# Course 1: Introduction to Some Computational Concepts

Yasuyuki Sawada, Yaolang Zhong

University of Tokyo

October 1, 2025

# The Economics of Computation

- Idea: Economics studies the allocation of scarce resources; computation allocates time and memory to reach a target accuracy.
- Research trade-offs:
  - accuracy vs. cost/time
  - generality vs. problem-specific efficiency
- Remarks:
  - total time = human coding time + machine runtime
  - hardware and algorithms jointly determine runtime
  - tolerances formalize what counts as "accurate enough": what error tolerance is defensible for this task?
- Two sources of errors:
  - Round-off error: from finite-precision arithmetic and representation.
  - Truncation error: from approximating a mathematical object by a finite or discrete

# Round-off error: essentials

- Floating point in one line: numbers are stored with a sign, a few significant digits, and a scale (like $\times 10^k$ or $\times 2^k$). Only finitely many reals are representable and spacing is uneven.
- Machine epsilon $\varepsilon_{\text{mach}}$: smallest $\varepsilon$ such that the machine recognizes $1+\varepsilon>1$ and $1-\varepsilon<1$; a local measure of rounding granularity near 1.
- Machine infinity and machine zero: the largest and smallest magnitudes still representable (in absolute value). Overflow occurs when a result exceeds machine infinity; underflow occurs when a nonzero result is rounded to machine zero.
- Example 1: 32 and 64 bit in laptop
  - 32-bit single: about 7 decimal digits; $\varepsilon_{\text{mach}} \approx 1.2 \times 10^{-7}$; representable magnitudes roughly $10^{-38}$ to $10^{38}$.
  - 64-bit double: about 16 decimal digits; $\varepsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$; magnitudes roughly $10^{-308}$ to $10^{308}$.

# Round-off: cancellation and remedies

- Error propagation: small initial errors can be magnified by arithmetic, especially subtraction of nearly equal numbers (cancellation).
- Example 2: solve $x^2 - 26x + 1 = 0$ for the small root.
  - True small root: $x^* = 13 - \sqrt{168} \approx 0.0385186\ldots$
  - Five-digit arithmetic (naive subtraction):

$$x^* \doteq 13.000 - 12.961 = 0.039 \equiv \hat{x}_1$$

  - Algebraically stable form (avoids subtracting close numbers):

$$13 - \sqrt{168} = \frac{1}{13 + \sqrt{168}} \doteq \frac{1}{25.961} \doteq 0.038519 \equiv \hat{x}_2$$

# Round-off: cancellation and remedies

- Practical habits to limit round-off:
  - rescale variables to typical magnitudes
  - avoid nearly singular transformations; prefer stable primitives
  - Example 3:
    - `hypot(a,b)`: computes $\sqrt{a^2+b^2}$ using scaling to avoid overflow/underflow and loss of significance when $|a|$ and $|b|$ differ greatly. Example: `hypot(3e200,4e200)` is finite, while $\sqrt{(3\cdot 10^{200})^2+(4\cdot 10^{200})^2}$ overflows.
    - `log1p(x)`: computes $\log(1+x)$ accurately for small $|x|$, avoiding cancellation when $1+x\approx 1$. Near zero it matches the series $x-\frac{x^2}{2}+\frac{x^3}{3}-\cdots$.
    - `expm1(x)`: computes $e^x-1$ accurately for small $|x|$, avoiding cancellation in $\exp(x)-1$. Near zero it matches $x+\frac{x^2}{2}+\frac{x^3}{6}+\cdots$ and preserves tiny negative results.
  - prefer additions of similar-scale terms; defer subtractions of nearly equal numbers via algebraic reformulation

# Truncation error

- Truncation arises when replacing a continuous or infinite object by a finite one: finite differences for derivatives, finite time steps, or truncated series.
- Example 4: the exponential function

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

must be approximated by a finite sum

$$\sum_{n=0}^{N} \frac{x^n}{n!}$$

with a remainder that shrinks as $N$ grows (up to round-off limits).

- Choosing $N$ or a step size is a trade-off: more terms or smaller steps reduce truncation error but may increase round-off and cost.

# Conditioning

Conditioning (of the problem): how much the true solution can change under small input changes.

- Setup and notation:
  - True problem: $Ax = b$. Perturbed problem: $(A + \Delta A)(x + \Delta x) = b + \Delta b$.
  - Here $\Delta A, \Delta b$ are small data perturbations; $\Delta x$ is the induced change in the solution.
  - A standard first-order bound (any consistent norm):

$$\frac{\|\Delta x\|}{\|x\|} \lesssim \kappa(A)\left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|}\right), \quad \text{where } \kappa(A) = \|A\|\,\|A^{-1}\|.$$

# Conditioning

## Example 5:

- Well-conditioned matrix (use the 1-norm for concreteness):
  - $A = I_2$: $\|A\|_1 = 1$, $\|A^{-1}\|_1 = 1 \Rightarrow \kappa_1(A) = 1$.
  - $A = \operatorname{diag}(1, 2)$: $\|A\|_1 = 2$, $A^{-1} = \operatorname{diag}(1, 1/2)$ so $\|A^{-1}\|_1 = 1 \Rightarrow \kappa_1(A) = 2$.
- Ill-conditioned matrix (nearly singular $2 \times 2$):
  - $A_\varepsilon = \begin{bmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{bmatrix}$, with $\varepsilon > 0$ small. $\det(A_\varepsilon) = \varepsilon$.
  - $A_\varepsilon^{-1} = \frac{1}{\varepsilon} \begin{bmatrix} 1+\varepsilon & -1 \\ -1 & 1 \end{bmatrix}$.
  - $\|A_\varepsilon\|_1 = \max\{2,\ 2+\varepsilon\} = 2+\varepsilon$.
  - $\|A_\varepsilon^{-1}\|_1 = \max\left\{\frac{2+\varepsilon}{\varepsilon},\ \frac{2}{\varepsilon}\right\} = \frac{2+\varepsilon}{\varepsilon}$.
  - Hence $\kappa_1(A_\varepsilon) = (2+\varepsilon) \cdot \frac{2+\varepsilon}{\varepsilon} \approx \frac{4}{\varepsilon}$ (blows up as $\varepsilon \to 0$). $\varepsilon = 10^{-6} \Rightarrow \kappa_1 \approx 4 \times 10^6$: tiny data errors can cause order-one relative errors in $x$.

# Stability: meaning and link to conditioning

Stability (of the algorithm): how much the computation amplifies rounding noise for a given problem.

- Setup:
  - True problem $Ax = b$ with exact solution $x$; the algorithm returns $\hat{x}$.
  - Forward error: how far the answer is from truth, $\|\hat{x} - x\|/\|x\|$.
  - Backward error: how much inputs must be nudged so $\hat{x}$ becomes exactly right, i.e., smallest $\Delta A, \Delta b$ with $(A + \Delta A)\hat{x} = b + \Delta b$.
- Backward-stable algorithm:
  - It promises tiny data nudges: $\|\Delta A\|/\|A\|,\ \|\Delta b\|/\|b\| = O(\varepsilon_{\text{mach}})$.
  - Interpretation: the computed $\hat{x}$ is the exact solution of a nearby problem.
- Link to conditioning:

$$\frac{\|\hat{x} - x\|}{\|x\|} \lesssim \kappa(A)\left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\right), \quad \kappa(A) = \|A\|\,\|A^{-1}\|.$$

# Stability

- Remarks:
  - stability is about the algorithm; conditioning is about the problem
  - even a stable algorithm can yield large forward error on an ill-conditioned problem
- Example 6: quadratic roots
  - Direct formula $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ is unstable for the small root when $b^2 \gg 4ac$ (cancellation).
  - Stable rearrangement: set $q = -\frac{1}{2}\big(b + \operatorname{sign}(b)\sqrt{b^2 - 4ac}\big)$, then $x_1 = q/a$ and $x_2 = c/q$. This avoids subtracting nearly equal numbers and behaves like a backward-stable computation.

# Rates of convergence

Let $e_k$ be the error at iteration $k$. If $e_{k+1} \approx C\, e_k^p$:

- linear: $p = 1$, $0 < C < 1$ (steady geometric decay)
- superlinear: $p > 1$ or $C \to 0$ (faster than linear)
- quadratic: $p = 2$ (e.g., Newton near the solution)

Example 7: Newton on $f(x) = x^2 - 2$: $x_{k+1} = \frac{1}{2}\left(x_k + 2/x_k\right)$. Near $\alpha = \sqrt{2}$, the error satisfies $e_{k+1} \approx \frac{f''(\alpha)}{2f'(\alpha)} e_k^2 = \frac{1}{2\sqrt{2}} e_k^2$ (quadratic). Starting $x_0 = 1$:

$$e_0 \approx -0.4142, \quad e_1 \approx 0.0858, \quad e_2 \approx 2.45 \times 10^{-3}, \quad e_3 \approx 2.1 \times 10^{-6}.$$

Note $e_2 \approx \frac{1}{2\sqrt{2}}\, e_1^2$, illustrating quadratic decay.

# Big-O for sequences

Big-O and little-o (for sequences $a_k, b_k > 0$):

$$a_k = O(b_k) \iff \exists\, C > 0,\ k_0 :\ |a_k| \leqslant C\, b_k \text{ for all } k \geqslant k_0, \qquad a_k = o(b_k) \iff \frac{|a_k|}{b_k} \to 0.$$

Example 8: Polynomial growth: $a_k = 3k^2 + 5k$, $b_k = k^2$. Then $a_k/b_k = 3 + 5/k \leqslant 8$ for all $k \geqslant 1$. Hence $a_k = O(b_k)$. Moreover, $5k = o(k^2)$ since $(5k)/k^2 \to 0$.

Connection to convergence rates: if $e_{k+1} \leqslant r\, e_k$ with $r \in (0,1)$, then $e_k \leqslant r^k e_0$, so $e_k = O(r^k)$ (linear/geometric convergence).

## Efficient Evaluation of Expressions

Example 8: evaluate a polynomial $P_n(x) = \sum_{k=0}^{n} a_k x^k$ with minimal work. Methods:

- Direct (naive): compute each $x^k$ via exponentiation and sum the terms.
- Alternative: expand $x^k$ as $x \cdot x \cdots x$ each time (no exponentiation, but many repeated multiplications).
- Better method (reuse powers):
  - build powers once: set $t = 1$; for $k = 1, \ldots, n$ do $t \leftarrow t \cdot x$ so $t = x^k$
  - accumulate: $S \leftarrow a_0$; for $k = 1, \ldots, n$ do $S \leftarrow S + a_k\, t$
  - intuition: replaces the $n-1$ exponentiations by $n-1$ multiplications to form $x^k$; avoids recomputing the same power for different terms
- Horner's method (nested form):

$$a_0 + a_1 x + \cdots + a_n x^n \;=\; a_0 + x\big(a_1 + x(\cdots + x(a_{n-1} + x a_n)\cdots)\big)$$

  - algorithm (top–down): $y \leftarrow a_n$; for $k = n-1, \ldots, 0$: $y \leftarrow a_k + x\, y$

## Operation Counts for $P_n(x) = \sum_{k=0}^{n} a_k x^k$

| Method | additions | multiplications | exponentiations |
|---|---|---|---|
| Direct (naive) | $n$ | $n$ | $n-1$ |
| Alternative (expand each $x^k$) | $n$ | $n + \frac{(n-1)n}{2}$ | $0$ |
| Better method (reuse powers) | $n$ | $2n-1$ | $0$ |
| Horner's method | $n$ | $n$ | $0$ |

Notes:

- Better method: $n-1$ multiplies to build $x^k$ for $k = 1..n$ plus $n$ multiplies for $a_k x^k$ (no multiply for $a_0$) gives $2n-1$.
- Horner: exactly one multiply and one add per coefficient, so $n$ multiplies and $n$ adds; no powers are formed.
- Lesson: algebraically equivalent rewrites can change runtime by large constants, even when all methods are $O(n)$.

# Finite-Difference Derivatives

For smooth $f$ and small $h$:

$$\begin{aligned} \text{Forward: } f'(x) &\approx \frac{f(x+h)-f(x)}{h} && \text{error } O(h) + O(\varepsilon_{\mathsf{mach}}/h), \\ \text{Central: } f'(x) &\approx \frac{f(x+h)-f(x-h)}{2h} && \text{error } O(h^2) + O(\varepsilon_{\mathsf{mach}}/h). \end{aligned}$$

- **Trade-off**: truncation ($\downarrow$ with $h$) vs. round-off ($\uparrow$ with $1/h$).
- Rule-of-thumb: choose $h$ to balance both (often $h \sim \sqrt{\varepsilon_{\mathsf{mach}}}$ in scaled units for central difference).

# Direct vs. iterative methods

- Direct methods (e.g., Gaussian elimination)
  - pros: predictable error behavior; one-shot solve; good when the matrix is dense and fits in memory
  - cons: fill-in and memory growth for sparse problems; factorization cost can dominate for very large systems
- Iterative methods (fixed-point, Newton, Krylov, etc.)
  - pros: exploit sparsity/structure; matrix–vector products only; early stopping via tolerances
  - cons: need decent preconditioning/initialization; convergence diagnostics and stopping rules matter
- Stopping rules
  - residual norms $\|b - Ax_k\|$, step norms $\|x_k - x_{k-1}\|$, and problem-scale-aware tolerances

# Direct vs. iterative methods

Example 9: Solve $Ax = b$ with $A = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

Direct (solve once):

$$A^{-1} = \tfrac{1}{11}\begin{bmatrix} 3 & -1 \\ -1 & 4 \end{bmatrix} \quad \Rightarrow \quad x^\star = A^{-1}b = \tfrac{1}{11}\begin{bmatrix} 1 \\ 7 \end{bmatrix} = \begin{bmatrix} 0.0909 \\ 0.6364 \end{bmatrix}.$$

# Direct vs. iterative methods

Iterative (Jacobi; start $x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$):

$$x^{(k+1)} = \begin{bmatrix} \frac{1-x_2^{(k)}}{4} \\ \frac{2-x_1^{(k)}}{3} \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 0.25 \\ 0.6667 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 0.0833 \\ 0.5833 \end{bmatrix},$$

$$x^{(3)} = \begin{bmatrix} 0.1042 \\ 0.6389 \end{bmatrix}, \quad x^{(5)} = \begin{bmatrix} 0.0920 \\ 0.6366 \end{bmatrix} \approx x^\star.$$

- direct: one factorization/solve gives $x^\star$ exactly (up to round-off)
- iterative: successive refinements approach $x^\star$; cost per step is a few multiplies/adds and one matrix–vector pattern (good for sparse $A$)
- stopping: halt when $\|b - Ax^{(k)}\|$ or $\|x^{(k)} - x^{(k-1)}\|$ drops below a tolerance tied to data scale

# References & Further Reading


- K. L. Judd, Numerical Methods in Economics, Chapter 2 "Elementary Concepts."
- "Notes for Chapter 2: Elementary Concepts" (slides/notes).