Chapter
Summary Lecture

Numerical Methods for Visual Computing
WS 17/18

Michael Moeller
Visual Scene Analysis
Department of Computer Science
and Electrical Engineering
University of Siegen

Summary

Sources of error

After formulating a problem the typical steps for its solution are

- modelling,
- discretization,
- developing an algorithm,
- and implementation.

Each of these steps may introduce errors. We have focused on errors introduced in the algorithm.
Part 1: Rounding errors arising from the representation

$$\underbrace{(-1)^s}_{\text{sign}} \cdot \underbrace{0.c_1 c_2 \ldots c_p}_{\text{mantissa in a certain basis}} \cdot \underbrace{b^{\pm q}}_{\text{basis } b\text{, exponent } q}$$

of a number in a computer.

Errors

When approximating some x by some $\tilde{x}$ the **relative error** is

$$\frac{\|x - \tilde{x}\|}{\|x\|}$$

If we ignore bounds on the exponent q, we can approximate any number up to **machine precision** ϵ, i.e., it holds for $x \neq 0$ that

$$\left| \frac{\mathrm{rd}(x) - x}{x} \right| \leq \epsilon := \frac{b^{1-p}}{2}.$$

Since rounding has to happen after each elementary operation, errors can still be amplified in a computation, most strikingly in the **catastrophic cancellation** that can occur when subtracting two numbers that are almost equal:

$$(1 \cdot 10^{-3} \oplus 1) \oplus (-1) = 1 \oplus (-1) = 0$$

Error amplification

If $x \in \mathbb{R}$ and $\tilde{x} \in \mathbb{R}$ have a certain relative/absolute error, i.e.

$$|x - \tilde{x}| \leq \delta_{abs} \quad \text{or} \quad \frac{|x - \tilde{x}|}{|x|} \leq \delta_{rel} \text{ for } x \neq 0$$

what is the relative/absolute error in the output of a function f, i.e.

$$|f(x) - f(\tilde{x})| \quad \text{or} \quad \left| \frac{f(x) - f(\tilde{x})}{f(x)} \right| \text{ for } f(x) \neq 0?$$

Error amplification

If we ignore higher order terms, the relative error $\frac{\tilde{x}_i - x_i}{x_i}$ of each component amplifies the relative error for the result by $M_i = \left| \frac{x_i}{f(x)} \frac{\partial f}{\partial x_i}(x) \right|$, i.e.

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \sum_{i=1}^{n} M_i \cdot \left| \frac{\tilde{x}_i - x_i}{x_i} \right|.$$

Least-squares estimates

A vector $x \in \mathbb{R}^m$ minimizes

$$\|Ax - b\|_2^2$$

for $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$ if and only if it satisfies the **Gaussian Normal Equation**

$$A^T Ax = A^T b.$$

We proved this using some tools from linear algebra.

Since the least-squares solution does not have to be unique, a common strategy is to pick the least-squares solution with smallest norm, called **minimal-norm solution**.

But how do we solve linear systems like the above?

Exact solutions

A common way to solve a linear system $Ax = b$ exactly is transform A to an upper triangular matrix. This can be achieved by the **Gaussian Elimination Algorithm:**

Set $A^{(1)} = A$, $b^{(1)} = b$.

- For $i = 1$ to $n - 1$, if $a_{i,i}^{(i)} \neq 0$
  - for $j = i + 1$ to n

    subtract $l_{ji} = \frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$ times the i-th row of $(A^{(i)}, b^{(i)})$

    from the j-th row to obtain $(A^{(i+1)}, b^{(i+1)})$.

The additional introduction of **pivoting**, i.e., permuting the rows of $A^{(i)}$ in every iteration in such a way that

$$|a_{ii}^{(i)}| = \max_{k \geq i} |a_{ki}^{(i)}|$$

makes the Gaussian elimination stable for most practical purposes, and yields a permutation matrix P as well as left and right triangular matrices L and R such that $PA = LR$.

Errors in linear systems

If

$$\frac{\|x - x^{\delta}\|_2}{\|x\|_2} = \delta,$$

and A is invertible, then

$$\frac{\|Ax - Ax^{\delta}\|_2}{\|Ax\|_2} \leq \underbrace{\|A\| \cdot \|A^{-1}\|}_{=:\kappa(A)} \cdot \delta,$$

where $\kappa(A)$ is the **condition number of A**.

If A is symmetric, then matrix norm

$$\|A\| = \sup_{\|z\|_2 \leq 1} \|Az\|_2$$

is equal to the largest eigenvalue in magnitude, i.e.,

$$\|A\| = \max_i |\lambda_i(A)|.$$

Iterative solutions of linear systems

Common way to solve $Ax = b$ iteratively is to formulate a **fixed-point iteration**, e.g.

$$x^{k+1} = x^k + B^{-1}(b - Ax^k)$$

for some matrix B that is easy to invert, e.g.

- $A = \tau I$, the **Richardson iteration**,
- $A = D$ for D being the diagonal of A, the **Jacobi method**,
- $A = L$ for L being the lower left of A, the **Gauß-Seidel method**.

Such methods are guaranteed to converge as soon as

$$\|B^{-1}(A - B)\| < 1.$$

Conjugate Gradient

A powerful and **matrix-free** strategy is given by the **conjugate gradient** method.

Let A be symmetric positive definite. We say that u and v are **conjugate** with respect to A if

$$\langle u, Av \rangle = 0.$$

We showed that having n many mutually conjugate vectors $\{p_0, \ldots, p_{n-1}\}$ means the solution of $Ax = b$ is

$$x = \sum_{i=0}^{n-1} \frac{\langle p_i, b \rangle}{\langle p_i, Ap_i \rangle} p_i.$$

The conjugate gradient method is a clever way of constructing such p_i!

Finding Eigenvalues

Eigenvector and eigenvalue

Let $A \in \mathbb{R}^{n \times n}$. We call $\lambda \in \mathbb{R}$ a (real) **eigenvalue** of A, if there exists a $0 \neq v \in \mathbb{R}^n$ such that

$$Av = \lambda v.$$

Any $v \neq 0$ that satisfies the above equation is called an **eigenvector** of A to the eigenvalue λ.

Symmetric matrices A are **diagonalizable**, i.e., there exists a diagonal matrix Σ and an invertible matrix U such that

$$A = U \Sigma U^{-1}.$$

The columns $U_{:,i}$ of U are eigenvectors of A. For symmetric matrix one even finds U to be orthogonal, i.e. $U^{-1} = U^T$.

Power method

Key insight: By iteratively applying the matrix A the component of the eigenvalue with largest magnitude dominates!

Power method. For $k = 1, \ldots, n_{\max}$ compute

- $\tilde{v}^k = Av^{k-1}$
- $v^k = \frac{\tilde{v}^k}{\langle v^0, \tilde{v}^k \rangle}$

Return $u_1^{approx} = \frac{v^{n_{\max}}}{\| v^{n_{\max}} \|_2}$ and $\lambda_1^{approx} = \frac{\langle \tilde{v}^{n_{\max}}, v^0 \rangle}{\langle v^{n_{\max}-1}, v^0 \rangle}$.

For the above method to work we need

- A to have a basis of eigenvectors, e.g. A symmetric.
- $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$
- $\alpha_1 \neq 0$.

All eigenvectors?

How do we compute all eigenvectors of A?

Idea: Apply the power method to multiple vectors and ensure that they remain mutually orthogonal.

Gram-Schmidt method: How to make n linearly independent vectors $V = (v_1, \dots v_n)$ orthogonal!

Assume we have $v_1^1, \dots, v_l^1$ that all satisfy $\|v_i^1\| = 1$ and $\langle v_i^1, v_j^1 \rangle = 0$ for $i \neq j$.

Compute

$$v_{l+1}^1 = \frac{1}{r_{l+1\,l+1}} \left(\tilde{v}_{l+1}^1 - \sum_{i=1}^{l} r_{il+1} v_i^1 \right)$$

for $r_{il+1} = \langle v_i^1, \tilde{v}_{l+1}^1 \rangle$, and $r_{l+1\,l+1}$ such that $\|v_{l+1}^1\| = 1$.

Ultimately yields $V = QR$ for an orthogonal matrix Q and an upper right matrix R!

QR-algorithm

We'd like to start with n vectors written in a matrix $V^0 = V$, and iteratively

1. Apply A to obtain $\tilde{V}^{k+1} = AV^k$.
2. Compute an orthogonalization of $\tilde{V}^{k+1}$ to obtain the next iterate V^{k+1}, $\tilde{V}^{k+1} = V^{k+1}R^{k+1}$.

A short computation shows that we can hope for

$$A^k := (V^k)^T AV^k$$

to converge to the eigenvalues on its diagonal.

Interestingly, this simplifies the algorithm to

1. Compute $A^k = QR$,
2. set $A^{k+1} = RQ$,

known as the **QR-algorithm**. The product of all Q yields the corresponding eigenvectors.

Nonlinear equation

For the eigenvalue problem we developed a specific solution to finding all λ_i such that

$$\det(A - \lambda_i I) = 0,$$

but heavily exploited our linear algebra knowledge.

But how do we solve

$$g(u) = 0$$

for some nonlinear $g : \mathbb{R}^n \to \mathbb{R}^n$ in general?

Reformulate as a **fixed-point problem**: Find suitable G such that

$$G(u) = u \quad \Leftrightarrow \quad g(u) = 0.$$

Fixed-point methods

We investigated

1. the simple choice $G(u) = u - \tau g(u)$,
2. and the choice $G(u) = u - (Jg(u))^{-1} g(u)$ based on Taylor's expansion.

We saw that the minimization of

$$\min_{u \in \mathbb{R}^n} E(u)$$

is a possible area of application, as

$$g(u) := \nabla E(u) = 0$$

is a necessary condition for the above problem.

In this context the first choice, $G(u) = u - \tau g(u)$, is called **gradient descent** method, and the second choice, $G(u) = u - (Jg(u))^{-1} g(u)$, is called **Newton method**.

Optimization methods

We found that

- the **Newton method** converges very fast (typically quadratic) in a local neighborhood of each optimal point, but is not guaranteed to converge at all!
- the **gradient descent** method only requires

$$\|\nabla E(u) - \nabla E(v)\| \leq L\|u - v\|, \qquad \forall u, v$$

for a constant $L \geq 0$, with a step size $\tau < 1/L$ to converge globally, but does not converge nearly as fast.

The idea of **globalized Newton methods** therefore is to try to take a Newton update step, check if a certain stability criterion is met, and fall back to gradient descent if the criterion is violated.

Interpolation

We discussed how to approximate a function $f : \mathbb{R} \to \mathbb{R}$ by simpler functions, e.g., by **polynomials**.

For $n+1$ distinct sample points $(x_i, f(x_i))$ there exists a polynomial p of degree n such that $p(x_i) = f(x_i)$. This can readily be seen from the **Lagrange form**

$$p(x) = \sum_{j=0}^{n} l_{jn}(x) f(x_j)$$

$$l_{jn}(x) = \frac{(x - x_0) \dots (x - x_{j-1})(x - x_{j+1}) \dots (x - x_n)}{(x_j - x_0) \dots (x_j - x_{j-1})(x_j - x_{j+1}) \dots (x_j - x_n)}.$$

We found that approximating a function f on an interval by equidistantly sampled points x_i often does not yield good results!

Chebyshev sampling

Minimizing an upper bound on the difference of a function f to the polynomial p of degree $n-1$ on $[-1,1]$ motivates the use of the **Chebyshev roots**

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \quad i = 1,\ldots,n,$$

which indeed improved our results in practice.

This lead us to investigate approximations more systematically: p^* is a **best approximation** of a function f in some set T of candidate functions with respect to the norm $\|\cdot\|$ if

$$\|p^* - f\| \leq \|p - f\| \qquad \forall p \in T.$$

Best approximations

If the norm $\|\cdot\|$ considered for the best approximation is induced by an inner product, the situation is easy:

Best approximation

Let $v_1, \ldots, v_n$ be pairwise orthogonal with $\|v_i\| = 1 \ \forall i$. For any $f \in V$ it holds that

$$\sum_{i=1}^{n} \langle v_i, f \rangle v_i$$

is the best approximation of f in $\text{span}(v_1, \ldots, v_n)$.

Other cases are more difficult, but can sometimes still be analyzed in detail, e.g.

Equi-oscillation property

A polynomial p of degree n is the best L^∞ approximation to a continuous function $f : [-1,\ 1] \to \mathbb{R}$ if and only if there exists a set of $n+2$ distinct points x_i and $\sigma \in \{-1, 1\}$ such that

$$f(x_i) - p(x_i) = \sigma\ (-1)^i\ \|f - p\|_\infty, \qquad i = 0, \ldots, n+1,$$

Splines

A lot of trouble has been caused by highly oscillatory behavior of higher order polynomials as the number of sampling points increases.

A different (more powerful) approach is to use **piecewise-polynomial approximations** also called **splines**.

A **linear spline** is nothing but the piecewise linear function

$$s(x) = \frac{x - x_i}{x_{i+1} - x_i} f(x_{i+1}) + \left(1 - \frac{x - x_i}{x_{i+1} - x_i}\right) f(x_i), \quad \text{for } x \in [x_i, x_{i+1}[.$$

Another commonly used form are **natural C^2-splines** where one approximates f on each interval by a cubic function, requires the overall function to be two times continuously differentiable, and demands the derivative at both end points of the entire interval to be zero.

Integration

If you have understood interpolation/approximation, you also have understood integration:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} f^i_{approx}(x)\,dx$$

For piecewise linear approximations one finds the **trapezoidal rule**

$$\int_a^b f(x)\,dx \approx h\left(\frac{f(t_0)}{2} + \frac{f(t_n)}{2} + \sum_{i=1}^{n-1} f(t_i)\right).$$

Approximations of different order with equidistant sampling points are known as the **Newton-Cotes** formulas.

Higher accuracy can be achieved by different sampling strategies for smooth functions.

Different integrals

Higher dimesional integrals simply reduce the a sequence of 1d integrals, e.g., if $\Omega = [a, b] \times [a, b]$, then

$$\int_\Omega f(x)\, dx = \int_a^b \int_a^b f(x_1, x_2)\, dx_1\, dx_2.$$

Line integral

The **line integral** of a function $f : U \subset \mathbb{R}^n \to \mathbb{R}$ along a piecewise smooth curve $C \subset U$ is defined as

$$\int_a^b f(r(t)) \cdot |r'(t)|\, dt$$

where $r : [a, b] \to C$ is an arbitrary bijective parametrization of the curve C such that $r(a)$ and $r(b)$ give the endpoints of C and $a < b$.

This also reduces the line integral to a type of integral we know how to approximate.

The end

Any questions about anything?