Lecture 7

Floating point arithmetic and stability

2.5 Machine representation of numbers

- Scientific notation:

$$\underbrace{-}_{\substack{\text{sign}\\ s}} \quad \underbrace{3.14159265}_{\substack{\text{mantissa}\\ \text{(significand)}\\ m}} \times \underbrace{10}_{\substack{\text{base}\\ \beta}} \quad \overbrace{-23}^{\substack{\text{exponent}\\ e}}$$

- A floating point number system is a (finite) subset $\mathbb{F}$ of $\mathbb{R}$:

$$(-1)^s \cdot m \cdot \beta^{e-t}$$

  A more common way:

$$x = (-1)^s \cdot (0.a_1 a_2 \cdots a_t) \cdot \beta^e = (-1)^s (\frac{a_1}{\beta} + \frac{a_2}{\beta^2} + \cdots + \frac{a_t}{\beta^t})$$

- Exponent range $[e_{\text{min}}, e_{\text{max}}]$

- Normalized if $a_1 \neq 0$. Use $e = e_{\text{min}} - 1$ to represent 0.

- Floating points are not evenly spaced. For example $\beta = 2$, $t = 3$, $e_{\text{min}} = -1$ and $e_{\text{max}} = 3$. The 21 nonnegative floating point numbers are

$$0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875,$$
$$1.0, 1.25, 1.50, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0$$

  Spacing of the floating point numbers jumps by a factor 2 at each power of 2.

Machine precision

- Machine epsilon: ϵ_M

  ϵ_M = distance between 1 and the next larger floating point number

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

  It is the spacing of the floating point numbers between 1.0 and β.

  The spacing between 1.0 and $1/\beta$ is $\beta^{-t} = O(\epsilon_M)/\beta$.

- roundoff unit $u = \frac{1}{2}\epsilon_M$, the most useful quantity in error analysis.

Rounding

In converting $x \in \mathbb{R}$ to $\mathrm{fl}\,(x) \in \mathbb{F}$.

- Rounding: $\mathrm{fl}\,(x)$ rounds up if $(t+1)$st digit is $\geq \beta/2$ and down otherwise.

- error due to the finite representation in floating point form of real (and complex) numbers in digital computers.

- The main causes of error are round-off error and data error.

Property 2.1 If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then

$$\mathrm{fl}\,(x) = x(1+\epsilon), \qquad |\epsilon| \leq u$$

IEEE Standards (rounding):

- single precision, $\beta = 2$, $t = 24$, $\epsilon_M \approx 10^{-8}$, $e_{\text{min}} = -125$, $e_{\text{max}} = 128$
- double precision, $\beta = 2$, $t = 53$, $\epsilon_M \approx 10^{-16}$, $e_{\text{min}} = -1021$, $e_{\text{max}} = 1024$
- MATLAB's `eps` gives $\epsilon_M \approx 2.220446049250313 \times 10^{-16}$ with `format long`. `single(x)` to convert `x` to single precision.
- Single precision:

0	00000000	00000000000000000000000
s	e	m

- Overflow and underflow
- IEEE floating point data types

	Single precision	Double precision
t	24 bits	53 bits
e	8 bits	11 bits
s	1 bit	1 bit
e_{max}	127	1023
e_{min}	-126	-1022
Smallest normalized	$2^{-126} \approx 10^{-38}$	$2^{-1022} \approx 10^{-308}$
Largest normalized	$e^{127} \approx 10^{38}$	$2^{1023} \approx^{308}$
ϵ_M	$2^{-24} \approx 6 \times 2^{-8}$	$2^{-53} \approx 10^{-16}$

- Read p.47-55

Special Quantities

- ∞ is returned when an operation overflows
- $x/\pm\infty = 0$ for any number x, $x/0 = \pm\infty$ for any nonzero number x
- Operations with infinity are defined as limits, e.g.

$$4 - \infty = \lim_{x \to \infty} 4 - x = -\infty$$

- NaN (Not a Number) is returned when the an operation has no well-defined finite or infinite result.
- Examples: $\infty - \infty, \infty/\infty, 0/0, \sqrt{-1}, NaN \odot 4$.

Two fundamental issues: stability and conditioning

1. Stability pertains to the perturbation behavior of an algorithm.

2. Conditioning pertains to the perturbation behavior of a mathematical problem.

Algorithms

- A mathematical problem may be defined as a function $f : X \to Y$ from a vector space X of data to a vector space Y of solutions.

- An algorithm $\tilde{f}$ for a problem f can be viewed as another map $\tilde{f} : X \to Y$.

- The problem of solving $Ax = b$: input data A and b and the solution is x. The computed solution is denoted by $\tilde{x}$

Accuracy

- An algorithm $\tilde{f}$ for a problem f is *accurate* if (for each x)

$$\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(x)\|} = O(\epsilon_M)$$

- If the problem f is ill-conditioned, the goal of accuracy is unreasonably ambitious. Rounding of the input data is unavoidable on a computer. The perturbation alone might lead to a significant change in the result.

Stability

- An algorithm $\tilde{f}$ for a problem f is *stable* if (for all x)

$$\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = O(\epsilon_M)$$

  for some $\tilde{x}$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_M)$$

- "Nearly the right answer to nearly the right question"

- The definitions of stability given here are useful in many parts of numerical linear algebra, the condition $O(\epsilon_M)$ is probably too strict to be appropriate for all numerical problems in other areas such as differential equations.

Backward stable

- An algorithm $\tilde{f}$ for a problem f is *backward stable* if (for all x)

$$\tilde{f}(x) = f(\tilde{x}) \text{ for some } \tilde{x} \text{ with } \frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_M)$$

- "exactly the right answer to nearly the right question"

- Stronger and simpler than stability: a tightening of the definition of stability.

Conditioning and condition number

- Well condition problem means small perturbations of x lead to only small changes in $f(x)$. An ill-conditioned problem means some small perturbation of x leads to a large change in $f(x)$.

- The relative condition number is defined by

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

- Example 2.3 (condition of a system of equations)

  Let $b \in \mathbb{R}^n$ be fixed. The condition number of computing the solution $Ax = b$ where A is nonsingular is

$$K(A) = \|A\|\|A^{-1}\|.$$

Conditioning vs. Stability

- Condition = sensitivity of the solution to perturbations in the data

- Backward stable algorithm produces accurate results on well conditioned problems.

- Backward stable algorithm could produce inaccurate answers for sensitive (ill-conditioned) problems, but the inaccuracy is proportional to the sensitivity.

- We concentrate on designing backward stable algorithms!

Accuracy of a backward stable algorithm

- If a backward stable algorithm is used to solve a problem f with condition number K, then

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O(K(x)\epsilon_M)$$

- It is accurate if the problem is well conditioned.

- Rule of thumb:

$$\text{forward error} \precsim \text{condition number x backward error}$$

Stability of floating point arithmetic

- $x \mapsto \text{fl}\,(x)$ is backward stable. For each $x \in \mathbb{R}$, there is $\epsilon \leq \epsilon_M$ such that

$$\text{fl}\,(x) = x(1+\epsilon)$$

- For all floating points x, y, $(x, y) \mapsto x * y$ is backward stable. There exists $\epsilon \leq \epsilon_M$ such that

$$\text{fl}\,(x * y) = (x * y)(1+\epsilon)$$

- For all floating points x, y, $(x, y) \mapsto \text{fl}\,(x+y) = \text{fl}\,(x) + \text{fl}\,(y)$ is backward stable

- Inner product $(x, y) \mapsto \text{fl}\,(x^T y)$ is backward stable, $x, y \in \mathbb{R}^n$.

- Outer product $(x, y) \mapsto \text{fl}\,(xy^T)$ is not backward stable, $x, y \in \mathbb{R}^n$.

- $x \mapsto \text{fl}\,(x+1)$ is not backward stable (consider $x \approx 0$)