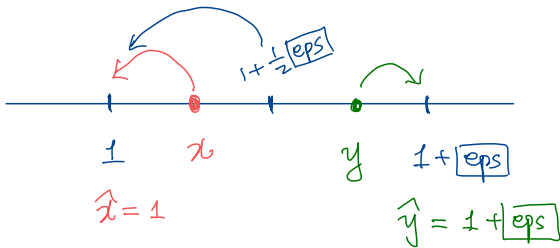
- Roundoff error.

$$>> (1 + 0.5 * \text{eps}) - 1$$

$$\text{ans} = 0$$

$$>> (1 + 0.51 * \text{eps}) - 1$$

$$\text{ans} = 2.2 \dots \text{e-16} \quad (\text{eps})$$



Convention: Midpoint \rightarrow round down

- Distribution of F-P numbers

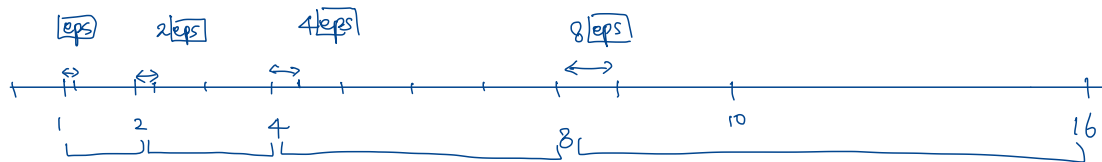
On $[2^E, 2^{E+1})$

- there are $2^d = 2^{52}$ f-p #'s
- they are uniformly spaced
- the gap btw two adjacent f-p #'s here

$$\text{is } \underline{2^E \boxed{\text{eps}}} = 2^E 2^{-d} = 2^{E-52}$$

Ex $E=0, [1, 2) : \text{Gap} = 2^0 \boxed{\text{eps}} = \boxed{\text{eps}}$

$E=1, [2, 4) : \text{Gap} = 2^1 \boxed{\text{eps}} = \underline{2} \boxed{\text{eps}} \dots$



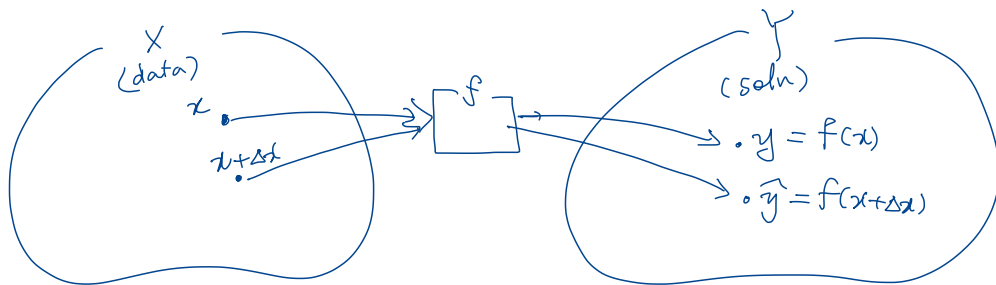
... , $10 - 8\boxed{\text{eps}}$, 10 , $10 + 8\boxed{\text{eps}}$, ...

... , $8 - 4\boxed{\text{eps}}$, 8 , $8 + 8\boxed{\text{eps}}$, ...

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 $f(x)$



Interested in magnification of input errors

Condition Number

relative error

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

$$\left| \frac{\text{rel. err. of output}}{\text{rel. err. of input}} \right| = \frac{|f(x) - f(x(1 + \epsilon))|}{|\epsilon f(x)|}.$$

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

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

"Kappa"

$\kappa(x)$ when f is clear from context.

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

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 $x \approx c$. (Catastrophic cancellation)

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. (well-condition)

Example: Conditioning of Function Evaluation

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

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

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

$(n + 1/2)\pi$
Vert. asympt. of $\tan(x)$.

Example: Conditioning of Root-Finding

$$ar^2 + br + c = 0$$

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

$$f'(a) = \frac{dr}{da}$$

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

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 (*) to get

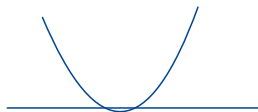
$$\kappa(a) = \left| \frac{af'(a)}{f(a)} \right| = \left| \frac{ar^2}{\pm r\sqrt{b^2 - 4ac}} \right| = \left| \frac{ar}{\sqrt{b^2 - 4ac}} \right|.$$

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

full problem

$$\vec{f} : \mathbb{R}^3 \rightarrow \mathbb{C}^2$$

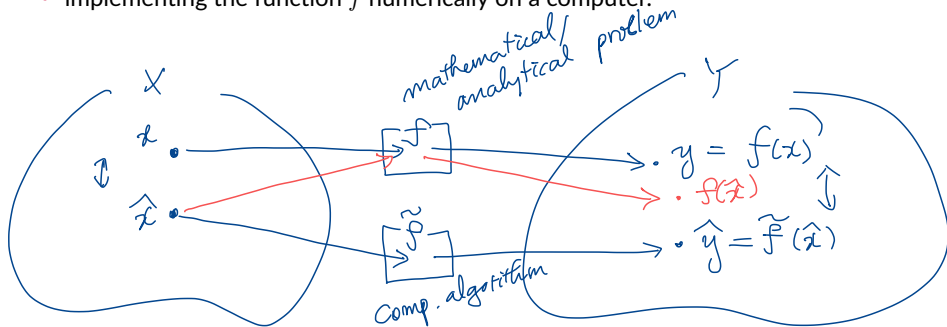
$$\vec{f}\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$



Stability

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.



Example: Horner's Method

Consider evaluating a polynomial

$$p(x) = c_n x^{n-1} + c_{n-1} x^{n-2} + \cdots + c_2 x + c_1.$$

Can rewrite it as

$$p(x) = (\cdots ((c_n x + c_{n-1})x + c_{n-2})x + \cdots + c_2)x + c_1,$$

```
function p = horner(c, x)
% HORNER evaluates polynomial using Horner's method.
    n = length(c);
    p = c(n);
    for k = n-1:-1:1
        p = p*x + c(k);
    end
end
```

Analysis: General Framework

The relative error of our interest is

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

where $\kappa = \kappa_f$ be the (relative) condition number of the exact problem f and
effect of using \tilde{f} rather than f *effect of perturbed input passed into f*

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

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 $r_2 = (c/a)/r_1$.