Recap lecture MATH3230A

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Numerical integration

General quadrature rule for approximating the integral of $f : [a, b] \to \mathbb{R}$ with $n+1$ points $x_0, \ldots, x_n \in [a, b]$

$$\int_a^b f(x)\,\mathrm{dx} \approx \alpha_0 f(x_0) + \cdots + \alpha_n f(x_n),$$

such that it is exact for polynomials of certain degrees.

Numerical integration

General quadrature rule for approximating the integral of $f : [a, b] \to \mathbb{R}$ with $n+1$ points $x_0, \dots, x_n \in [a, b]$

$$\int_a^b f(x)\,\mathrm{dx} \approx \alpha_0 f(x_0) + \cdots + \alpha_n f(x_n),$$

such that it is exact for polynomials of certain degrees.

Newton–Cotes rules:

- Equally-spaced points $a = x_0 < x_1 < \cdots < x_n = b$, $x_i = a + ih$ with $h = \frac{b-a}{n}$.
- Exact for polynomials of degree $\leq n$.

Gaussian (Gauss–Legendre) rules:

- Points are roots of certain (Legendre) polynomials (not equally-spaced).
- Exact for polynomials of degree $\leq 2n + 1$.

Newton–Cotes

2-point Newton–Cotes (aka Trapezoidal rule) $x_0 = a$, $x_1 = b$:

$$\int_a^b f(x)\,\mathrm{dx} \approx \frac{b-a}{2}(f(a) + f(b)).$$

Composite Trapezoidal rule with $h = x_{i+1} - x_i = \frac{b-a}{n}$

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{dx} \approx \frac{h}{2}(f(x_i) + f(x_{i+1})), \quad \int_a^b f(x)\,\mathrm{dx} \approx \sum_{i=0}^{n-1} \frac{h}{2}(f(x_i) + f(x_{i+1}))$$

Newton–Cotes

3-point Newton–Cotes (aka Simpson's rule) $x_0 = a$, $x_1 = \frac{1}{2}(a+b)$, $x_2 = b$:

$$\int_a^b f(x)\,\mathrm{dx} \approx \frac{b-a}{6}(f(a) + 4f((a+b)/2) + f(b)).$$

Composite Simpson's rule

$$\int_a^b f(x)\,\mathrm{dx} \approx \sum_{i=0}^{n-1} \frac{h}{6}(f(x_i) + 4f((x_i + x_{i+1})/2) + f(x_{i+1}))$$

Newton–Cotes

General procedure:

- Set $x_i = a + ih$, $h = \frac{b-a}{n}$ as the equally-spaced partition of $[a, b]$.
- Since Newton–Cotes must be exact for all polynomials of degree $\leq n$, compute for $0 \leq k \leq n$

$$I_k := \int_a^b x^k \,\mathrm{dx} = \sum_{i=0}^{n} \alpha_i x_i^k.$$

- Deduce α_i by solving a linear system

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} I_0 \\ I_1 \\ \vdots \\ I_n \end{pmatrix}$$

- Alternatively

$$\alpha_i = \int_a^b l_i(x) \,\mathrm{dx} = \int_a^b \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \,\mathrm{dx}.$$

Gaussian/Gauss–Legendre

Procedure for $f : [-1, 1] \to \mathbb{R}$:

- Set $x_0, \ldots, x_n$ as the roots of the $(n+1)^{th}$ Legendre polynomial $P_{n+1}(x)$.
- Compute

$$\alpha_i = \int_a^b l_i(x)\,\mathrm{dx} = \int_a^b \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,\mathrm{dx}.$$

Gaussian/Gauss–Legendre

Procedure for $f : [-1, 1] \to \mathbb{R}$:

- Set $x_0, \dots, x_n$ as the roots of the $(n+1)^{th}$ Legendre polynomial $P_{n+1}(x)$.
- Compute

$$\alpha_i = \int_a^b l_i(x)\,\mathrm{dx} = \int_a^b \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,\mathrm{dx}.$$

Procedure for $f : [a, b] \to \mathbb{R}$:

- Use linear transformation

$$L(x) = \frac{b+a}{2} + \frac{b-a}{2}x.$$

- Transform the integrals:

$$\int_a^b f(y)\,dy = \int_{-1}^{1} f(L(x))\frac{b-a}{2}\,\mathrm{dx} =: \int_{-1}^{1} g(x)\,\mathrm{dx} \approx \sum_{i=0}^{n} \alpha_i g(x_i).$$

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Interpolating data

General problem: given a data set of a function $f : \mathbb{R} \to \mathbb{R}$:

x	x_0	x_1	$\cdots$	x_n
$f(x)$	f_0	f_1	$\cdots$	f_n

find a polynomial $p(x)$ of degree $\leq n$ such that

$$p(x_i) = f_i.$$

Interpolating data

General problem: given a data set of a function $f : \mathbb{R} \to \mathbb{R}$:

x	x_0	x_1	$\cdots$	x_n
$f(x)$	f_0	f_1	$\cdots$	f_n

find a polynomial $p(x)$ of degree $\leq n$ such that

$$p(x_i) = f_i.$$

Method 1: Vandermonde interpolation - Solve the matrix-vector problem

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$$

and set

$$V(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_n x^n.$$

Interpolating data

General problem: given a data set of a function $f : \mathbb{R} \to \mathbb{R}$:

x	x_0	x_1	$\cdots$	x_n
$f(x)$	f_0	f_1	$\cdots$	f_n

find a polynomial $p(x)$ of degree $\leq n$ such that

$$p(x_i) = f_i.$$

Method 2: Lagrange interpolation - Define

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j},$$

and set

$$L(x) = f_0 l_0(x) + f_1 l_1(x) + \cdots + f_n l_n(x).$$

Interpolating data

General problem: given a data set of a function $f : \mathbb{R} \to \mathbb{R}$:

x	x_0	x_1	$\cdots$	x_n
$f(x)$	f_0	f_1	$\cdots$	f_n

find a polynomial $p(x)$ of degree $\leq n$ such that

$$p(x_i) = f_i.$$

Method 3: Newton interpolation - Compute the divided difference table and obtain

$$c_k = f[x_0, \ldots, x_k] \text{ for } 0 \leq k \leq n,$$

and set

$$N(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n \prod_{j=0}^{n-1}(x - x_j)$$

Properties

- Any two polynomials $p(x)$ and $q(x)$ of deg $\leq n$ agreeing on $n+1$ points must coincide.
- The Lagrange interpolation $L(x)$ and the Newton interpolation $N(x)$ are the same polynomial.
- The divided difference is symmetric with respect to perturbations in the argument:

$$f[x_0, \ldots, x_k] = f[z_0, \cdots, z_k]$$

  for any perturbation $(z_0, \cdots, z_k)$ of $(x_0, \ldots, x_k)$.
- The ordering in the divided difference table does not matter.
- The error of interpolating $f(x)$ with polynomial $p(x)$ of degree $\leq n$ with $n+1$ distinct points is

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0) \cdots (x - x_n).$$

Chebyshev polynomials

Aim: to find the best points $x_0 < x_1 < \cdots < x_n$ in $[a, b]$ such that the interpolation error

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)\cdots(x - x_n)$$

is minimized.

Chebyshev polynomials

Aim: to find the best points $x_0 < x_1 < \cdots < x_n$ in $[a, b]$ such that the interpolation error

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)\cdots(x - x_n)$$

is minimized.

For a monic polynomial $g : [-1, 1] \to \mathbb{R}$ of degree $n + 1$, it holds that

$$\max_{x\in[-1,1]} |g(x)| \geq 2^{-n},$$

and so for $f : [-1, 1] \to \mathbb{R}$, the best error estimate is

$$|f(x) - p(x)| \leq \frac{\max_{x\in[-1,1]} |f^{(n+1)}(x)|}{2^n(n+1)!}$$

if we choose $x_0, \ldots, x_n$ as roots of the Chebyshev polynomial $T_{n+1}(x)$.

Chebyshev polynomials

The Chebyshev polynomials are defined recursively:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Properties include:

- $T_n(x) = \cos(n\cos^{-1}(x))$ for $x \in [-1, 1]$.
- $|T_n(x)| \leq 1$ for $x \in [-1, 1]$.
- $T_n(\cos\frac{j\pi}{n}) = (-1)^j$ for $0 \leq j \leq n$.
- $T_n(\cos\frac{2j-1}{2n}\pi) = 0$ for $0 \leq j \leq n$.
- The monic polynomial $\widehat{T}_{n+1}(x) = 2^{-n}T_{n+1}(x)$ satisfies

$$\max_{x\in[-1,1]} |\widehat{T}_{n+1}(x)| = 2^{-n}.$$

Hermite's interpolation

General problem: given a data set of a function $f : \mathbb{R} \to \mathbb{R}$ and its derivative:

x	x_0	x_1	$\cdots$	x_n
$f(x)$	f_0	f_1	$\cdots$	f_n
$f'(x)$	f'_0	f'_1	$\cdots$	f'_n

find a polynomial $p(x)$ of degree $\leq 2n+1$ such that

$$p(x_i) = f_i, \quad p'(x_i) = f'_i.$$

Hermite's interpolation

Method 1 - Lagrange form: Set

$$u_i(x) = (1 - 2l_i'(x_i)(x - x_i))l_i^2(x), \quad v_i(x) = (x - x_i)l_i^2(x),$$

where

$$u_i(x_j) = v_i'(x_j) = \begin{cases} 0 & \text{if } j \neq i, \\ 1 & \text{if } j = i, \end{cases}$$
$$u_i'(x_j) = 0, \quad v_i(x_j) = 0 \text{ for } 0 \leq j \leq n.$$

Then, define

$$H_{2n+1}(x) = \sum_{i=0}^{n} f_i u_i(x) + \sum_{i=0}^{n} f_i' v_i(x).$$

Hermite's interpolation

Method 2 - Newton form: Set

$$z_{2i} = z_{2i+1} = x_i \text{ for } i = 0, \dots n,$$

and

$$f[z_{2i}, z_{2i+1}] = f_i' \text{ for } i = 0, \dots, n,$$

in the table of divided difference. Then, define

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \dots, z_k](x - z_0) \cdots (x - z_{k-1}).$$

Hermite's interpolation

Method 2 - Newton form: Set

$$z_{2i} = z_{2i+1} = x_i \text{ for } i = 0, \dots n,$$

and

$$f[z_{2i}, z_{2i+1}] = f_i' \text{ for } i = 0, \dots, n,$$

in the table of divided difference. Then, define

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \dots, z_k](x - z_0) \cdots (x - z_{k-1}).$$

Error estimate for $f \in C^{2n+2}[a, b]$ and any $x \in [a, b]$:

$$f(x) - H_{2n+1}(x) = \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}(x - x_0)^2 \cdots (x - x_n)^2.$$

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Vector and matrix norms

For $x \in \mathbb{R}^n$ and $p \geq 1$

$$\|x\|_p = \begin{cases} \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1 \leq i \leq n} |x_i| & \text{if } p = \infty. \end{cases}$$

For $A \in \mathbb{R}^{n \times n}$, the induced p-matrix norm is

$$\|A\|_p = \max_{\|x\|_p = 1} \|Ax\|_p.$$

Vector and matrix norms

For $x \in \mathbb{R}^n$ and $p \geq 1$

$$\|x\|_p = \begin{cases} \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} & \text{if } p < \infty, \\ \max_{1\leq i\leq n} |x_i| & \text{if } p = \infty. \end{cases}$$

For $A \in \mathbb{R}^{n\times n}$, the induced p-matrix norm is

$$\|A\|_p = \max_{\|x\|_p=1} \|Ax\|_p.$$

Properties:

- $\|A\|_1$ is the maximum column sum.
- $\|A\|_\infty$ is the maximum row sum.
- $\|A\|_2 = \sqrt{\lambda_{\max}(A^TA)}$.
- $\|AB\|_p \leq \|A\|_p\|B\|_p$.
- $\|Ax\|_p \leq \|A\|_p\|x\|_p$.

Sensitivity of linear systems

Given invertible matrices A and $\tilde{A}$ with vector b, and solutions x and $\tilde{x}$:

$$Ax = b, \quad \tilde{A}\tilde{x} = b,$$

we seek an upper bound on the relative error $\frac{\|x-\tilde{x}\|}{\|x\|}$.

Sensitivity of linear systems

Given invertible matrices A and $\tilde{A}$ with vector b, and solutions x and $\tilde{x}$:

$$Ax = b, \quad \tilde{A}\tilde{x} = b,$$

we seek an upper bound on the relative error $\frac{\|x-\tilde{x}\|}{\|x\|}$.

First result:

$$\text{If } \frac{\|x-\tilde{x}\|}{\|x\|} \leq \theta < 1, \text{ then } \frac{\|x-\tilde{x}\|}{\|\tilde{x}\|} \leq \frac{\theta}{1-\theta}.$$

Second result: Set $E = A - \tilde{A}$ and use $Ax = b = \tilde{A}\tilde{x} = A\tilde{x} + E\tilde{x}$ to get

$$x - \tilde{x} = A^{-1}E\tilde{x}.$$

Then,

$$\frac{\|x-\tilde{x}\|}{\|\tilde{x}\|} \leq \|A^{-1}E\| = \|A^{-1}\tilde{A} - I\| \quad \Rightarrow \quad \frac{\|x-\tilde{x}\|}{\|x\|} \leq \frac{\|A^{-1}E\|}{1-\|A^{-1}E\|}.$$

Sensitivity of linear systems

Given invertible matrices A and $\tilde{A}$ with vector b, and solutions x and $\tilde{x}$:

$$Ax = b, \quad \tilde{A}\tilde{x} = b,$$

we seek an upper bound on the relative error $\frac{\|x-\tilde{x}\|}{\|x\|}$.

Use

$$\|A^{-1}E\| \leq \|A^{-1}\|\|A\|\frac{\|E\|}{\|A\|} = \kappa(A)\frac{\|A-\tilde{A}\|}{\|A\|},$$

where the condition number is $\kappa(A) := \|A^{-1}\|\|A\| \geq 1$, we have

$$\frac{\|x-\tilde{x}\|}{\|x\|} \leq \frac{\kappa(A)\frac{\|A-\tilde{A}\|}{\|A\|}}{1-\kappa(A)\frac{\|E\|}{\|A\|}} =: c(E)\kappa(A)\frac{\|A-\tilde{A}\|}{\|A\|}.$$

Solving systems of equations

To solve

$$Ax = b, \text{ where } A \in \mathbb{R}^{n \times n}, \quad b \in \mathbb{R}^n,$$

we have

Solving systems of equations

To solve

$$Ax = b, \text{ where } A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^n,$$

we have

- Forward substitution if A is lower triangular.
- Backward substitution if A is upper triangular.
- Cholesky factorization $A = U^T U$ into upper triangular U if A is symmetric and positive definite (SPD). Then solve

$$Ux = y, \quad U^T y = b.$$

- LU factorization $A = LU$ into lower tri. L and upper tri. U if A is not SPD. Then solve

$$Ux = y, \quad Ly = b.$$

- LU with partial pivot in case the pivot at some stage is zero! This leads to the factorization

$$PA = LU$$

for some permutation matrix P.

Non-square systems

If $A \in \mathbb{R}^{m\times n}$, $m \neq n$, then there may not be a solution/infinitely many solutions to

$$Ax = b \text{ for } b \in \mathbb{R}^m, x \in \mathbb{R}^n.$$

- Overdetermined case $m > n$. Choose the solution x_* with smallest error $\|Ax - b\|_2$. The solution x_* is given by the normal equation:

  $$x_* = (A^T A)^{-1} A^T b,$$

  obtained by differentiating

  $$\frac{d}{dt} f(x_* + ty)|_{t=0} = 0, \text{ where } f(x) = \|Ax - b\|_2.$$

- Undetermined case $m < n$. Choose the solution x^* with the smallest 2-norm amongst all other solutions. Obtained by differentiating the Lagrangian:

  $$L(x, \mu) = \|x\|_2^2 - \mu^T (Ax - b)$$

  and set all partial derivatives to zero. The solution is

  $$x^* = A^T (AA^T)^{-1} b.$$

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Scalar nonlinear equations

To solve

$$f(x) = 0$$

for some nonlinear function $f : \mathbb{R} \to \mathbb{R}$, we have

Scalar nonlinear equations

To solve

$$f(x) = 0$$

for some nonlinear function $f : \mathbb{R} \to \mathbb{R}$, we have

- Bisection method
  - Only needs $f(a)f(b) < 0$ for some interval $[a, b]$ where f is continuous.
  - Always converges if $x_0 \in (a, b)$.
  - R-linear convergence.
- Newton's method
  - Need f to be differentiable and $f'(x_*) \neq 0$.
  - Converges if x_0 is close to x_*.
  - Q-Quadratic convergence.
- Quasi-Newton methods
  - Replace $f'(x_k)$ in Newton's method with simpler approximations.
  - Converges if x_0 is close to x_*.
  - Q-linear convergence (Constant slope) or Order $p = \frac{1+\sqrt{5}}{2}$ (Secant method)

Fixed-point iterative methods

Newton's method, constant slope method and Secant method are special cases of fixed point iterative methods:

$$x_{k+1} = \varphi(x_k) \quad k = 0, 1, 2, \ldots.$$

Fixed-point iterative methods

Newton's method, constant slope method and Secant method are special cases of fixed point iterative methods:

$$x_{k+1} = \varphi(x_k) \quad k = 0, 1, 2, \ldots.$$

Main result for fixed-point iterative methods:

- If $|\varphi'(x_*)| < 1$, then there exists $\delta = \delta(x_*)$ such that if $x_0 \in [x_* - \delta, x_* + \delta]$, the fixed-point iterative method converges.
- If $\varphi'(x_*) \neq 0$, the convergence is Q-linear.
- If $\varphi'(x_*) = \varphi''(x_*) = \cdots = \varphi^{(p-1)}(x_*) = 0$ but $\varphi^{(p)}(x_*) \neq 0$, then the convergence is of order p.

Systems of nonlinear equations

To solve the nonlinear system of equations

$$F(x) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{pmatrix} = 0$$

we have

Systems of nonlinear equations

To solve the nonlinear system of equations

$$F(x) = \begin{pmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{pmatrix} = 0$$

we have

- Newton's method: (Quadratic local convergence)

$$x_{k+1} = x_k - (DF(x_k))^{-1} F(x_k).$$

- Broyden's method: (Linear local convergence)

$$x_{k+1} = x_k - A_k^{-1} F(x_k).$$

- Steepest descent: (Linear global convergence)

$$x_{k+1} = x_k - \alpha_k \nabla g(x_k), \quad g(x) = \frac{1}{2} \|F(x)\|_2^2,$$

$$\alpha_k = \arg\min_{s \geq 0} g(x_k - s \nabla g(x_k)).$$

Broyden's method

$$x_{k+1} = x_k - A_k^{-1} F(x_k).$$

Key properties of matrix A_k:

- (Secant condition) $A_k(x_k - x_{k-1}) = F(x_k) - F(x_{k-1})$.
- (Rank-one update) $A_k = A_{k-1} + p_k \otimes d_{k-1}$, $d_{k-1} = A_{k-1}^{-1} F(x_{k-1})$.
- (Orthogonal property) $A_k y = A_{k-1} y$ for all $y \cdot (x_k - x_{k-1}) = 0$.

Broyden's method

$$x_{k+1} = x_k - A_k^{-1} F(x_k).$$

Key properties of matrix A_k:

- (Secant condition) $A_k(x_k - x_{k-1}) = F(x_k) - F(x_{k-1})$.
- (Rank-one update) $A_k = A_{k-1} + p_k \otimes d_{k-1}$, $d_{k-1} = A_{k-1}^{-1} F(x_{k-1})$.
- (Orthogonal property) $A_k y = A_{k-1} y$ for all $y \cdot (x_k - x_{k-1}) = 0$.

The "good" Broyden method: Given x_0 and invertible A_0,

$$d_k = A_k^{-1} F(x_k) \quad \mapsto \quad x_{k+1} = x_k + d_k$$

$$\mapsto \quad A_{k+1} = A_k + \frac{F(x_k) - F(x_{k-1}) - A_k d_k}{d_k \cdot d_k} \otimes d_k.$$

Broyden's method

$$x_{k+1} = x_k - A_k^{-1} F(x_k).$$

Key properties of matrix A_k:

- (Secant condition) $A_k(x_k - x_{k-1}) = F(x_k) - F(x_{k-1})$.
- (Rank-one update) $A_k = A_{k-1} + p_k \otimes d_{k-1}$, $d_{k-1} = A_{k-1}^{-1} F(x_{k-1})$.
- (Orthogonal property) $A_k y = A_{k-1} y$ for all $y \cdot (x_k - x_{k-1}) = 0$.

The "good" Broyden method: Given x_0 and invertible A_0,

$$d_k = A_k^{-1} F(x_k) \quad \mapsto \quad x_{k+1} = x_k + d_k$$
$$\mapsto \quad A_{k+1} = A_k + \frac{F(x_k) - F(x_{k-1}) - A_k d_k}{d_k \cdot d_k} \otimes d_k.$$

The "bad" Broyden method employs the Sherman–Morrison formula to get A_{k+1}^{-1} directly without inverting a matrix:

$$d_k = A_k^{-1} F(x_k) \quad \mapsto \quad x_{k+1} = x_k + d_k$$
$$\mapsto \quad A_{k+1}^{-1} = A_k^{-1} + \frac{A_k^{-1}(F(x_k) \otimes d_k) A_k^{-1}}{d_k \cdot d_k + d_k \cdot A_k^{-1} F(x_k)}.$$

Steepest descent

Instead of solving $F(x) = 0$, the Steepest descent method finds the minimum of $g(x) = \frac{1}{2}\|F(x)\|_2^2$.

Steepest descent

Instead of solving $F(x) = 0$, the Steepest descent method finds the minimum of $g(x) = \frac{1}{2}\|F(x)\|_2^2$.

Procedure: Starting from x_0,

- Obtain search direction d_k and search step α_k.
- Update $x_{k+1} = x_k + \alpha_k d_k$ such that $g(x_{k+1}) < g(x_k)$.

Steepest descent

Instead of solving $F(x) = 0$, the Steepest descent method finds the minimum of $g(x) = \frac{1}{2}\|F(x)\|_2^2$.

Procedure: Starting from x_0,

- Obtain search direction d_k and search step α_k.
- Update $x_{k+1} = x_k + \alpha_k d_k$ such that $g(x_{k+1}) < g(x_k)$.

Example

- One choice of search direction is the negative gradient $d_k = -\nabla g(x_k)$.
- One choice of search step is such that

$$g(x_k + \alpha_k d_k) \leq g(x_k + sd_k) \text{ for any } s \in \mathbb{R}.$$

- If α_k is chosen as above, then $d_k \cdot d_{k+1} = 0$ (Zig-zag motion).

Outline

- Numerical integration
- Polynomial interpolation
- Linear systems of equations
- Nonlinear equations/systems
- Floating point arithmetic

Scientific notation

A decimal can be represented as

$$a = \pm r \times 10^n,$$

where

- $r \in [0.1, 1)$ is the mantissa.
- $n \in \mathbb{Z}$ is the exponent.

Scientific notation

A decimal can be represented as

$$a = \pm r \times 10^n,$$

where

- $r \in [0.1, 1)$ is the mantissa.
- $n \in \mathbb{Z}$ is the exponent.

A binary number can be represented as

$$a = \pm (q)_2 \times 2^{\tilde{m}},$$

where

- q is the mantissa.
- $\tilde{m}$ is the exponent.

Single/double precision format

The single precision floating point format with 32-bits for normalized binary numbers is

$$s|m_1 m_2 \cdots m_8|f_1 f_2 \cdots f_{23} \quad \mapsto \quad a = (-1)^s (1.f_1 \ldots f_{23})_2 \times 2^{(m_1 \ldots m_8)_2 - 127}$$

- Biased exponent $(m_1 \ldots m_8)_2 - 127$ is used to represent an equal number of non-negative and negative exponents.
- The cases $(m_1 \ldots m_8)_2 = (0 \ldots 0)_2$ or $(1 \ldots 1)_2$ are reserved for special values such as 0, ∞ and NaN.
- Smallest positive normalized number $a_{\min}$ and largest finite normalized number $a_{\max}$ are

$$a_{\min} \mapsto 0|0 \cdots 01|0 \cdots 0 \mapsto 2^{-126},$$
$$a_{\max} \mapsto 0|1 \cdots 10|1 \cdots 1 \mapsto (2 - 2^{-23}) \times 2^{127}.$$

Single/double precision format

The double precision floating point format with 64-bits for normalized binary numbers is

$$s|m_1 m_2 \cdots m_{11}|f_1 f_2 \cdots f_{52} \quad \mapsto \quad a = (-1)^s (1.f_1 \ldots f_{52})_2 \times 2^{(m_1 \ldots m_{11})_2 - 1023}$$

- Double precision is used if we need to have twice as much accuracy than single precision.
- Smallest positive normalized number a_{min} and largest finite normalized number a_{max} are

$$a_{\text{min}} \mapsto 0|0 \cdots 01|0 \cdots 0 \mapsto 2^{-1022},$$
$$a_{\text{max}} \mapsto 0|1 \cdots 10|1 \cdots 1 \mapsto (2 - 2^{-52}) \times 2^{1023}.$$

Rounding/Chopping

Two ways to obtain from a decimal number $x = x_0.x_1 \ldots x_m$ with m digits to a decimal number with $n < m$ digits:

- Rounding:

$$x_r = \begin{cases} x_0.x_1 \ldots x_n & \text{if } x_{n+1} \in \{0,1,2,3,4\}, \\ x_0.x_1 \ldots x_n + 10^{-n} & \text{if } x_{n+1} \in \{5,6,7,8,9\}. \end{cases}$$

- Chopping:

$$x_c = x_0.x_1 \ldots x_n.$$

Rounding/Chopping

Two ways to obtain from a decimal number $x = x_0.x_1 \dots x_m$ with m digits to a decimal number with $n < m$ digits:

- Rounding:

$$x_r = \begin{cases} x_0.x_1 \dots x_n & \text{if } x_{n+1} \in \{0, 1, 2, 3, 4\}, \\ x_0.x_1 \dots x_n + 10^{-n} & \text{if } x_{n+1} \in \{5, 6, 7, 8, 9\}. \end{cases}$$

- Chopping:

$$x_c = x_0.x_1 \dots x_n.$$

Estimates on relative errors:

$$\frac{|x - x_r|}{|x|} \leq \frac{1}{2} \times 10^{-n}, \quad \frac{|x - x_c|}{|x|} \leq 10^{-n}.$$

Similar when rounding/chopping binary numbers.

Machine precision

The machine precision/machine epsilon ε_M can be defined in two ways:

Machine precision

The machine precision/machine epsilon ε_M can be defined in two ways:

- The upper bound on the relative error of rounding a number a in between a_{min} and $a_{\text{min},2}$ (or $a_{\text{max},2}$ and a_{max}):

$$\frac{|a_{\text{min}} - a|}{|a|} \leq \varepsilon_M.$$

- The upper bound on the relative error of approximating a given real number x by a nearby machine number $\hat{x}$:

$$\frac{|x - \hat{x}|}{|x|} \leq \varepsilon_M.$$

Machine precision

The machine precision/machine epsilon ε_M can be defined in two ways:

- The upper bound on the relative error of rounding a number a in between a_{min} and $a_{\text{min},2}$ (or $a_{\text{max},2}$ and a_{max}):

$$\frac{|a_{\text{min}} - a|}{|a|} \leq \varepsilon_M.$$

- The upper bound on the relative error of approximating a given real number x by a nearby machine number $\hat{x}$:

$$\frac{|x - \hat{x}|}{|x|} \leq \varepsilon_M.$$

Easier way: The machine epsilon is $\varepsilon_M = 2^{-y}$ where y is the number of bits reserved for the manitssa.

Loss of significance

Any instance of creating a number x where

- $x < a_{\text{min}}$ leads to underflow, and x is set to zero.
- $x > a_{\text{max}}$ leads to overflow, and the computation is halted.

Loss of significance

Any instance of creating a number x where

- $x < a_{\text{min}}$ leads to underflow, and x is set to zero.
- $x > a_{\text{max}}$ leads to overflow, and the computation is halted.

Loss of significance occurs during the subtraction of two nearly equal numbers and the effects of finite precision arithmetic used in the calculation:

Loss of significance

Any instance of creating a number x where

- $x < a_{\text{min}}$ leads to underflow, and x is set to zero.
- $x > a_{\text{max}}$ leads to overflow, and the computation is halted.

Loss of significance occurs during the subtraction of two nearly equal numbers and the effects of finite precision arithmetic used in the calculation: E.g. Evaluating $f(x) = \sqrt{x+1} - \sqrt{x}$ at $x = 100$ to 6 significant digits:

$\sqrt{101} - \sqrt{100} = 0.0499000$ (computed from rounding $\sqrt{101}$ to 6 sign. digits),
$\sqrt{101} - \sqrt{100} = 0.0498756$ (true value to 6 sign. digits)

leading to a loss of 4 digits of accuracy.

Loss of significance

Any instance of creating a number x where

- $x < a_{\min}$ leads to underflow, and x is set to zero.
- $x > a_{\max}$ leads to overflow, and the computation is halted.

Loss of significance occurs during the subtraction of two nearly equal numbers and the effects of finite precision arithmetic used in the calculation: E.g. Evaluating $f(x) = \sqrt{x+1} - \sqrt{x}$ at $x = 100$ to 6 significant digits:

$\sqrt{101} - \sqrt{100} = 0.0499000$ (computed from rounding $\sqrt{101}$ to 6 sign. digits),
$\sqrt{101} - \sqrt{100} = 0.0498756$ (true value to 6 sign. digits)

leading to a loss of 4 digits of accuracy.

Remedy? Rewrite expression that does not involve subtraction/use double precision.

Error analysis

For a non-machine number x, its closest machine number $fl(x)$ satisfies

$$fl(x) = x(1+\varepsilon) \text{ for } |\varepsilon| \leq \varepsilon_M.$$

Error analysis

For a non-machine number x, its closest machine number $fl(x)$ satisfies

$$fl(x) = x(1+\varepsilon) \text{ for } |\varepsilon| \leq \varepsilon_M.$$

This relation is used to analyse the relative errors we make when performing computer arithmetic that do not obey the usual rules of arithmetic due to rounding.

- Forward error analysis measures the relative error between $x \odot y$ and $fl(fl(x) \odot fl(y))$:

$$\frac{|x \odot y - fl(fl(x) \odot fl(y))|}{|x \odot y|} \leq C\varepsilon_M.$$

- Backward error analysis is concerned with showing the computed value $\hat{z}$ of $x \odot y$ is an exact calculation with perturbed data:

$$\hat{z} = (x+\delta_x) \odot (y+\delta_y) \text{ with } |\delta_x|, |\delta_y| \leq \varepsilon_M$$