CS 450: Numerical Analysis – Lecture 1

Course Overview, Syllabus & Technical Details, Error Analysis

Edgar Solomonik

August 25, 2026

Today

- Course overview
- Syllabus & technical details
- Error analysis

What is a "numerical" problem?

Prototypical form. A problem is a function

$$g : \mathbb{C}^m \times \mathbb{C}^n \longrightarrow \{0, 1\}.$$

For input $z \in \mathbb{C}^m$, an output $x \in \mathbb{C}^n$ is valid if and only if

$$g(z, x) = 1.$$

The problem is "well-posed" if, for every $z \in \mathbb{C}^m$, there exists a unique $x \in \mathbb{C}^n$ such that $g(z, x) = 1$, and the resulting solution map $f : \mathbb{C}^m \to \mathbb{C}^n$, defined by $g(z, f(z)) = 1$ for every $z \in \mathbb{C}^m$, is continuous. Otherwise, the problem is "ill-posed." Instead of $\mathbb{C}^m$ or $\mathbb{C}^n$, most of the course will focus on $\mathbb{R}^m$ and $\mathbb{R}^n$, but we may also consider functions instead of vectors.

Course structure

Error, conditioning, and floating point.

- How do we represent $a \in \mathbb{R}$ on a computer?
- How do we analyze the propagation of error in algorithms and problems?

$$x = f(z), \qquad \widehat{x} = f(z + \delta z).$$

Linear systems of equations. Given $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that

$$Ax = b.$$

For fixed A,

$$g(b, x) = \begin{cases} 1, & Ax = b, \\ 0, & \text{otherwise}, \end{cases} \qquad f(b) = A^{-1}b.$$

The problem is well-posed if A is invertible.

Linear least-squares problems. For a fixed $A \in \mathbb{R}^{m \times n}$ and given $b \in \mathbb{R}^m$, solve

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2.$$

Eigenvalue problems. Given $A \in \mathbb{C}^{n \times n}$, find pairs (x, λ) with $x \neq 0$ such that

$$Ax = \lambda x.$$

Nonlinear systems of equations. Given a function $f : \mathbb{R}^m \to \mathbb{R}^n$, solve

$$f(x) = 0.$$

Optimization. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, solve

$$\min_{x \in \mathbb{R}^n} f(x).$$

Interpolation. Given

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R} \times \mathbb{R},$$

find $h : \mathbb{R} \to \mathbb{R}$ such that

$$h(x_i) = y_i \qquad \text{for every } i.$$

This is not well-posed in this form; we need to restrict h further.

Quadrature & differentiation. Given $f : \mathbb{R} \to \mathbb{R}$ and $a < b$, find f' or

$$\int_a^b f(x)\, dx.$$

The course focus is quadrature.

Ordinary differential equations: initial-value problem. Given a differential equation $D(f) = 0$ and initial conditions $f(0), \ldots$, find $f(t)$ for $t \geq 0$ such that

$$D(f) = 0.$$

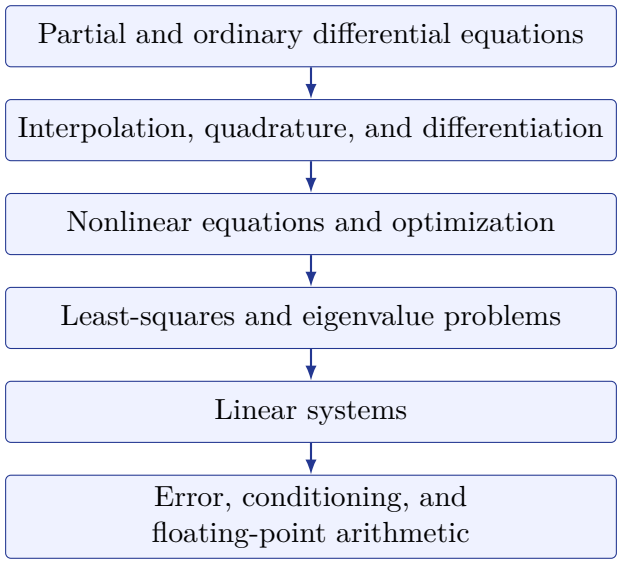
ODE boundary-value problem. Given D, and f, f', or f'' at a and b, find f on $[a, b]$.

Partial differential equations. Let $f : \mathbb{R}^m \to \mathbb{R}$ for $m = 2$ or $m = 3$. Given boundary data $f(x)$ for $x \in \partial\Omega$, find $f(x)$ for $x \in \Omega$ such that

$$(Df)(x) = 0 \qquad \text{for every } x \in \Omega.$$

Applications & connections

Material is cumulative.



Arrows indicate reduction to or dependence on more foundational numerical subproblems.

- PDEs are a standard tool for modeling the physical world.
- Optimization is a broad area; the focus here is on smooth, nonlinear functions. It is also relevant to ML when such functions are part of a model.
- Linear least squares and EVP: elementary model fitting, also basic ML.

- Linear systems: a basic building block for everything numerical. They appear in the mathematical formulation of many problems, including other domains, e.g., shortest-path problems, automatic differentiation, quantum computing, and network analysis.

Syllabus

Course webpage. relate.cs.illinois.edu/course/cs450-f26/

Relate: a web-based course platform for programming, assignments, and assessment. You can code in Python on the website; a standalone client is not needed.

Textbook. *Scientific Computing: An Introductory Survey*, Michael T. Heath.

See also the slides via the webpage; they are free for students. The course and lectures will follow the same structure and content, with minor deviations.

Homework, quizzes, grading policy.

Grade %	Component
20	About one homework every two weeks, plus an addendum problem for the 4-credit-hour section. Typically three problems: one theory, one implementation, and one application.
10	About one quiz per lecture: about ten questions in multiple-choice or numerical-entry format; multiple attempts are allowed.
70	About four exams plus a final. Questions use the same style as the quizzes; midterms have about 25 questions and the final exam about 40. Retakes are allowed, with score $\frac{2}{3}\max\{\text{retake}, \text{original}\} + \frac{1}{3}\,\text{original}.$

Refer to the grading policy on the course webpage for more details.

CBTF/PrairieTest exam registration. https://www.prairietest.org/

CBTF reservation instructions: https://cbtf.illinois.edu/students/res

Resources. Office hours and Piazza; see the webpage.

Integrity policy. You can work together and share ideas, but write proofs and code independently. No generative AI on homework or exams; see the course policy for other permitted uses.

Error & conditioning

Suppose we have a well-posed problem $f : \mathbb{R} \to \mathbb{R}$. Throughout this discussion, z denotes the input and x the exact output, with

$$x = f(z).$$

The "accuracy" of a candidate solution $\hat{x}$ is measured in terms of the signed forward error

$$\hat{x} - x.$$

$$\begin{aligned} \text{absolute error:} \quad & |\hat{x} - x|\,, \\ \text{relative error:} \quad & \frac{|\hat{x} - x|}{|x|}. \end{aligned}$$

For relative errors and condition numbers, assume that the quantities in their denominators are nonzero.

Order convention. $\hat{x} - x$ versus $x - \hat{x}$ does not matter, but the convention must be consistent. Errors of opposite sign may cancel; we often want the "magnitude" of error in some final quantity.

Sources of error. Data error is present in the input; truncation or discretization error comes from approximation; roundoff error comes from finite-precision representation and arithmetic.

For a candidate output $\widehat{x}$, define the set of compatible input perturbations

$$\mathcal{M}(z, \widehat{x}) = \{\delta z : f(z + \delta z) = \widehat{x}\}.$$

The absolute and relative backward errors are

$$\eta_{\text{abs}}(\widehat{x}; z) = \inf_{\delta z \in \mathcal{M}(z,\widehat{x})} |\delta z|, \qquad \eta_{\text{rel}}(\widehat{x}; z) = \inf_{\delta z \in \mathcal{M}(z,\widehat{x})} \frac{|\delta z|}{|z|}.$$

If $\mathcal{M}(z, \widehat{x})$ is empty, the backward error is $+\infty$. A particular $\delta z \in \mathcal{M}(z, \widehat{x})$ is a *backward-error witness*; constructing one gives an upper bound on the backward error. Equivalently, with $\widehat{z} = z + \delta z$,

$$\widehat{x} = f(\widehat{z}).$$

- "Correct answer to a perturbed problem."
- If the input is uncertain, backward error below that uncertainty indicates that the computation has not materially degraded the data; forward error still depends on conditioning.
- Often easier to bound: ask whether the computed output is the exact solution of a nearby problem.

Conditioning. Given an input perturbation δz, how much does the output differ? Let

$$\widehat{x} = f(z + \delta z) \qquad \text{and} \qquad x = f(z)$$

for some small δz. Assume that f is twice continuously differentiable near z.
The absolute condition number is

$$\kappa_{\text{abs}}(f, z) = \lim_{\delta z \to 0} \frac{|f(z + \delta z) - f(z)|}{|\delta z|} = |f'(z)|.$$

Taylor expansion gives

$$f(z + \delta z) - f(z) = f'(z)\, \delta z + O(|\delta z|^2),$$

and hence

$$|f(z + \delta z) - f(z)| = \underbrace{|f'(z)|}_{\kappa_{\text{abs}}(f,z)} |\delta z| + O(|\delta z|^2).$$

Here $R(\delta z) = O(|\delta z|^2)$ as $\delta z \to 0$ means that, for some constants $C, \eta > 0$ independent of δz,

$$|R(\delta z)| \le C\, |\delta z|^2 \qquad \text{whenever} \qquad 0 < |\delta z| < \eta.$$

Unlike $T(n) = O(n^2)$ in complexity analysis, where $n \to \infty$, this notation describes decay as $\delta z \to 0$. The hidden constant may depend on f and the fixed point z, but not on δz. Since

$$O(|\delta z|^2)/\,|\delta z| = O(|\delta z|) \longrightarrow 0,$$

the quadratic remainder is lower order than the linear term, unless $f'(z) = 0$.
The relative condition number is

$$\kappa(f, z) = \kappa_{\text{rel}}(f, z) = \lim_{\delta z \to 0} \frac{|f(z + \delta z) - f(z)|\,/\,|f(z)|}{|\delta z|\,/\,|z|} = \kappa_{\text{abs}}(f, z) \frac{|z|}{|f(z)|}.$$

Therefore,

$$\frac{|f(z + \delta z) - f(z)|}{|f(z)|} = \kappa(f, z) \frac{|\delta z|}{|z|} + O\left(\left(\frac{|\delta z|}{|z|}\right)^2\right). \tag{1.1}$$

Usually, we are interested in an upper bound on κ, since the leading term in (1.1) can be bounded using such a bound. When considering $f(z)$ for $z \in D$, define

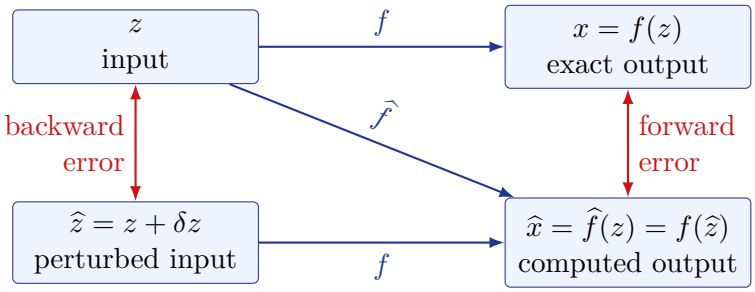
$$\kappa(f) = \sup_{z \in D} \kappa(f, z),$$

i.e., the worst case over inputs.

Problem versus algorithm. Conditioning is a property of the problem; stability is a property of an algorithm. Forward and backward error describe a computed result. A quantitative definition of backward stability is given after the floating-point arithmetic model below.

Visualization of forward and backward error

Let z be the input, $x = f(z)$ the exact output, and $\widehat{x} = \widehat{f}(z)$ the output produced by an algorithm. A backward-error witness is a nearby input $\widehat{z}$ satisfying $\widehat{x} = f(\widehat{z})$.



The condition number relates the two errors. If $\widehat{x} = \widehat{f}(z) = f(\widehat{z})$, then, for a small backward error,

$$\frac{|\widehat{x} - x|}{|x|} = \kappa(f, z) \frac{|\widehat{z} - z|}{|z|} + O\left(\left(\frac{|\widehat{z} - z|}{|z|}\right)^2\right).$$

CS 450: Numerical Analysis – Lecture 2

Floating-Point Representation, Arithmetic, and Stability

Edgar Solomonik

August 27, 2026

Today

- Brief review: forward error, backward error, and conditioning
- Floating-point representation and rounding
- Floating-point arithmetic, exceptional behavior, and backward stability

Brief review

For input z, exact output $x = f(z)$, and computed output $\widehat{x}$,

$$\text{relative forward error} = \frac{|\widehat{x} - x|}{|x|}, \qquad \text{relative backward error} = \inf_{\delta z:\, f(z+\delta z)=\widehat{x}} \frac{|\delta z|}{|z|}.$$

For a differentiable scalar problem, the local relative condition number is

$$\kappa(f, z) = \lim_{\delta z \to 0} \frac{|f(z+\delta z) - f(z)| \,/\, |f(z)|}{|\delta z| \,/\, |z|} = |f'(z)| \, \frac{|z|}{|f(z)|}.$$

Thus, to first order,

$$\text{relative forward error} \approx \kappa(f, z) \times \text{relative backward error}.$$

Floating-point representation

Scientific notation. Computers encode real numbers to fixed precision while retaining a large dynamic range. For example,

$$G = 6.6743 \times 10^{-11} \ \mathrm{m^3\,kg^{-1}\,s^{-2}}$$

uses a *significand* 6.6743 and an *exponent* -11. Binary floating point uses the same idea with base 2.

A binary (b_s, b_e) format. Let b_s be the number of stored fraction bits and b_e the number of exponent-field bits. A finite, nonzero *normal* floating-point number has the form

$$x = (-1)^\sigma (1.s_1 s_2 \cdots s_{b_s})_2 \, 2^e = (-1)^\sigma \left(1 + \sum_{i=1}^{b_s} s_i 2^{-i}\right) 2^e, \qquad s_i \in \{0, 1\}, \tag{2.1}$$

where $e_{\min} \le e \le e_{\max}$. The leading 1 is implicit, so the total significand precision is $p = b_s + 1$ bits. Including the sign bit, the format uses

$$1 + b_s + b_e$$

bits. The exponent field is stored using a bias; the all-zero and all-one exponent patterns are reserved for subnormal numbers, signed zeros, infinities, and NaNs.

The normal finite set is

$$\mathcal{F}_{\text{normal}} = \left\{ (-1)^\sigma \left(1 + \sum_{i=1}^{b_s} s_i 2^{-i}\right) 2^e : \sigma, s_i \in \{0, 1\},\ e_{\min} \le e \le e_{\max} \right\}.$$

Its positive range is

$$x_{\min}^{\text{normal}} = 2^{e_{\min}}, \qquad x_{\max} = (2 - 2^{-b_s}) 2^{e_{\max}}.$$

format	total bits	fraction bits b_s	exponent bits b_e	normal exponent range
IEEE binary32 (single)	32	23	8	$[-126, 127]$
IEEE binary64 (double)	64	52	11	$[-1022, 1023]$

Single and double precision are standard in scientific computing. Half precision and still lower precisions are common in machine learning and can also be used within mixed-precision scientific algorithms.

Spacing, machine epsilon, and unit roundoff. For normal numbers in the binade $[2^e, 2^{e+1})$, adjacent floating-point numbers are separated by

$$\operatorname{ulp}(x) = 2^{e-b_s}.$$

In particular, the gap from 1 to the next larger floating-point number is

$$\varepsilon_{\text{mach}} = 2^{-b_s}.$$

In these notes, $\varepsilon_{\text{mach}}$ denotes this gap. Under round-to-nearest arithmetic, the maximum error is one-half of a gap, so the *unit roundoff* is

$$u = \frac{\varepsilon_{\text{mach}}}{2} = 2^{-(b_s+1)} = 2^{-p}.$$

Some references use the phrase "machine epsilon" for u instead; the convention should always be checked.

Let $\mathcal{F}$ denote the set of finite representable values. Ignoring overflow and underflow for the moment, round to nearest defines

$$\operatorname{fl}(x) \in \arg\min_{y \in \mathcal{F}} |x - y|.$$

When two candidates are equally near, IEEE arithmetic uses *ties to even*: choose the candidate whose final stored significand bit is 0. If x and $\operatorname{fl}(x)$ lie in the normal range, then

$$\operatorname{fl}(x) = x(1+\delta), \qquad |\delta| \le u. \tag{2.2}$$

Thus every normal real number in range has a floating-point representative with uniformly small relative error.

Floating-point arithmetic

Rounding an arithmetic result. Suppose $b_s = 4$. Then

$$(1.0110)_2\, 2^2 + (1.0001)_2\, 2^1 = (1.11101)_2\, 2^2 \xrightarrow[\text{ties to even}]{\text{round to nearest}} (1.1110)_2\, 2^2.$$

The exact sum lies halfway between two representable numbers, and the one ending in a stored 0 is selected.

For an arithmetic operation $\circ \in \{+, -, \times, /\}$, if the exact result is nonzero and normal and no exceptional behavior occurs, the standard model is

$$\operatorname{fl}(a \circ b) = (a \circ b)(1+\delta), \qquad |\delta| \le u. \tag{2.3}$$

Equation (2.3) models the rounding of one operation on machine inputs. When real-valued data must first be represented, those input-rounding errors must also be included.

In-class exercise: floating-point addition is not associative. Assume decimal arithmetic with four significant digits, rounding after every operation. Evaluate

$$\operatorname{fl}\big(\operatorname{fl}(10^4 + 1) - 10^4\big) \qquad \text{and} \qquad \operatorname{fl}\big(10^4 + \operatorname{fl}(1 - 10^4)\big).$$

Work individually for one minute, then compare with a neighbor before continuing.

Since $10^4 + 1$ rounds to 10^4, while $1 - 10^4 = -9999$ is representable,

$$\operatorname{fl}\big(\operatorname{fl}(10^4 + 1) - 10^4\big) = 0, \qquad \operatorname{fl}\big(10^4 + \operatorname{fl}(1 - 10^4)\big) = 1.$$

Therefore, floating-point addition is not associative. Algebraically equivalent evaluation orders can produce different results.

Addition, input representation, and cancellation. Given real x, y, let

$$\widehat{x} = \mathrm{fl}(x) = x(1+\delta_x), \qquad \widehat{y} = \mathrm{fl}(y) = y(1+\delta_y),$$

and round their sum once more:

$$\widehat{s} = \mathrm{fl}(\widehat{x}+\widehat{y}) = \big((1+\delta_x)x + (1+\delta_y)y\big)(1+\delta_s), \qquad |\delta_x|, |\delta_y|, |\delta_s| \le u.$$

Expanding the product and bounding every perturbation gives

$$\begin{aligned} |\widehat{s} - (x+y)| &\le u(|x|+|y|) + u\,|x+y| + u^2(|x|+|y|) \\ &\le (2u+u^2)(|x|+|y|). \end{aligned}$$

Hence, when $x + y \neq 0$,

$$\frac{|\widehat{s} - (x+y)|}{|x+y|} \le (2u+u^2)\frac{|x|+|y|}{|x+y|}. \tag{2.4}$$

The factor

$$\kappa_+(x,y) = \frac{|x|+|y|}{|x+y|}$$

is the relative condition number of addition under componentwise relative perturbations. It is large when $x \approx -y$.

Cancellation. A small final result can expose errors already present in the operands. The final subtraction of two machine numbers may itself be exact; the large relative error arises because the subtraction problem is ill-conditioned when the operands nearly cancel. If x and y are already machine numbers, only the last rounding in (2.3) remains, and its relative error is at most u provided the exact sum is normal.

Multiplication and division. If the real inputs are rounded before a multiplication, then

$$\widehat{p} = \mathrm{fl}\big(\mathrm{fl}(x)\,\mathrm{fl}(y)\big) = xy(1+\delta_x)(1+\delta_y)(1+\delta_p), \qquad |\delta_x|, |\delta_y|, |\delta_p| \le u.$$

Therefore,

$$\frac{|\widehat{p} - xy|}{|xy|} \le (1+u)^3 - 1 = 3u + 3u^2 + u^3.$$

This is an error bound for this particular sequence of three roundings, not a condition number. Division has an analogous first-order bound away from division by zero, overflow, and underflow. Unlike addition with cancellation, multiplication and division have uniformly modest relative sensitivity for nonzero operands.

Exceptional behavior and subnormal numbers

The relative models (2.2) and (2.3) exclude overflow, underflow, and special values. IEEE arithmetic assigns reserved bit patterns to handle these cases.

Signed zero, infinity, and NaN.

- $+0$ and -0 compare equal, but the sign may affect later operations; for example, reciprocals have opposite signed infinities.
- If a finite result exceeds the largest representable magnitude, *overflow* normally produces $+\infty$ or $-\infty$. Division of a nonzero finite number by signed zero also produces signed infinity.
- Invalid operations such as $0/0$ and $\infty - \infty$ produce *NaN* (not a number). NaNs generally propagate through subsequent arithmetic, and ordinary ordered comparisons with NaN are false.

Subnormal numbers and gradual underflow. Normal numbers use an implicit leading 1. Subnormal numbers instead have the form

$$x = (-1)^\sigma (0.s_1 s_2 \cdots s_{b_s})_2\, 2^{e_{\min}}.$$

They fill the gap between zero and the smallest positive normal number with constant spacing

$$\Delta_{\mathrm{sub}} = 2^{e_{\min} - b_s}.$$

The smallest positive subnormal number is Δ_{sub}. Under round to nearest, a subnormal result has the absolute-error bound

$$|\mathrm{fl}(x) - x| \le \frac{1}{2}\Delta_{\text{sub}},$$

but there is no uniform $O(u)$ relative-error bound near zero; the relative error may be of order one.

format	smallest positive normal	smallest positive subnormal	largest finite
binary32	2^{-126}	2^{-149}	$(2-2^{-23})2^{127}$
binary64	2^{-1022}	2^{-1074}	$(2-2^{-52})2^{1023}$

Gradual underflow has an important consequence: for distinct finite machine numbers x and y, their exact difference cannot round to zero. Thus,

$$\mathrm{fl}(x-y) = 0 \quad \Longleftrightarrow \quad x = y,$$

provided gradual underflow is enabled rather than a flush-to-zero mode. The difference may instead overflow, but it will not become zero.

Backward stability

Let f denote the exact problem and let $\widehat{f}$ denote a floating-point algorithm. Using the backward-error definition above, the algorithm is *backward stable* on an input set D if, for every $z \in D$, its computed result $\widehat{x} = \widehat{f}(z)$ satisfies

$$\eta_{\text{rel}}(\widehat{x}; z) = \inf_{\delta z:\, f(z+\delta z) = \widehat{x}} \frac{|\delta z|}{|z|} \le Cu + O(u^2), \tag{2.5}$$

where C is a modest constant, possibly depending on the problem dimension or operation count, but not on u. Equivalently, there is a nearby input $z + \delta z$ such that

$$\widehat{f}(z) = f(z+\delta z), \qquad \frac{|\delta z|}{|z|} \le Cu + O(u^2).$$

For vector or matrix data, the same definition uses a specified norm or a componentwise perturbation measure. At zero inputs, an absolute or mixed error measure is needed.

Combining backward stability with conditioning gives the first-order estimate

$$\frac{\left|\widehat{f}(z) - f(z)\right|}{|f(z)|} \lesssim \kappa(f, z)\, Cu.$$

Thus backward stability does not by itself guarantee a small forward error: the underlying problem must also be well-conditioned.

Example: one floating-point addition. For machine inputs x, y with a normal, finite exact sum,

$$\widehat{s} = \mathrm{fl}(x+y) = (x+y)(1+\delta) = x(1+\delta) + y(1+\delta), \qquad |\delta| \le u.$$

The computed value is the exact sum of inputs whose componentwise relative perturbations are at most u, so one floating-point addition is componentwise backward stable. Nevertheless, when $x \approx -y$, the condition number $\kappa_+(x, y)$ is large and the forward relative error can be large. This separates algorithmic stability from problem conditioning.

AI-use disclosure. *These notes were prepared with the assistance of Claude and ChatGPT, including transcription from handwritten lecture notes and drafting or editing of prose, examples, and derivations. The author reviewed the final document and assumes responsibility for its content.*