

# CS381: Numerical Computation & Softwares

## Error Analysis

Jakramate Bootkrajang

Department of Computer Science

Chiang Mai University

Rev 2 – 2019

“Theory is the first term in the Taylor series of practice”

- *Thomas M. Cover* -

เนื้อหาในบทนี้เราจะเรียนรู้เกี่ยวกับ

- ความจริงที่ว่าคอมพิวเตอร์สามารถเก็บตัวเลขที่มีความละเอียดจำกัดเท่านั้น
- มาตรฐาน IEEE สำหรับการเก็บตัวเลขแบบมีทศนิยมในคอมพิวเตอร์
- Numerical errors และ condition numbers

# การคำนวณทางวิทยาศาสตร์บนคอมพิวเตอร์

การคำนวณพื้นฐานดังตัวอย่างเหล่านี้ถูกใช้เป็นประจำ

- Example 1:  $8 + 8 = 16$  ?
- Example 2:  $(\sqrt{5})^2 = 5$  ?
- Example 3:  $1.1 + 0.1 = 1.2$  ?

ฝั่งซ้ายของสมการเท่ากับฝั่งขวาของสมการหรือไม่ – ตามหลักการแล้วควรจะเท่ากัน

# การคำนวณทางวิทยาศาสตร์บนคอมพิวเตอร์

ลองทำใน JULIA ดู

Let's try this with JULIA:

```
julia>(1.1 + 0.1) == 1.2
```

```
false
```

```
julia> 1.1 + 0.1
```

```
1.20000000000000002
```

Problem: numerical error:  $\approx 2 * 10^{-16}$ .

Why ?

การกระทำทางคณิตศาสตร์ทำโดยบวกเลขโดดที่เป็นตัวแทนของตัวเลข

ไม่ได้เอาตัวเลขบวกกลับกันโดยตรง

# Number representation in a computer

- จำนวนเต็ม (Integers)
  - ▶ A number that can be written without a fractional component, e.g., 1, -5 ,0 102
  - ▶ คอมพิวเตอร์เก็บจำนวนเต็มโดยแปลงตัวเลขให้อยู่ในรูปของเลขฐาน 2 โดยตรง
- จำนวนที่อยู่ในรูปของทศนิยม (floating-point numbers)
  - ▶ เก็บในรูปแบบของ Scientific notation ได้แก่  $0.xxx \times B^e$
  - ▶ จำเป็นต้องมีมาตรฐานในการจัดเก็บตัวเลขเหล่านี้ (IEEE standard)
- Try this in JULIA: `bits(3)` and `bits(3.0)` — ฟังก์ชัน `bits()` ใช้เรียกดู number representation ในหน่วยความจำ

# IEEE standard for Floating Point Numbers

IEEE standard for double-precision (64 bits) floating point numbers:

$$x = \pm(1 + m) \cdot 2^e \quad \text{with} \quad m = \sum_{i=1}^{52} m_i 2^{-i} \quad \text{and} \quad e = \sum_{i=0}^{10} c_i 2^i - \bar{c}$$

$m$  เรียกว่า mantissa และ  $e$  เรียกว่า exponent

เนื้อที่หน่วยความจำที่ต้องใช้ในการเก็บตัวเลข:

- 1 bit สำหรับเครื่องหมาย ค่าบวกแทนด้วยเลข 0 ค่าลบแทนด้วยเลข 1
- 11 bits สำหรับ exponent;  $\bar{c}$  เรียกว่า offset มีค่าเท่ากับ 1023
- 52 bits สำหรับ mantissa

รวมทั้งหมด  $1 + 11 + 52 = 64$  bits หรือ 8 bytes.





## A closer look at the example

```
julia>bits(1.1)
```

```
"001111111110001100110011001100110011001100110011001100110011010"
```

```
julia>bits(0.1)
```

```
"00111111011100110011001100110011001100110011001100110011010"
```

```
julia>bits(1.2)
```

```
"00111111111001100110011001100110011001100110011001100110011"
```

```
julia>bits(1.1+0.1)
```

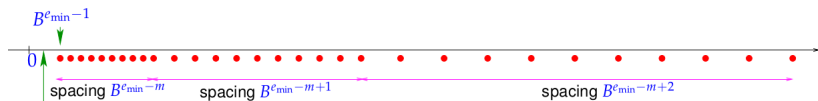
```
"00111111111001100110011001100110011001100110011001100110100"
```

# Some properties of (double-precision) floating point numbers

$$x = \pm(1 + m) \cdot 2^e$$

- เนื่องจากจำนวนบิตมีจำกัด คอมพิวเตอร์สามารถแสดงค่าตัวเลขได้บางตัวเท่านั้น
  - ▶ เซทของค่าที่แสดงได้  $M_I$  เป็นเซทย่อยของเซทของจำนวนจริง  $M \subset \mathbb{R}$
- ขอบเขตของตัวเลข
  - ▶ ตัวเลขที่มี  $2^e$  น้อยกว่า  $2^{-1022}$  ถือว่ามีค่าเท่ากับ 0
  - ▶ ตัวเลขที่มีค่ามากกว่า  $2^{1023}(2 - 2^{-52})$  ถือว่าเกินขอบเขต (overflow) (แทนด้วย Inf)
- ถ้าค่าของตัวเลขไม่ overflow นั่นคือไม่เกิน  $2^{1023}(2 - 2^{-52})$ , ตัวเลข นั้นจะถูกปัด (round) ให้เป็นค่าที่ใกล้ที่สุดที่คอมพิวเตอร์สามารถแสดงได้
  - ▶ ตัวอย่างเช่น 1.19999999999999996 จะแสดงเป็น 1.2

# Floating Point Numbers distribution



- ตัวเลขที่มีค่า  $1 + m$  ระหว่าง 1 ถึง  $1 + 2^{-52}$  ไม่สามารถแสดงได้บนคอมพิวเตอร์
- The (relative) rounding error,  $\text{eps} = 2^{-52}$ , is called *machine precision*.
- The absolute rounding error,  $\text{eps} * 2^e$ , depends on exponent  $e$ .

(ถ้าเราทำงานกับตัวเลขขนาดใหญ่ เราจะต้องประสบปัญหาของ rounding errors ที่มากขึ้น)

- Important eps (ย่อมาจาก *epsilon*) มีค่าเท่ากับ  $2^{-52} \approx 2 \times 10^{-16}$

# Numerical instability

การคำนวณทางวิทยาศาสตร์ อาจไม่ได้ผลลัพธ์ตามที่เรต้องการ เนื่องจากปัญหาต่อไปนี้

- Roundoff error
- Overflow
- Cancellation

- ผลลัพธ์จากการคำนวณอาจถูกปัดเป็นตัวเลขที่คอมพิวเตอร์สามารถแสดงได้
- หากมีการคำนวณหลายขั้นตอน rounding error อาจสะสมจนเกิดปัญหา
  - ▶ เช่น การทดสอบ `if(x == s)` อาจอันตรายถ้า  $x$  เป็นผลลัพธ์ของการคำนวณ เพราะ  $x$  อาจไม่มีโอกาสมีค่าเท่ากับ  $s$  เลย
  - ▶ แต่ควรใช้การทดสอบว่า  $x$  อยู่ในช่วงที่ห่างจาก  $s$  ไม่เกิน machine precision

```
if(abs(x-s) < eps)
```

- ผลลัพธ์จากการคำนวณอาจเกินขอบเขตที่คอมพิวเตอร์สามารถแสดงค่าได้
- แก้ไขโดยการจัดลำดับการคำนวณใหม่ เช่น การคำนวณขนาดเวกเตอร์ในสองมิติ

$$r = \sqrt{x^2 + y^2}$$

straightforward evaluation: overflow, when  $|x| > \sqrt{\max |\mathbb{M}|}$  or  $|y| > \sqrt{\max |\mathbb{M}|}$ .

$$r = \begin{cases} |x| \sqrt{1 + (y/x)^2} & , \text{ if } |x| \geq |y| , \\ |y| \sqrt{1 + (x/y)^2} & , \text{ if } |y| > |x| . \end{cases}$$

➤ no overflow!

# Cancellation

- **Cancellation or Loss of significance** occurs in numerical calculations when too many significant digits cancel.
- ส่วนมากเกิดขึ้นเมื่อเรานำตัวเลขสองตัวที่ใกล้เคียงกันมาลบกัน
- สมมติว่าตัวเลขทั้งสองมีความแม่นยำถึงหลักที่ 6 การลบกันของตัวเลขส่งผลให้ หลักที่แม่นยำหักล้างกันไป ส่งผลให้มีหลักที่สำคัญน้อยเข้ามาอยู่ในผลลัพธ์

$$\begin{array}{rcl} x & = & 0.123467* & \leftarrow \text{7th digit perturbed} \\ y & = & 0.123456* & \leftarrow \text{7th digit perturbed} \\ \hline x - y & = & 0.000011* = 0.11*000 \cdot 10^{-4} & \leftarrow \text{3rd digit perturbed} \end{array}$$

# Cancellation Example

- เราทราบว่าอนุพันธ์ของฟังก์ชัน  $f$  ณ จุด  $x$  มีค่าเท่ากับลิมิต

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (1)$$

- ลองเขียนโปรแกรมเพื่อคำนวณอนุพันธ์ของ  $e^x$  ณ จุดที่  $x = 0$  เทียบกับอนุพันธ์ที่ได้จากการคำนวณเชิงสัญลักษณ์แล้วดู error ที่เกิดขึ้น
  - ▶ ถ้ายังพอจำได้ อนุพันธ์ของ  $\frac{de^x}{dx} = e^x$
- คาดว่าถ้า  $h \rightarrow 0$  ค่าที่ประมาณได้น่าจะเข้าใกล้คำตอบจริงมากขึ้น



# Cancellation Example

โค้ด Julia ที่ใช้ในการทดลอง

```
# cancellation errors during derivative approximation

f(x) = exp(x) # the exponential function
df(x) = exp(x) # its derivative (analytic)

@printf("%3s %16s %16s %16s\n", "h", "Exact", "Approx", "Error")
for i=1:16
    h      = 1/(10^i)
    approx = (f(0+h)-f(0))/h # approx derivative at x=0
    exact  = df(0)           # exact derivative from calculus
    err    = abs(exact-approx)/abs(exact) # relative error
    @printf("%.1e, %.16f, %.16f, %.16f\n", h, exact, approx, err )
end
```

# Cancellation Example

สังเกตว่าเมื่อ  $h$  มีค่าน้อยลงเกินกว่า  $10^{-8}$  ค่าที่ได้จากการประมาณกลับทำให้เกิด error ที่มากขึ้น  
ทั้งนี้เกิดจาก  $f(x+h)$  และ  $f(x)$  มีค่าใกล้เคียงกันมาก ทำให้เกิด cancellation error ขึ้น

h	Exact	Approx	Error
1.0e-01,	1.0000000000000000,	1.0517091807564771,	0.0517091807564771
1.0e-02,	1.0000000000000000,	1.0050167084167949,	0.0050167084167949
1.0e-03,	1.0000000000000000,	1.0005001667083846,	0.0005001667083846
1.0e-04,	1.0000000000000000,	1.0000500016671410,	0.0000500016671410
1.0e-05,	1.0000000000000000,	1.0000050000069649,	0.0000050000069649
1.0e-06,	1.0000000000000000,	1.0000004999621837,	0.0000004999621837
1.0e-07,	1.0000000000000000,	1.0000000494336803,	0.0000000494336803
1.0e-08,	1.0000000000000000,	0.9999999939225290,	0.0000000060774710
1.0e-09,	1.0000000000000000,	1.0000000827403710,	0.0000000827403710
1.0e-10,	1.0000000000000000,	1.0000000827403710,	0.0000000827403710
1.0e-11,	1.0000000000000000,	1.0000000827403710,	0.0000000827403710
1.0e-12,	1.0000000000000000,	1.0000889005823410,	0.0000889005823410
1.0e-13,	1.0000000000000000,	0.9992007221626409,	0.0007992778373591
1.0e-14,	1.0000000000000000,	0.9992007221626409,	0.0007992778373591
1.0e-15,	1.0000000000000000,	1.1102230246251565,	0.1102230246251565
1.0e-16,	1.0000000000000000,	0.0000000000000000,	1.0000000000000000

# Real world Numerical Catastrophes

## Numerical Catastrophes

**Ariane 5 rocket.** [June 4, 1996]

- 10 year, \$7 billion ESA project exploded after launch.
- 64-bit float converted to 16 bit signed int.
- Unanticipated overflow.



Copyright, Arianespace

**Vancouver stock exchange.** [November, 1983]

- Index undervalued by 44%.
- Recalculated index after each trade by adding **change** in price.
- 22 months of accumulated truncation error.

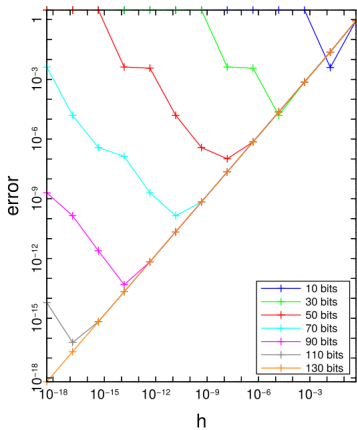
**Patriot missile accident.** [February 25, 1991]

- Failed to track scud; hit Army barracks, killed 28.
- Inaccuracy in measuring time in 1/20 of a second since using 24 bit binary floating point.



# Avoiding cancellation

ขยายจำนวนบิตในการเก็บข้อมูล (impractical)



# Avoiding cancellation

- การหลีกเลี่ยงผลกระทบจาก cancellation error สามารถทำได้โดยการเขียน expressions ในรูปแบบใหม่ที่ได้รับผลจาก error น้อยกว่า
- หากเป็นไปได้ ให้พยายามเขียน expression ที่ไม่ต้องทำการลบเลขสองตัวที่มีค่าใกล้เคียงกัน
- ตัวอย่างเช่น
  - ▶ แทนที่จะคำนวณ  $x^2 - y^2$  ก็เลือกใช้  $(x + y)(x - y)$  แทน
  - ▶ แทนที่จะคำนวณ  $\int_0^x \sin t dt$  ด้วย  $1 - \cos x$  ก็เลือกใช้  $2 \sin^2(x/2)$  แทน

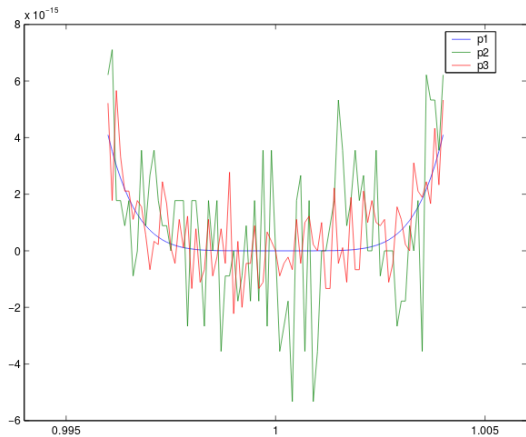
## Avoiding cancellation

หรืออีกตัวอย่าง

$$\begin{aligned} p(x) &= (x - 1)^6 \\ &= x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1 \\ &= 1 + x(-6 + x(15 + x(-20 + x(15 + x(-6 + x)))))) \end{aligned}$$

These 3 forms are equivalent at the theoretical level, but they yield very different results when they are implemented.

# Avoiding cancellation



แล้วเราจะทราบได้อย่างไรว่าควรจะใช้ expression ไหน



## Condition of mathematical operations

เราสามารถมองการกระทำทางคณิตศาสตร์เป็นฟังก์ชันได้ โดยฟังก์ชันนี้อยู่ในรูปของ  $f: X \rightarrow Y$  ในที่นี้  $X$  แทนปริภูมิของข้อมูลขาเข้า และ  $Y$  แทนปริภูมิของคำตอบ

- เราสนใจพฤติกรรมของฟังก์ชัน  $f$  เมื่อเราใส่ข้อมูลขาเข้า  $x$  ให้ไป
- เราจะเรียกฟังก์ชันนั้นว่าเป็น well-conditioned function ถ้าการเปลี่ยนแปลงใน  $x$  ส่งผลต่อการเปลี่ยนแปลงของผลลัพธ์  $f(x)$  น้อย
- ฟังก์ชันนั้นว่าเป็น ill-conditioned function ถ้าการเปลี่ยนแปลงใน  $x$  ส่งผลต่อการเปลี่ยนแปลงของผลลัพธ์  $f(x)$  มาก

# Condition number of a scalar operation

- Operation ลักษณะนี้สามารถมองเป็นฟังก์ชัน  $f: \mathbb{R} \rightarrow \mathbb{R}$ 
  - ▶ ตัวอย่างของ operation ลักษณะนี้ได้แก่ การยกกำลัง:  $f(x) = x^2$
- หากฟังก์ชันดังกล่าวสามารถหาอนุพันธ์ได้ **absolute condition number** ( $\kappa$ ) จะมีค่าเท่ากับ

$$\kappa = \lim_{\Delta x \rightarrow 0} \frac{\|f(x + \Delta x) - f(x)\|}{\|\Delta x\|} = f'(x)$$

# Condition number of a vector-valued operation

- Operation ลักษณะนี้สามารถมองเป็นฟังก์ชัน  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ 
  - ▶ ตัวอย่างของ operation ลักษณะนี้ได้แก่ การบวก:  $f([x_1, x_2]) = x_1 + x_2$
- หากฟังก์ชันดังกล่าวสามารถหาอนุพันธ์ลำดับที่สองได้ Absolute condition number จะถูกนิยามเป็นขนาดของ Jacobian matrix ของ  $f$
- Jacobian matrix คือเมทริกซ์ของอนุพันธ์ย่อย (partial derivative) ที่ตำแหน่ง  $i, j$  มีค่าเท่ากับ  $\frac{\partial f_i}{\partial x_j}$ 
  - ▶ ในกรณีที่  $f$  ให้ผลลัพธ์เป็น vector  $f_i$  หมายถึง component ที่  $i$  ของ vector ผลลัพธ์

## Condition number of a vector-valued operation

- ขนาด (norm) ของเมทริกซ์ที่ใช้กันมากคือ  $\|\cdot\|_\infty := \max_i |x_i|$  (สมาชิกที่ค่าสัมบูรณ์มีค่ามากที่สุด)
- ถ้า  $\|J\| := \kappa \gg 1$ ,  $f$  จะถือว่าเป็น ill-conditioned operation

## ตัวอย่างการหา Jacobian

Consider a continuous twice differentiable function  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$f(x_1, x_2) = \begin{bmatrix} x_1^2 x_2 \\ 5x_1 + \sin x_2 \end{bmatrix}$$

we have  $f_1(x_1, x_2) = x_1^2 x_2$  and  $f_2(x_1, x_2) = 5x_1 + \sin x_2$  and the Jacobian of  $f$  is

$$J_f(x_1, x_2) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 x_2 & x_1^2 \\ 5 & \cos x_2 \end{bmatrix}$$

## Relative Condition number

นอกจาก absolute condition number เราสามารถนิยาม **relative condition number** ที่นำค่าของข้อมูลเข้า และผลลัพธ์มา normalise ดังนี้

$$\begin{aligned}\kappa &= \lim_{\Delta x \rightarrow 0} \frac{\|f(x + \Delta x) - f(x)\| / \|f(x)\|}{\|\Delta x\| / \|x\|} \\ &= \frac{\|J\|}{\|f(x)\| / \|x\|}\end{aligned}$$

## ตัวอย่าง

ลองคำนวณฟังก์ชันต่อไปนี้ด้วยมือดู

$$f(x) = \sin(10^8 x) \quad (2)$$

ที่  $x = \pi$  เราจะได้ว่า  $f(\pi) = 0$

แต่ถ้าใช้คอมพิวเตอร์

```
julia>sin(10^8 pi)
-3.9082928156687315e - 8
```

เพราะ absolute condition number  $c = 10^8 \cos(10^8 \pi) = 10^8$

## ตัวอย่าง

- พิจารณาฟังก์ชัน  $f(x) = x/2$ , พบว่า Jacobian มีค่าเท่ากับ  $J = f' = 1/2$  ดังนั้น relative condition number คือ

$$\kappa = \frac{\|J\|}{\|f(x)\|/\|x\|} = \frac{1/2}{(x/2)/x} = 1 \quad (3)$$

ดังนั้นแล้ว problem ดังกล่าวถือว่าเป็น well-conditioned problem.

- พิจารณาฟังก์ชัน  $f(x) = \sqrt{x}$ , พบว่า Jacobian มีค่าเท่ากับ  $J = f' = 1/(2\sqrt{x})$  ดังนั้น relative condition number คือ

$$\kappa = \frac{\|J\|}{\|f(x)\|/\|x\|} = \frac{1/(2\sqrt{x})}{\sqrt{x}/x} = \frac{1}{2} \quad (4)$$

ดังนั้นแล้ว problem ดังกล่าวถือว่าเป็น well-conditioned problem เช่นกัน.



## ตัวอย่าง

- พิจารณาฟังก์ชัน  $f(\vec{x}) = x_1 - x_2$ , พบว่า Jacobian มีค่าเท่ากับ

$$J = \left[ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right] = [1 \quad -1]$$

พบว่า  $\|J\|_\infty = 1$  ดังนั้น relative condition number

$$\kappa = \frac{\|J\|_\infty}{\|f(\vec{x})\|_\infty / \|\vec{x}\|_\infty} = \frac{1}{|x_1 - x_2| / \max\{|x_1|, |x_2|\}} \quad (5)$$

ดังนั้นแล้ว problem ดังกล่าวจะ ill-conditioned เมื่อ  $|x_1 - x_2| \approx 0$  หรือ  $x_1 \approx x_2$  ซึ่งตรงกับสิ่งที่เราพบในการทดลองเรื่อง cancellation error

# Summary

- คอมพิวเตอร์ไม่สามารถแสดงค่าจำนวนจริงได้หมด แสดงได้แต่เซตย่อยของจำนวนจริงเท่านั้น
- การดำเนินการทางคณิตศาสตร์อาจได้รับผลกระทบจาก numerical error ได้
- เราสนใจที่จะหาขั้นตอนวิธีในการคำนวณทางคณิตศาสตร์ที่ดี ที่สามารถลด numerical error ได้บ้าง
- เราได้เรียนรู้ไปว่าเราสามารถประเมินลักษณะของปัญหาการคำนวณว่าจะเกิด error มากน้อยแค่ไหนโดยอาศัยตัวเลขที่เรียกว่า condition number

# Assignments

- Show that the largest possible number in 64-bit IEEE floating point is

$$2^{1023}(2 - 2^{-52})$$

- Write a program to convert IEEE bit string to real number. (LAB)

- Numerical Methods for Computational Science and Engineering: Prof. Ralf Hiptmair: <https://people.math.ethz.ch/~grsam/HS16/NumCSE/NumCSE16.pdf>