ACM 106a: Lecture 1

Agenda

- Introduction to numerical linear algebra
- Common problems
- First examples
- Inexact computation
- What is this course about?

Typical numerical linear algebra problems

- Systems of linear equations: solve

$$\boldsymbol{Ax = b}, \qquad \boldsymbol{A} \in \mathbf{R}^{n\times n}$$

- Overdetermined system of equations (more equations than unknowns): solve

$$\min_{\boldsymbol{x}} \|\boldsymbol{Ax - b}\|^2, \qquad \boldsymbol{A} \in \mathbf{R}^{n\times m}$$

- Eigenvalue problems: find $\boldsymbol{\lambda} \in \mathbf{R}$ and $\boldsymbol{x} \in \mathbf{R}^n$ s.t.

$$\boldsymbol{Ax} = \boldsymbol{\lambda x}, \qquad \boldsymbol{A} \in \mathbf{R}^{n\times n}$$

- Many others

Systems of linear equations are everywhere

- Many physical phenomena can be modelled as differential equations
- Many of these equations are linear
- Examples:
  - Maxwell's equations in electromagnetism
  - Heat diffusion
  - Acoustic wave propagation
- Often needs to solve these equations numerically

NLA is everywhere! Even when the differential equations are nonlinear (e.g. fluid flow)

Example

Suppose we wish to solve

$$-y'' + \sigma y' = f, \qquad 0 < x < 1.$$

- Unknown function $y(x)$ we wish to compute
- Parameter σ and right-hand side $f(x)$ are given
- Boundary conditions: $y(0) = a$, $y(1) = b$.

We wish to evaluate y numerically

Possible solution

$$-y'' + \sigma y' = f, \qquad 0 < x < 1.$$

Discretization:

- grid $x_i = i/N$, $i = 0, ..., N$ $(h = 1/N)$
- approximation

$$-y''(x_i) \sim \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

  and *centered difference* for the first derivative

$$y'(x_i) \sim \frac{y_{i+1} - y_{i-1}}{2h}$$

  Other choice: *forward difference* $y'(x_i) \sim \frac{y_{i+1} - y_i}{h}$

- Difference approximation, $f_i = f(x_i)$,

$$-\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \sigma \frac{y_{i+1} - y_{i-1}}{2h} = f_i, \quad 0 \le i < N.$$

Linear system

Assume $y(0) = y(1) = 0$. Linear system takes the form

$$-\left(1 - \frac{\sigma h}{2}\right) y_{i+1} + 2y_i - \left(1 + \frac{\sigma h}{2}\right) y_{i-1} = h^2 f_i.$$

Solve tridiagonal system: $a = 2$, $b = -(1 - \sigma h/2)$, $c = -(1 + \sigma h/2)$

$$\begin{pmatrix} a & b & & & \\ c & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & b \\ & & & c & a \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{N-1} \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \end{pmatrix}$$

Many techniques are available to solve such systems efficiently

What this course is about?

- How efficiently and accurately can we solve linear systems? (this course)
- How well does the numerical solution approximate the continuous solution? (ACM 106b, ACM 106c)

Other example: Maxwell's equations

$(\boldsymbol{E}(\boldsymbol{x},\boldsymbol{t}), \boldsymbol{H}(\boldsymbol{x},\boldsymbol{t}))$ electromagnetic field in $\mathbf{R}^3$ (variable $\boldsymbol{x}$), $\boldsymbol{t} \in \mathbf{R}$ is time)

Maxwell's equations in linear materials

$$\nabla \times \boldsymbol{E} = -\boldsymbol{\mu}\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{t}}$$

$$\nabla \times \boldsymbol{H} = \boldsymbol{J} + \boldsymbol{\epsilon}\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{t}}$$

$$\nabla \cdot \boldsymbol{\epsilon E} = \boldsymbol{\rho}$$

$$\nabla \cdot \boldsymbol{\mu H} = \mathbf{0}$$

$\boldsymbol{\epsilon}$ is the electrical permittivity and $\boldsymbol{\mu}$ is the magnetic permeability of the material; $\boldsymbol{\rho}$ is the free electric charge density, and $\boldsymbol{J}$ the free current density

Discretize $\to$ Huge linear system to solve!

Other example: data fitting

- We are given a set of observations $(x_i, y_i), i = 1, \ldots, n$
- Would like to a fit a model, e.g.

$$p(x) = b_0 + b_1 x + \ldots + b_m x^m$$

- Least squares fit: find the polynomial that is ‘closest’ to the data

$$\min_b \sum_{i=1}^{n} (y_i - p(x_i))^2.$$

- Matrix-vector notation

$$X = \begin{pmatrix} 1 & x_1 & \cdots & x_1^m \\ 1 & x_2 & \cdots & x_2^m \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^m \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

so that

$$\sum_{i=1}^{n}(y_i - p(x_i))^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|^2$$

Least Squares

Gauss (1809), Legendre (1805)

$$\min_{\boldsymbol{b}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|^2$$

Least squares fit given by solution to *normal equations*

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y}$$

Why?

Need to solve a linear systems of equations

Important subject: (much) much more later!

Another theme of this course: numerical stability

- Computer arithmetic is inexact (finite memory)
- Issues arise from inexact computations
- Interested in robust and stable algorithms

Number representation

Floating point representation

$$x = \pm .d_1 \dots d_s \dots 10^e$$

e.g.

$$10/3 = \pm 0.33 \dots 3 \dots 10^1$$

Representation in base b

$$x = .d_1 \dots d_s \dots b^e$$

Other common representation: *binary representation* where $b = 2$

10/3 in base 2?

IEEE floating point numbers with base 2

Used in almost every computer

$$x = \pm(.d_1 \ldots d_s)_2 \cdot 2^e$$

- $.d_1 \ldots d_s$ is the *mantissa* ($d_i \in \{0,1\}$, $d_1 = 1$ if $x \neq 0$)
- s is the mantissa length
- e is the exponent $e_{\min} \leq e \leq e_{\max}$

Interpretation

$$x = (d_1 2^{-1} + d_2 2^{-2} + \ldots d_s 2^{-s}) \cdot 2^e$$

- Finite set of unequispaced numbers
- Smallest positive number

$$x_{\min} = 2^{e_{\min}-1}$$

- Largest positive number

$$x_{\max} = (1 - 2^{-s}) 2^{e_{\max}}$$

IEEE floating point standard

Single precision

$$s = 24, \quad e_{\min} = -125, \; e_{\max} = 128$$

Requires 32 bits: 1 sign bit + 23 bits for mantissa + 8 bits for exponent

Double precision

$$s = 53, \quad e_{\min} = -1021, \; e_{\max} = 1024$$

Requires 64 bits: 1 sign bit + 52 bits for mantissa + 11 bits for exponent

Used in almost all modern computers

Machine precision

Definition: the *machine precision* of a binary floating point number system with mantissa length s is

$$\epsilon_M = 2^{-s}$$

Example: IEEE std. double precision

$$\epsilon_M = 2^{-53} \approx 1.1 \cdot 10^{-16}$$

Interpretation: $1 + 2\epsilon_M$ is the smallest floating point number greater than 1.

Rounding error

- $fl(x)$ is the floating point representation of x
- Numbers are rounded to the nearest floating point number; e.g.

$$fl(x) = \begin{cases} 1 & 1 \leq x < 1 + \epsilon_M \\ 1 + 2\epsilon_M & 1 + \epsilon_M \leq x \leq 1 + 2\epsilon_M \end{cases}$$

  Gives another interpretation of ϵ_M

- Rounding error and machine precision

$$\frac{|fl(x) - x|}{|x|} \leq \epsilon_M$$

  - machine precision bounds the relative error
  - number of correct decimal digits is about 16 in IEEE double precision
  - fundamental limit on accuracy of numerical computation

Floating point arithmetic

Computations reduce to elementary operations: $+$, $-$, $\times$, $\div$

Model for computation: roundoff error: x and y are floating point numbers and op is one of the four basic operations

$$x \,\tilde{\text{op}}\, y = fl(x \text{ op } y)$$

Fundamental axiom of floating point arithmetic

$$x \,\tilde{\text{op}}\, y = (x \text{ op } y)(1+\epsilon)$$

here $|\epsilon| \leq 2^{-s}$ where results are rounded (binary number system)

Relative error

$$\frac{|x \,\tilde{\text{op}}\, y - x \text{ op } y|}{|x \text{ op } y|} \leq \epsilon_M$$

Consequences

- Simple operations can be inexact
- Important to keep this in mind when designing algorithms
- Goal: robustness