

NUMERICAL OPTIMIZER's

Numerical Optimization Methods Summary

Line Search Based Methods

General Formula:

$$x_{k+1} = x_k + \alpha_k p_k$$

Where p_k is the search direction and α_k is the step length determined by a line search.

1. Steepest Descent

Formula:

$$p_k = -\nabla f(x_k)$$

Explanation:

The most basic method, using the negative gradient as the search direction. It often converges slowly, especially for ill-conditioned problems, with a linear convergence rate.

2. Newton's Method (Unconstrained)

Formula:

$$\begin{aligned}\nabla^2 f(x_k) p_k^N &= -\nabla f(x_k) \\ x_{k+1} &= x_k + \alpha_k p_k^N \quad (\text{often } \alpha_k = 1)\end{aligned}$$

Explanation:

Uses a quadratic model of the function and steps to its minimum. Converges quadratically near a solution if the Hessian is positive definite but requires Hessian computation.

3. Newton's Method with Hessian Modification

Explanation:

Modifies the Hessian $\nabla^2 f_k$ when it's not positive definite to ensure a descent direction. Various strategies exist for this modification.

3a. Eigenvalue Modification

Formula:

$$\begin{aligned}B_k &= Q \Lambda' Q^T \quad (\text{where } \Lambda' \text{ has modified eigenvalues of } \nabla^2 f_k) \\ B_k p_k &= -\nabla f_k\end{aligned}$$

Explanation:

Computes the Hessian's spectral decomposition and replaces non-positive eigenvalues. Ensures the modified Hessian B_k is positive definite.

3b. Adding a Multiple of the Identity

Formula:

$$B_k = \nabla^2 f_k + \tau I \quad (\text{for sufficiently large } \tau > 0)$$

$$B_k p_k = -\nabla f_k$$

Explanation:

Adds a scaled identity matrix to the Hessian to make it positive definite. Simple but finding an appropriate τ can be complex.

3c. Modified Cholesky Factorization

Formula:

Factorize $P^T(\nabla^2 f_k + E)P = LL^T$ (E is diagonal correction)

Solve $LL^T(P^T p_k) = -P^T \nabla f_k$

Explanation:

Computes a Cholesky factorization while adding diagonal elements if needed. Ensures positive definiteness during the factorization process.

3d. Modified Symmetric Indefinite Factorization

Formula:

Factorize $P^T(\nabla^2 f_k + E)P = LBL^T$ (B is block diagonal)

Explanation:

Uses a symmetric indefinite factorization, adding modifications E to ensure sufficient positive definiteness. Handles indefinite Hessians directly.

Trust-Region Based Methods

General Subproblem:

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p \quad \text{s.t.} \quad \|p\| \leq \Delta_k$$

Where $m_k(p)$ is a model (usually quadratic) and Δ_k is the trust-region radius.

4. Dogleg Method

Formula:

Path combines Cauchy point p_k^C and Newton step p_k^N . Solution p_k lies on this path within radius Δ_k .

Explanation:

Approximates the trust-region subproblem solution using a path made of the steepest descent direction and the Newton direction. Efficient for small/medium problems.

5. Two-Dimensional Subspace Minimization

Formula:

Minimize $m_k(p)$ over $p \in \text{span}\{\nabla f_k, (B_k + \alpha I)^{-1} \nabla f_k\}$ within $\|p\| \leq \Delta_k$.

Explanation:

Solves the trust-region subproblem by minimizing the model over a 2D subspace. Captures essential information from gradient and Newton directions.

6. Trust-Region Newton Method

Formula:

Solve the general trust-region subproblem (see above) with $B_k = \nabla^2 f_k$ or approximation. Update

$x_{k+1} = x_k + p_k$ and adjust Δ_k .

Explanation:

The fundamental trust-region approach using a quadratic model based on the Hessian. Radius Δ_k is adjusted based on model agreement.

7. Steihaug's Method (Trust-Region Newton-CG)

Algorithm: Uses Conjugate Gradient (CG) method (Algorithm 7.1).

Explanation:

Applies CG iteratively to approximately solve the trust-region subproblem $B_k p = -\nabla f_k$. Stops early if trust boundary is hit or negative curvature found; suitable for large scale.

Conjugate Gradient Methods

8. Linear Conjugate Gradient Method

Formula: (Iterative updates involving α_k, β_k - See Algorithm 5.2)

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}, \quad \beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

Explanation:

Solves symmetric positive definite linear systems $Ax = b$ iteratively. Crucial for large-scale optimization, finds exact solution in n steps (exact arithmetic).

9. Nonlinear Conjugate Gradient Methods

Formula:

$$p_k = -\nabla f_k + \beta_k p_{k-1}$$

Explanation:

Adapts linear CG for general nonlinear optimization, avoiding Hessian storage. Different methods use different formulas for the scalar β_k .

9a. Fletcher-Reeves

Formula:

$$\beta_k^{FR} = \frac{\|\nabla f_k\|^2}{\|\nabla f_{k-1}\|^2}$$

Explanation:

One choice for β_k in nonlinear CG. Has strong convergence theory but can perform poorly in practice.

9b. Polak-Ribière

Formula:

$$\beta_k^{PR} = \frac{\nabla f_k^T (\nabla f_k - \nabla f_{k-1})}{\|\nabla f_{k-1}\|^2} \quad (\text{often use } \max(\beta_k^{PR}, 0))$$

Explanation:

Another choice for β_k , often performs better than Fletcher-Reeves. Convergence theory is weaker, modifications often needed.

Quasi-Newton Methods

General Idea: Build Hessian approximation B_k (or inverse H_k) using gradient information.

General Formula: Search direction $p_k = -H_k \nabla f_k$ or solve $B_k p_k = -\nabla f_k$.

10. BFGS Method

Formula: (Rank-two update for B_k or H_k , see Eq 6.17/6.19)

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Explanation:

Most popular quasi-Newton method using a rank-two update. Requires $s_k^T y_k > 0$ (via line search) and typically converges superlinearly.

11. SR1 Method

Formula: (Rank-one update, see Eq 6.24)

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

Explanation:

Uses a symmetric rank-one update. Can handle indefinite Hessians but update may need skipping if denominator is small.

12. Limited-Memory BFGS (L-BFGS)

Formula: Implicit update using last m pairs (s_i, y_i) . Direction via two-loop recursion (Algorithm 7.4).

Explanation:

Variant of BFGS for large problems, storing only recent m correction pairs instead of full matrix. Uses recursion to compute search direction.

13. Compact Representation of BFGS Updating

Formula: Represents B_k using compact matrix products (Eq 7.20-7.23).

Explanation:

Alternative way to represent BFGS updates. Useful for understanding L-BFGS and connections to other methods.

14. Sparse Quasi-Newton Updates

Explanation:

Attempts to maintain sparsity in the Hessian approximation B_k . Generally less effective than L-BFGS for large-scale problems.

Derivative-Free Optimization (DFO) Methods

15. Model-Based DFO Methods

Formula: Minimize model $m_k(p) \approx f(x_k + p)$ subject to $\|p\| \leq \Delta_k$.

Explanation:

Builds and minimizes a model (e.g., quadratic) based on function values at sampled points. Manages interpolation set geometry.

16. Coordinate Search Method

Algorithm: Explores along coordinate directions $\pm e_i$.

Explanation:

Very simple DFO method that searches along axes. Can be very slow but easy to implement.

17. Pattern-Search Methods

Algorithm: Explores a pattern/frame of points around x_k (See Section 9.3).

Explanation:

Maintains a step length and explores points in a predefined pattern. Reduces step if no improvement found; requires pattern to span space.

18. Nelder-Mead Method

Algorithm: Uses simplex transformations (reflection, expansion, contraction) (See Section 9.5).

Explanation:

Heuristic method maintaining a simplex of $n + 1$ points. Modifies simplex based on function values; convergence not guaranteed for all functions.

19. Implicit Filtering

Formula: Uses finite difference gradient $\nabla_h f(x)$ with decreasing h_k .

$$x_{k+1} = x_k - \alpha_k \nabla_{h_k} f(x_k)$$

Explanation:

Designed for noisy functions (smooth + high-frequency noise). Mimics steepest descent with decreasing finite difference steps to filter noise.

Least-Squares and Nonlinear Equations

20. Gauss-Newton Method

Formula:

$$(J_k^T J_k) p_k^{GN} = -J_k^T r_k$$

$$x_{k+1} = x_k + \alpha_k p_k^{GN}$$

Explanation:

For nonlinear least squares $\min \|r(x)\|^2$. Approximates Hessian by $J(x)^T J(x)$, works well for small residuals.

21. Levenberg-Marquardt Method (LMM)

Formula:

$$(J_k^T J_k + \lambda_k I) p_k^{LM} = -J_k^T r_k$$

$$x_{k+1} = x_k + p_k^{LM}$$

Explanation:

Improves Gauss-Newton using a damping parameter λ_k . Blends Gauss-Newton ($\lambda_k \rightarrow 0$) and steepest descent ($\lambda_k \rightarrow \infty$).

22. Newton's Method for Nonlinear Equations

Formula: Solve $r(x) = 0$.

$$J(x_k)p_k = -r(x_k)$$

$$x_{k+1} = x_k + p_k$$

Explanation:

Solves systems of equations using a linear model based on the Jacobian $J(x_k)$. Converges quadratically if $J(x_*)$ is nonsingular.

23. Inexact Newton Methods (for nonlinear equations)

Formula: Find p_k such that $\|J_k p_k + r_k\| \leq \eta_k \|r_k\|$.

Explanation:

Solves the Newton system $J_k p_k = -r_k$ approximately using an iterative solver. Useful for large-scale equation systems.

24. Broyden's Method

Formula: (Rank-one update for Jacobian approx A_k , Eq 11.24)

$$A_{k+1} = A_k + \frac{(y_k - A_k s_k) s_k^T}{s_k^T s_k}$$

$$A_k p_k = -r_k$$

Explanation:

Quasi-Newton method for $r(x) = 0$, avoids recomputing Jacobian J_k . Uses rank-one updates for the Jacobian approximation A_k .

25. Tensor Methods (for nonlinear equations)

Formula: Uses model $M_k(p) = r_k + J_k p + \frac{1}{2} T_k p p$.

Explanation:

Extends Newton's method using second-order (tensor) information. Aims for better global behavior or handling singular Jacobians.

26. Continuation/Homotopy Methods

Algorithm: Traces path of solutions for $H(x, \lambda) = 0$ from $\lambda = 0$ to $\lambda = 1$.

Explanation:

Finds solutions by following a path from a known solution of a simpler problem. Useful for difficult problems or finding multiple solutions.

Linear Programming (LP) Methods

27. Simplex Method

Algorithm: Iterates through feasible vertices (See Chapter 13).

Explanation:

Classic algorithm for LP. Moves between adjacent vertices of the feasible polytope, improving the objective until optimum is reached.

28. Dual Simplex Method

Algorithm: Applies simplex logic to the dual problem (See Section 13.6).

Explanation:

Maintains dual feasibility while working towards primal feasibility. Useful for re-optimization after adding constraints.

29. Primal-Dual Interior-Point Methods (Path-Following)

Formula: Follows central path defined by perturbed KKT conditions:

$$A^T \lambda + s = c, \quad Ax = b, \quad XSe = \mu e$$

Explanation:

Iterates stay strictly feasible ($x > 0, s > 0$), following central path towards optimality as $\mu \rightarrow 0$. Uses Newton steps on perturbed KKT system.

30. Potential-Reduction Methods

Formula: Minimize potential function, e.g., $\Phi_\rho(x, s) = \rho \log(x^T s) - \sum \log(x_i s_i)$.

Explanation:

Interior-point method that directly reduces a potential function measuring optimality and centrality. Does not explicitly track the central path.

Quadratic Programming (QP) Methods

31. Direct KKT Solution Methods (for Equality Constrained QP)

Formula: Solve KKT system:

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ -\lambda \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}$$

Explanation:

Solves equality constrained QP $\min \frac{1}{2} x^T G x + c^T x$ s.t. $Ax = b$ by directly solving the KKT linear system. Different factorization methods exist.

31a. Factoring the Full KKT System

Explanation:

Uses a symmetric indefinite factorization (like LDL^T) on the full KKT matrix. Handles indefinite G .

31b. Schur-Complement Method

Formula: Solve $(AG^{-1}A^T)\lambda = -(AG^{-1}c + b)$ for λ , then find x .

Explanation:

Eliminates x to solve for multipliers λ first. Requires G to be invertible.

31c. Null-Space Method

Formula: Decompose $x = Yx_Y + Zx_Z$. Solve reduced system $(Z^T G Z)x_Z = -Z^T(GYx_Y + c)$.

Explanation:

Reduces the problem by eliminating equality constraints using null-space Z . Solves a smaller system in the Z -space variables.

32. Iterative KKT Solution Methods (for Equality Constrained QP)

Explanation:

Uses iterative solvers (like CG) for the KKT system or related systems. Suitable for large-scale equality constrained QPs.

32a. CG Applied to the Reduced System

Explanation:

Applies Conjugate Gradient to solve the null-space system $(Z^T G Z)x_Z = \dots$. Requires $Z^T G Z$ to be positive definite.

32b. Projected CG Method

Explanation:

Applies Conjugate Gradient while projecting iterates to maintain feasibility $Ax = b$. Works directly in the original variable space.

33. Active-Set Methods (for Inequality Constrained QP)

Algorithm: Iteratively solves equality-constrained QP subproblems based on a working set W_k of active constraints (Algorithm 16.3).

Explanation:

For inequality constrained QP. Maintains an active set estimate, solves equality subproblems, and updates the set based on multipliers/violations.