Computer Arithmetic

Simplified form for representing a floating-point number x in **base** b:

$$x = (f)_b \times b^{(e)_b}$$

$f = \pm(.\, d_1 d_2 \cdots d_t)_b$ **mantissa** (or **significand**); $0 \le f < 1$.
$e = \pm(c_{s-1} c_{s-2} \cdots c_0)_b$ integer **exponent** (or **characteristic**); $E_{\min} \le e \le E_{\max}$
A computer in which numbers are represented as above is said to have t base-b digits **precision**. The exponent governs the **range** of representable numbers.

Terms: **normalized** mantissa, **significant** digits, OFL (overflow level, $N_{\max}$), UFL (underflow level, $N_{\min}$), **overflow**, **underflow**

The real numbers that are exactly representable as floating-point numbers are discrete, belong to $[-N_{\max}, -N_{\min}] \cup \{0\} \cup [N_{\min}, N_{\max}]$, and are denser towards 0.

For the rest of the real numbers:
those in $[-N_{\max}, -N_{\min}] \cup [N_{\min}, N_{\max}]$ are rounded/chopped (x represented as $fl(x)$);
those in $(-\infty, -N_{\max}) \cup (N_{\max}, \infty)$ overflow;
those in $(-N_{\min}, 0) \cup (0, N_{\min})$ underflow.

Computer Arithmetic

Relative round-off error (representation error): $\delta = \dfrac{fl(x) - x}{x}$

Bound for δ: $|\delta| \le \frac{1}{2} b^{1-t}$, i.e. $fl(x)$ is correct in t significant b-digits.

Terms: **absolute** error, **relative** error

All computer operations are designed so that **the error of a computation is the round-off error of the correct result**.

Machine epsilon $\varepsilon_{\text{mach}}$: The smallest (non-normalised) floating-point number with the property $1 + \varepsilon_{\text{mach}} > 1$.

Bounds for $\varepsilon_{\text{mach}}$ and δ:

$$\frac{1}{2} b^{1-t} \le \varepsilon_{\text{mach}} \le b^{1-t} \text{ and } -\varepsilon_{\text{mach}} \le \delta \le \varepsilon_{\text{mach}}$$

$$0 < \text{UFL} < \varepsilon_{\text{mach}} < \text{OFL}$$

Computer Arithmetic

Saturation: The phenomenon in which a non-zero number is added to another and the latter is left unchanged.

Catastrophic cancellation: The phenomenon in which adding nearly opposite (or subtracting nearly equal) numbers results in having no (or very few) correct digits (i.e. the past errors in the nearly opposite or nearly equal numbers, propagate from the least significant digits to the most significant ones).

Condition number of a function: $\kappa_f(x) = |\dfrac{x f'(x)}{f(x)}|$; measure of how the error in x propagates in $f(x)$ (error propagation in computation)

$$|\text{relative forward error}| \approx \text{condition number} \times |\text{relative backward error}|$$

$$\frac{y - \hat{y}}{y} \approx \kappa_f \frac{x - \hat{x}}{x}, \; y = f(x)$$

It does not change by re-writing a certain computation in an equivalent way.

Stability refers to the error propagation in a numerical algorithm, i.e. the particular way a certain computation is carried out, and may change if the computation is performed by a different algorithm.

Matrices, solving square linear systems

Matrix terminology:
triangular (upper/lower), unit triangular (upper/lower), tridiagonal, (l, u)-banded,
permutation, elementary Gauss transformation,
orthogonal, positive definite, diagonally dominant, symmetric

Important algorithms:
Gauss elimination,
Gauss elimination with partial pivoting (applicable to all matrices, gives L unit lower triangular, U upper triangular and P permutation, such that $PA = LU$),
back substitution, forward substitution

Important flops counts:

matrix	LU	LU piv (part)	LU piv compl.	f/s	b/s	sol of m sys.	inverse
general $n \times n$	$\frac{n^3}{3}$	$\frac{n^3}{3}$	$\frac{2n^3}{3}$	$\frac{n^2}{2}$	$\frac{n^2}{2}$	$\frac{n^3}{3} + mn^2$	n^3
(l, u)-banded	nlu	$2nlu$	$-$	ln	un	$2nlu + m(l+u)n$	$n^2(l+u)$
symmetric	$n^3/6$						

General $n \times m$ matrix times $m \times k$ matrix (or vector for $k = 1$): nmk

Norms, condition numbers

Definition of a norm: three properties (see notes)

Common *vector norms*: max (infinity) $\|x\|_\infty \equiv \max_{i=1}^{n} \{|x_i|\}$,

Euclidean (2-norm) $\|x\|_2 \equiv \sqrt{(x, x)} \equiv (\sum_{i=1}^{n} x_i^2)^{1/2}$, one-norm $\|x\|_1 \equiv \sum_{i=1}^{n} |x_i|$.

For any vector $x \in \mathbb{R}^n$, we have $\|x\|_\infty \le \|x\|_2 \le \|x\|_1$.

Common *matrix norms*: p-norms $\|A\|_p \equiv \max_{x \ne 0} \{ \frac{\|Ax\|_p}{\|x\|_p} \}$, $p = 1, 2, \infty$,

$\|A\|_\infty = \max_{i=1}^{n} \{ \sum_{j=1}^{n} |a_{ij}| \}$, $\|A\|_1 = \max_{j=1}^{n} \{ \sum_{i=1}^{n} |a_{ij}| \}$

For the p-norms, another three properties hold (see notes).

Condition number of a non-singular matrix A: $\kappa_a(A) = \|A\|_a \|A^{-1}\|_a$. It is a measure of the relative sensitivity of the solution x of $Ax = b$ to relative changes in A and b:

$$\frac{\|x - \hat{x}\|_a}{\|x\|_a} \le \kappa_a(A) \left(\frac{\|b - \hat{b}\|_a}{\|b\|_a} + \frac{\|A - \hat{A}\|_a}{\|A\|_a} + \frac{\|r\|_a}{\|b\|_a} \right)$$

Nonlinear equations -- terms and concepts

General form: $f(x) = 0$

Multiplicity of root: If $f \in \mathbb{C}^m$ and $f(x^*) = 0$, $f'(x^*) = 0, \cdots, f^{(m-1)}(x^*) = 0$, but $f^{(m)}(x^*) \ne 0$, for some $m \ge 1$, then x^* is a root of multiplicity m.

General form of a $n \times n$ system of nonlinear equations: $f: \mathbb{R}^n \to \mathbb{R}^n$, $\bar{f}(\bar{x}) = \bar{0}$

Jacobian matrix: $(J(\bar{x}))_{ij} = \frac{\partial f_i}{\partial x_j}(\bar{x})$

Fixed point x of function $g(x)$: $x = g(x)$.

Contraction in set $S \subset \mathbb{R}^n$: a function $g: \mathbb{R}^n \to \mathbb{R}^n$ for which there is constant λ, with $0 \le \lambda < 1$, such that

$$\|g(x) - g(z)\| \le \lambda \|x - z\|, \ \forall\ x, z \in S. \qquad (1)$$

See Theorems 1-4 about existence and uniqueness or roots or fixed points.

Numerical methods for solving nonlinear equations

```
General nonlinear solver
guess $x^{(0)}$ (and possibly $x^{(-1)}$ or more)
for $k = 1, \cdots$, maxit
   compute $x^{(k)}$ using previous approximations and information from $f$
   if stopping criterion satisfied exit, endif
endfor
```

Rate of convergence of sequences / iterative methods

Sequence $x^{(0)}, x^{(1)}, \cdots$ converges to x^*, with rate of convergence β and asymptotic error constant C:

$$\lim_{k \to \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^\beta} = C$$

See Theorem 6 about rate of convergence of fixed-point iteration methods.

Numerical methods for solving nonlinear equations

• *Bisection* method

Bisection is applicable when f is continuous and there are two points L and R ($L < R$), such that $f(L) f(R) < 0$. Bisection chooses $M = L + (R - L)/2$ as next approximation and continues to $[L, M]$ if $f(L) f(M) < 0$, or to $[M, R]$ if $f(M) f(R) < 0$. Bisection always converges (when applicable) and is of first order. Requires one function evaluation per iteration.

• *Newton's* method

Newton's is applicable if f is differentiable and $f'(x^{(k)}) \ne 0$:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \qquad (5)$$

Converges if f twice differentiable and if $x^{(0)}$ «close enough» to root. When it converges, it is usually of second order. Requires one function and one derivative evaluation per iteration.

Numerical methods for solving nonlinear equations

• General form of *fixed-point iteration*
Assume we have found (or been given) function $g(x)$ such that

$$f(x) = 0 \iff x = g(x) \tag{3}$$

Then, the associated fixed-point iteration method computes

$$x^{(k+1)} = g(x^{(k)})$$

Newton's is a fixed-point iteration method with $g(x) = x - f(x)/f'(x)$.
See Theorems 4b, 5 and 6 about convergence of fixed-point iteration methods.

• *Secant* method
Secant is applicable if f is continuous and $f(x^{(k)}) \neq f(x^{(k-1)})$:

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} \tag{7}$$

Secant does not always converge; when it converges it is of order approximately 1.6. Requires one function evaluation per iteration.

• *Newton's* for systems
Newton's is applicable if all components of f are differentiable (with respect to each variable) and $J(\bar{x}^{(k)})$ is nonsingular:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - J^{-1}(\bar{x}^{(k)})\,\bar{f}(\bar{x}^{(k)}). \tag{8}$$

Polynomial interpolation

Generic interpolation problem: Given data (x_i, y_i), $i = 0, \cdots, n$, find a function (possibly a polynomial) $p(x)$, that interpolates the data, i.e. $p(x_i) = y_i$, $i = 0, \cdots, n$.

Existence and uniqueness theorem: Given data (x_i, y_i), $i = 0, \cdots, n$, with x_i distinct, there exists a unique polynomial $p_n(x)$ of degree n or less that interpolates the data.

Polynomial interpolation error formula: Let $p_n(x)$ be the polynomial of degree n or less that interpolates f at x_i, $i = 0, \cdots, n$. If f has $n+1$ continuous derivatives, then, for any x,

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

for some $\xi \in \mathrm{ospr}\{x, x_0, x_1, \cdots, x_n\}$, that depends on x.

Notes: If $a = \min_i \{x_i\}$ and $b = \max_i \{x_i\}$, then $\mathrm{ospr}\{x_0, x_1, \cdots, x_n\} = (a, b)$.
If $a \le x \le b$, $\mathrm{ospr}\{x, x_0, x_1, \cdots, x_n\} = (a, b)$.
If $x < a$, $\mathrm{ospr}\{x, x_0, x_1, \cdots, x_n\} = (x, b)$.
If $b < x$, $\mathrm{ospr}\{x, x_0, x_1, \cdots, x_n\} = (a, x)$.

Construction of polynomial interpolants

Properly posed polynomial interpolation problem: Given data (x_i, y_i), $i = 0, \cdots, n$, with x_i distinct, find the (unique) polynomial $p_n(x)$, of degree n or less that interpolates the data.

Three ways of contructing:
(1) Monomials, Vandermonde matrix: easy for small n, very inefficient and inaccurate (unstable) for large n, construction requires the solution of an $(n+1) \times (n+1)$ dense and ill-conditioned linear system, easy to evaluate (Horner's), differentiate and integrate.

(2) Lagrange basis functions, explicit formula: often convenient for mathematical manipulation, easy to construct, not very stable, reasonably easy to evaluate, hard to differentiate and integrate.

(3) Newton's basis functions, recursive algorithm, NDD table: reasonably efficient to construct, stable, easy to evaluate (Horner's), relatively hard to differentiate and integrate, adding data is easy.

Problems with polynomial interpolation

Polynomial interpolants of high degree

− oscillate a lot close to the endpoints;
− lead to ill-conditioned computations;
− are sensitive to small perturbations of the coefficients;
− are costly to construct (NDD table takes $O(n^2)$ time for n data points) and evaluate (the evaluation on one point takes $O(n)$ time, while the number of evaluation points is often much larger than the number of data points).

Polynomial interpolants do not guarantee that the error decreases as the number of data increases.

Chebyshev data points minimize the $\prod_{i=0}^{n}(x - x_i)$ part of the error among all choices of $n+1$ data points in a given interval, and partly only address the oscillatory behaviour problem.

Piecewise polynomial interpolation is a good alternative, that resolves all the above problems.

Piecewise Polynomial Interpolation

Piecewise polynomial of degree k defined with respect to knots x_i, $i = 0, \cdots, n$:

$$p(x) = \begin{cases} a_{01} + a_{11}x + a_{21}x^2 + \cdots + a_{k1}x^k & \text{for } x_0 \le x < x_1 \\ a_{02} + a_{12}x + a_{22}x^2 + \cdots + a_{k2}x^k & \text{for } x_1 \le x < x_2 \\ \cdots\cdots\cdots & \\ a_{0n} + a_{1n}x + a_{2n}x^2 + \cdots + a_{kn}x^k & \text{for } x_{n-1} \le x \le x_n \end{cases}$$

There are $n(k+1)$ coefficients -- free parameters, degrees of freedom -- in the representation of p. Piecewise polynomials may not be continuous or may not have continuous derivatives of some order on the knots.

By imposing continuity conditions on the interior knots, some coefficients are no longer free parameters.

If we impose continuity conditions up to derivatives of order $k-1$, we get splines. Splines have $n(k+1) - (n-1)k = n + k$ coefficients -- free parameters, degrees of freedom.

Example: **Linear splines** are piecewise polynomials of degree 1 which are $\mathbb{C}^0$ continuous on the knots x_i, $i = 1, \cdots, n-1$, and have $n+1$ degrees of freedom.

Example: **Cubic splines** are piecewise polynomials of degree 3 which are $\mathbb{C}^2$ continuous on the knots x_i, $i = 1, \cdots, n-1$, and have $n+3$ degrees of freedom.

Linear splines

A **linear spline** $L(x)$ takes the form

$$L(x) = \begin{cases} L_1(x) \equiv a_{01} + a_{11}x & \text{for } x_0 \le x < x_1 \\ L_2(x) \equiv a_{02} + a_{12}x & \text{for } x_1 \le x < x_2 \\ \cdots\cdots\cdots & \\ L_n(x) \equiv a_{0n} + a_{1n}x & \text{for } x_{n-1} \le x \le x_n \end{cases}$$

and satisfies the $n-1$ *continuity conditions*

$$L_i(x_i) = L_{i+1}(x_i), \quad i = 1, \cdots, n-1. \qquad \text{(A)}$$

The **linear spline interpolant** $L(x)$ of data (x_i, y_i), $i = 0, \cdots, n$, is the linear spline (i.e. a pp of degree 1 satisfying (A)), satisfying the *interpolation conditions*

$$L(x_i) = y_i, \quad i = 0, \cdots, n. \qquad \text{(B)}$$

The linear spline interpolant $L(x)$ of data (x_i, y_i), $i = 0, \cdots, n$, is given by

$$L(x) = \left\{ y_{i-1} \frac{x - x_i}{x_{i-1} - x_i} + y_i \frac{x - x_{i-1}}{x_i - x_{i-1}} \right. \text{ for } x_{i-1} \le x \le x_i,\ i = 1, \cdots, n. \qquad \text{(C)}$$

Thus, $L(x)$ is given by a ready-to-evaluate formula. (No construction cost.)
The evaluation of a linear spline requires $O(1)$ time for one evaluation point.

Cubic splines

A **cubic spline** $C(x)$ takes the form

$$C(x) = \begin{cases} C_1(x) \equiv a_{01} + a_{11}x + a_{21}x^2 + a_{31}x^3 & \text{for } x_0 \le x < x_1 \\ C_2(x) \equiv a_{02} + a_{12}x + a_{22}x^2 + a_{32}x^3 & \text{for } x_1 \le x < x_2 \\ \cdots\cdots\cdots & \\ C_n(x) \equiv a_{0n} + a_{1n}x + a_{2n}x^2 + a_{3n}x^3 & \text{for } x_{n-1} \le x \le x_n \end{cases} \qquad (0)$$

and satisfies the $3(n-1)$ *continuity conditions*

$$C_i(x_i) = C_{i+1}(x_i), \quad i = 1, \cdots, n-1, \qquad \text{(0a)}$$

$$C_i'(x_i) = C_{i+1}'(x_i), \quad i = 1, \cdots, n-1, \qquad \text{(0b)}$$

$$C_i''(x_i) = C_{i+1}''(x_i), \quad i = 1, \cdots, n-1. \qquad \text{(0c)}$$

A **cubic spline interpolant** $C(x)$ of data (x_i, y_i), $i = 0, \cdots, n$, is a cubic spline (i.e. a pp of degree 3 satisfying (0a)-(0b)-(0c)), satisfying the *interpolation conditions*

$$C(x_i) = y_i, \quad i = 0, \cdots, n, \qquad (1)$$

and a set of two other conditions, referred to as *end-conditions*.

Cubic splines -- Typical sets of end-conditions

Clamped (derivative) end-conditions:

$$C'(x_i) = y_i', \quad i = 0, n, \qquad \text{(2a)}$$

if y_i', $i = 0, n$, are given.

Approximate clamped (derivative) end-conditions:

$$C'(x_i) = \text{approximation to } y_i', \quad i = 0, n \qquad \text{(2b)}$$

if y_i', $i = 0, n$, are not given (the approximations to y_i' can be formed by cubic polynomial interpolation based on 4 endpoint values).

Natural end-conditions:

$$C''(x_i) = 0, \quad i = 0, n. \qquad \text{(2c)}$$

Not-a-knot end-conditions:

$$C''' \text{ continuous on } x_i, \quad i = 1, n-1. \qquad \text{(2d)}$$

These are equivalently written as

$$C_i'''(x_i) = C_{i+1}'''(x_i), \quad i = 1, n-1.$$

Note: only **one** of these sets (pairs) of end-conditions is satisfied.

Cubic splines

Note: There is no ready-to-evaluate (no cost) formula for cubic spline interpolants, such as (C) for linear spline interpolants.

It can be shown that the computation to construct a cubic spline interpolant (i.e. to compute the coefficients a_{ji}, $j = 0, 1, 2, 3$, $i = 1, \cdots, n$, of $C(x)$) given data (x_i, y_i), $i = 0, \cdots, n$, is equivalent to solving an $(n+1) \times (n+1)$ tridiagonal symmetric linear system, i.e. $O(n)$. This linear system is usually well-conditioned.

The evaluation of a cubic spline requires $O(1)$ time for one evaluation point.

Spline interpolation error bounds

Linear spline interpolant error bound: Let $L(x)$ be the linear spline w.r.t. x_i, $i = 0, \cdots, n$, interpolating f at x_i, $i = 0, \cdots, n$. If $f(x) \in \mathbb{C}^2$, then for $x \in [x_0, x_n]$,

$$|f(x) - L(x)| \le \frac{1}{8} \max_{x_0 \le x \le x_n} |f^{(2)}(x)| \max_{i=1}^{n} (x_i - x_{i-1})^2. \qquad (7)$$

The linear spline error decreases by a factor of approximately 4, when n doubles. Linear spline interpolation is of order 2.

Cubic spline interpolant error bound: Let $C(x)$ be the (clamped) cubic spline w.r.t. x_i, $i = 0, \cdots, n$, interpolating f at x_i, $i = 0, \cdots, n$. If $f(x) \in \mathbb{C}^4$, then for $x \in [x_0, x_n]$,

$$|f(x) - C(x)| \le \frac{5}{384} \max_{x_0 \le x \le x_n} |f^{(4)}(x)| \max_{i=1}^{n} (x_i - x_{i-1})^4. \qquad (19)$$

The cubic spline error decreases by a factor of approximately 16, when n doubles. Cubic spline interpolation is of order 4.

Piecewise polynomial interpolants guarantee that the error decreases as the number of data increases.

Piecewise polynomial interpolants do not (normally) suffer from oscillatory behaviour.

Notice the different use of term ‘‘order’’ (and ‘‘rate’’) for convergence of spline interpolants and for convergence of iterative methods for nonlinear equations.