Floating Point Numbers

For any base β:

$$x = \pm r \times \beta^n, \qquad 1 \le r < \beta.$$

Single-Precision IEEE (32-bit):

$$\underbrace{\text{(1 bit) sign} = s}_{\text{sign}} \quad \underbrace{\text{(8 bits) biased exponent} = c}_{\text{exponent}} \quad \underbrace{\text{(23 bits) mantissa} = f}_{\text{mantissa}}$$

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

Steps for IEEE:

1. Convert into binary.
2. Write into normalized scientific notation (binary $\times 2^k$) where k makes it of the form $1.xxxx$.
3. Compute C, F, S: $C = 127 + k$, F is fractional bits of $1.F$, and $S = 0$ (positive), 1 (negative).

Error Propagation:

Let $x \in \mathbb{R}$ and its floating point representation is written as $fl(x) = x(1+\delta)$.

Absolute error $= |fl(x) - x|$

Relative error $= \dfrac{|fl(x) - x|}{|x|}$

Loss of Significance: Subtracting close numbers cancels leading bits and can create large relative error. Reformulate expressions to avoid subtracting nearly equal quantities.

Numerical Approximations

We can use Taylor and Maclaurin series to represent smooth functions.

$$f(x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(c)}{i!}(x-c)^i$$

Finite difference approximations:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h} \quad \text{(Forward Euler)}$$

$$f'(a) \approx \frac{f(a) - f(a-h)}{h} \quad \text{(Backward Euler)}$$

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h} \quad \text{(Central Difference)}$$

Additionally, you can solve order of accuracy by utilizing Taylor expansions:

$$f(a+h) = \sum_{i=0}^{\infty} \frac{f^{(i)}(a)}{i!}h^i, \qquad f(a-h) = \sum_{i=0}^{\infty} \frac{(-1)^i f^{(i)}(a)}{i!}h^i$$

Substitute into your formula, cancel terms, and the first nonzero neglected term gives truncation error $O(h^p)$.

Root Finding

Bisection

Given $f(x)$ and $a, b \in \mathbb{R}$ s.t. $f(a)$ and $f(b)$ satisfies **I.V.T.**
While stopping criterion* is **False**:

1. **Estimate** $x_n = \dfrac{a_n + b_n}{2}$.
2. **If** $|f(x_n)| < \epsilon$, return x_n.
3. **If** $f(x_n) \cdot f(a_n) < 0$, $a_{n+1} = a_n$ and $b_{n+1} = x_n$.
4. **Else**, $a_{n+1} = x_n$ and $b_{n+1} = b_n$.
5. $n = n + 1$, **Repeat**.

***Stopping criterion is either number of iterations,** $|b - a| < \epsilon$, **and** $|f(c)| < \epsilon$.

Error and Iteration Estimate for Bisection

Denote the interval of the n-th iteration as $[a_n, b_n]$, **then** the numerical error becomes

$$E_n = |c_n - r| \le \frac{1}{2^n}|b_0 - a_0|.$$

Given tolerance is ϵ, we want $E_n \le \epsilon$, namely

$$\frac{|b-a|}{2^n} \le \epsilon \quad \Longrightarrow \quad n \ge \log_2 \frac{b-a}{\epsilon}.$$

Fixed-Point Iteration

Let $f(x)$ be continuous on an interval $[a, b]$ that contains a root r, i.e., $f(r) = 0$. Rewrite the equation $f(x) = 0$ in the equivalent fixed-point form

$$x = x - f(x)g(x),$$

where $g : [a, b] \to \mathbb{R}$ and $g(r) \neq 0$.
Choose an initial guess $x_0 \in [a, b]$ and define the iterative scheme

$$x_{n+1} = x_n - f(x_n)g(x_n), \quad n = 0, 1, 2, \ldots$$

Continue the iteration until a prescribed tolerance is satisfied, for example

$$|x_{n+1} - x_n| < \varepsilon.$$

Error Relation

Let $E_n = |x_n - r|$ denote the error at step n. If g is differentiable, then by the Mean Value Theorem there exists ξ between x_n and r such that

$$E_{n+1} = |x_{n+1} - r| = |g(x_n) - g(r)| = |g'(\xi)|\, E_n.$$

Convergence Conditions

The iteration converges to r if:

1. $r = g(r)$,
2. $|g'(r)| < 1$ (local convergence),
3. $|g'(x)| < 1 \quad \forall x \in [a, b]$ (guaranteed convergence on $[a, b]$).

Newton's Method

An application of fixed point where $g(x) = \frac{1}{f'(x)}$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots$$

Choose an initial guess x_0 sufficiently close to r and iterate until a prescribed tolerance is met, for example

$$|x_{n+1} - x_n| < \varepsilon.$$

Requirements for Quadratic Convergence

If $f'(r) \neq 0$, Newton's method exhibits quadratic convergence.

Convergence Conditions

Newton's method converges to r if:

1. $f(r) = 0$,
2. $f'(r) \neq 0$ (simple root),
3. x_0 is sufficiently close to r,
4. f is twice continuously differentiable near r.

Secant Method

Approximates $f'(x_n)$ using finite differences.

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Algorithm:

1. Choose x_0, x_1.
2. Compute x_{n+1} using formula above.
3. Stop if $|x_{n+1} - x_n| < \varepsilon$.

Requires no derivative evaluation.

LU Decomposition

Suppose we have an $n \times n$ matrix where A is invertible.
Let

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ \cdot & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

The matrix U is the row-echelon form of A after elimination; multipliers fill L.

Algorithm (Doolittle, no pivoting)

Goal: $A = LU$ with L unit lower triangular and U upper triangular.

1. Set $L = I$, $U = A$.
2. For $k = 1, \ldots, n-1$:
   (a) Require pivot $U_{k,k} \neq 0$.
   (b) For $i = k+1, \ldots, n$: set multiplier $m_{ik} = U_{i,k}/U_{k,k}$.
   (c) Store $L_{i,k} = m_{ik}$.
   (d) Row update on U: $U_{i,k:n} = U_{i,k:n} - m_{ik}U_{k,k:n}$.
3. Solve systems using forward substitution ($Ly = b$), then backward substitution ($Ux = y$).

Requirements / limits:

- No-pivot LU can fail if a pivot is zero or very small.
- For stability in practice, use pivoting ($PA = LU$).

Gaussian Elimination with Partial Pivoting

Goal: Solve $Ax = b$ stably and/or compute $PA = LU$.

1. For each column $k = 1, \ldots, n-1$, choose pivot row

$$p = \arg\max_{i \geq k} |A_{i,k}|.$$

2. Swap rows k and p in A (and in b; record in permutation P).
3. If $A_{k,k} = 0$ after swap, matrix is singular (stop).
4. For $i = k+1, \ldots, n$: compute $m_{ik} = A_{i,k}/A_{k,k}$ and eliminate

$$A_{i,k:n} = A_{i,k:n} - m_{ik}A_{k,k:n}, \quad b_i = b_i - m_{ik}b_k.$$

5. Back-substitute from upper-triangular system.

Requirements / limits:

- Partial pivoting greatly improves stability vs no pivoting.
- Growth factor can still be large for rare matrices; complete pivoting is more robust but costlier.

Matrix Norms and Condition Number

For nonsingular A, condition number in norm $\|\cdot\|$:

$$\kappa(A) = \|A\| \, \|A^{-1}\| \geq 1.$$

Large $\kappa(A)$ means sensitivity to perturbations in data/roundoff.

Norms used in practice:

$$\|A\|_1 = \max_j \sum_i |a_{ij}| \quad \text{(column-sum norm)}$$

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| \quad \text{(row-sum norm)}$$

$$\|A\|_F = \left(\sum_{i,j} |a_{ij}|^2\right)^{1/2} \quad \text{(Frobenius norm)}$$

$$\|A\|_2 = \sigma_{\max}(A) = \sqrt{\lambda_{\max}(A^T A)} \quad \text{(2-norm)}$$

Computational steps:

1. Choose norm (often 1 or ∞ for cheap computation).
2. Compute $\|A\|$ from entries.
3. Compute/estimate $\|A^{-1}\|$ (usually via linear solves, not explicit inverse).
4. Return $\kappa(A) = \|A\| \, \|A^{-1}\|$.

Important limits:

- Explicitly forming A^{-1} is expensive and can be unstable.
- Different norms give different κ values, but all indicate conditioning.

Newton's Method for Systems

Let $F : \mathbb{R}^n \to \mathbb{R}^n$. We seek $F(x) = 0$.
Define the Jacobian:

$$J_F(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Newton iteration:

$$J_F(x_n)\, s_n = -F(x_n), \qquad x_{n+1} = x_n + s_n.$$

Quadratic convergence holds if:

1. $J_F(r)$ is nonsingular,
2. F is continuously differentiable,
3. x_0 sufficiently close to r.

Broyden's Method (for Systems)

Quasi-Newton method: avoid recomputing Jacobian every step.

Start with x_0, initial Jacobian approximation B_0 (often $J_F(x_0)$ or I).
For $n = 0, 1, 2, \dots$:

1. Solve linear system
$$B_n s_n = -F(x_n).$$
2. Update iterate: $x_{n+1} = x_n + s_n$.
3. Compute
$$y_n = F(x_{n+1}) - F(x_n).$$
4. Rank-1 update (good Broyden):
$$B_{n+1} = B_n + \frac{(y_n - B_n s_n)s_n^T}{s_n^T s_n}.$$
5. Stop when $\|F(x_{n+1})\| < \varepsilon$ or $\|s_n\| < \varepsilon$.

Requirements / limits:

- Need B_n nonsingular for the step solve.
- Usually superlinear (not quadratic like exact Newton near root).
- Often cheaper per iteration than Newton for large systems.