1 Gradient-Based Optimization

1.1 General Algorithm for Smooth Functions

All algorithms for unconstrained gradient-based optimization can be described as follows. We start with iteration number $k = 0$ and a starting point, x_k.

1. **Test for convergence.** If the conditions for convergence are satisfied, then we can stop and x_k is the solution.
2. **Compute a search direction.** Compute the vector p_k that defines the direction in n-space along which we will search.
3. **Compute the step length.** Find a positive scalar, α_k such that $f(x_k + \alpha_k p_k) < f(x_k)$.
4. **Update the design variables.** Set $x_{k+1} = x_k + \alpha_k p_k$, $k = k + 1$ and go back to 1.

$$x_{k+1} = x_k + \underbrace{\alpha_k p_k}_{\Delta x_k} \tag{1}$$

There are two subproblems in this type of algorithm for each *major iteration*: computing the search direction p_k and finding the step size (controlled by α_k). The difference between the various types of gradient-based algorithms is the method that is used for computing the search direction.

1.2 Optimality Conditions

Consider a function $f(x)$ where x is the n-vector $x = [x_1, x_2, \ldots, x_n]^T$.

The *gradient vector* of this function is given by the partial derivatives with respect to each of the independent variables,

$$\nabla f(x) \equiv g(x) \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{2}$$

In the multivariate case, the gradient vector is perpendicular to the the *hyperplane* tangent to the contour surfaces of constant f.

Higher derivatives of multi-variable functions are defined as in the single-variable case, but note that the number of gradient components increase by a factor of n for each differentiation.

While the gradient of a function of n variables is an n-vector, the "second derivative" of an n-variable function is defined by n^2 partial derivatives (the derivatives of the n first partial derivatives with respect to the n variables):

$$\frac{\partial^2 f}{\partial x_i \partial x_j}, \quad i \neq j \quad \text{and} \quad \frac{\partial^2 f}{\partial x_i^2}, \quad i = j. \tag{3}$$

If the partial derivatives $\partial f/\partial x_i$, $\partial f/\partial x_j$ and $\partial^2 f/\partial x_i \partial x_j$ are continuous and f is single valued, then $\partial^2 f/\partial x_i \partial x_j$ exists and $\partial^2 f/\partial x_i \partial x_j = \partial^2 f/\partial x_j \partial x_i$. Therefore the second-order partial derivatives can be represented by a square symmetric matrix called the *Hessian matrix*,

$$\nabla^2 f(x) \equiv H(x) \equiv \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial^2 x_n}, \end{bmatrix} \tag{4}$$

which contains $n(n+1)/2$ independent elements.

If f is quadratic, the Hessian of f is constant, and the function can be expressed as

$$f(x) = \frac{1}{2} x^T H x + g^T x + \alpha. \tag{5}$$

As in the single-variable case the optimality conditions can be derived from the Taylor-series expansion of f about x^*:

$$f(x^* + \varepsilon p) = f(x^*) + \varepsilon p^T g(x^*) + \frac{1}{2}\varepsilon^2 p^T H(x^* + \varepsilon\theta p)p, \tag{6}$$

where $0 \leq \theta \leq 1$, ε is a scalar, and p is an n-vector.

For x^* to be a local minimum, then for any vector p there must be a finite ε such that $f(x^* + \varepsilon p) \geq f(x^*)$, i.e. there is a neighborhood in which this condition holds. If this condition is satisfied, then $f(x^* + \varepsilon p) - f(x^*) \geq 0$ and the first and second order terms in the Taylor-series expansion must be greater than or equal to zero.

As in the single variable case, and for the same reason, we start by considering the first order terms. Since p is an arbitrary vector and ε can be positive or negative, every component of the gradient vector $g(x^*)$ must be zero.

Now we have to consider the second order term, $\varepsilon^2 p^T H(x^* + \varepsilon\theta p)p$. For this term to be non-negative, $H(x^* + \varepsilon\theta p)$ has to be positive semi-definite, and by continuity, the Hessian at the optimum, $H(x^*)$ must also be positive semi-definite.

Necessary conditions (for a local minimum):

$$\|g(x^*)\| = 0 \quad \text{and} \quad H(x^*) \quad \text{is positive semi-definite.} \tag{7}$$

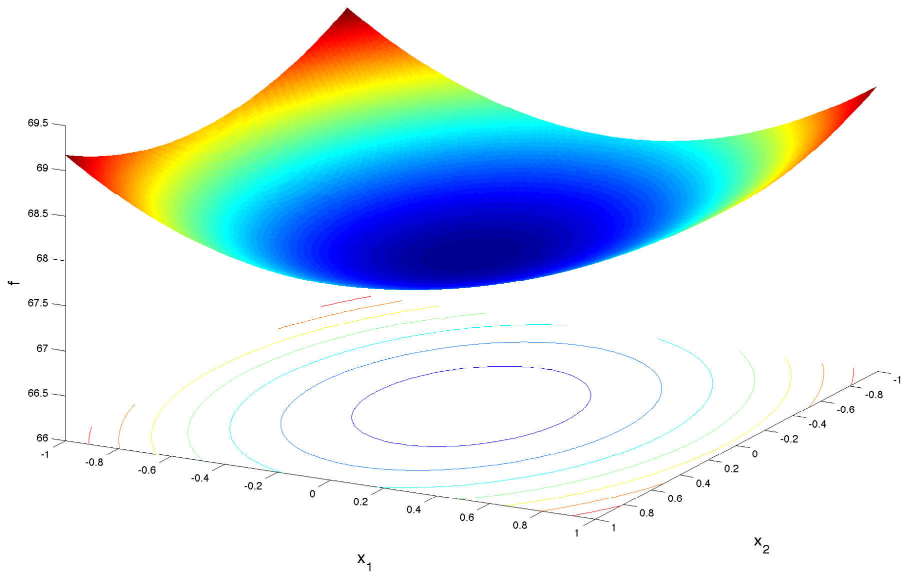
Sufficient conditions (for a strong local minimum):

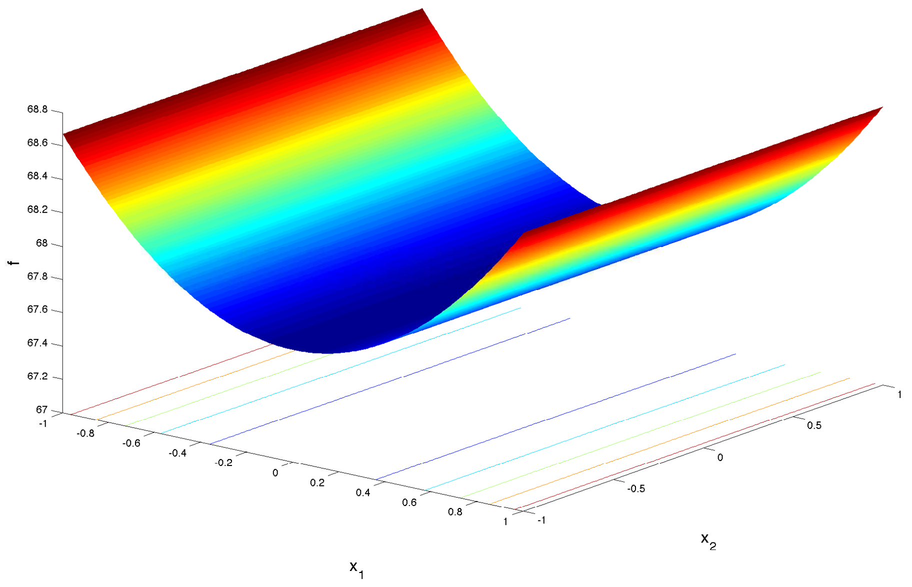
$$\|g(x^*)\| = 0 \quad \text{and} \quad H(x^*) \quad \text{is positive definite.} \tag{8}$$

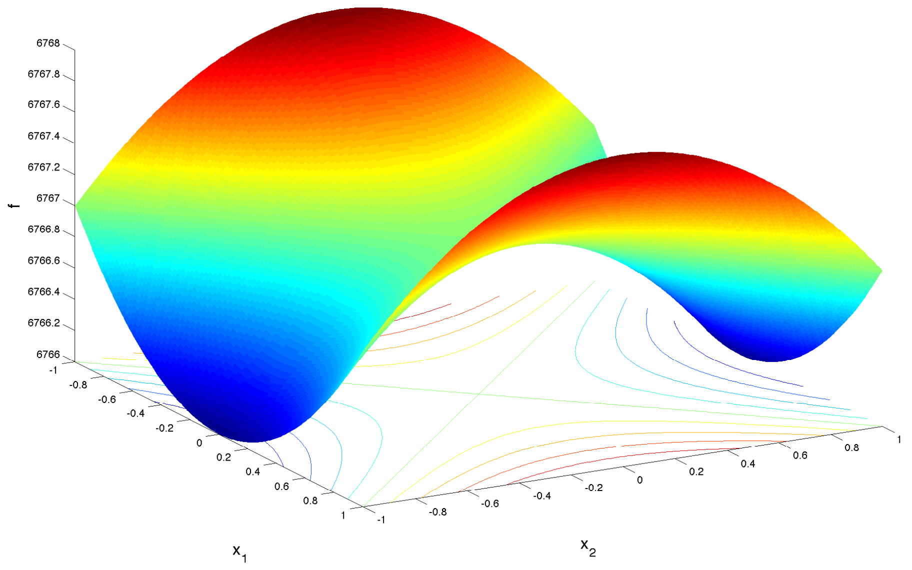
Some definitions from linear algebra that might be helpful:

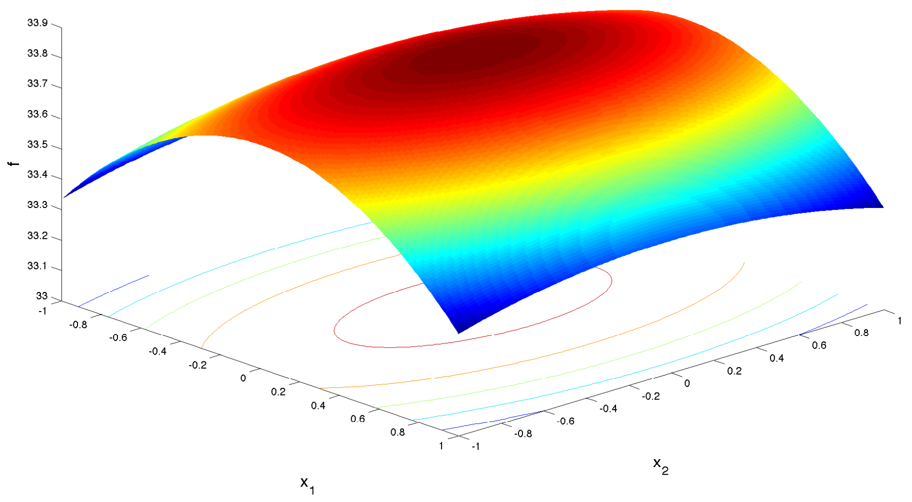
- The matrix $H \in \mathbb{R}^{n \times n}$ is *positive definite* if $p^T H p > 0$ for all nonzero vectors $p \in \mathbb{R}^n$ (If $H = H^T$ then all the eigenvalues of H are strictly positive)
- The matrix $H \in \mathbb{R}^{n \times n}$ is *positive semi-definite* if $p^T H p \geq 0$ for all vectors $p \in \mathbb{R}^n$ (If $H = H^T$ then the eigenvalues of H are positive or zero)
- The matrix $H \in \mathbb{R}^{n \times n}$ is *indefinite* if there exists $p, q \in \mathbb{R}^n$ such that $p^T H p > 0$ and $q^T H q < 0$.
(If $H = H^T$ then H has eigenvalues of mixed sign.)
- The matrix $H \in \mathbb{R}^{n \times n}$ is *negative definite* if $p^T H p < 0$ for all nonzero vectors $p \in \mathbb{R}^n$ (If $H = H^T$ then all the eigenvalues of H are strictly negative)

2-D Critical Point Examples









Example 1.1: Critical Points of a Function
Consider the function:

$$f(x) = 1.5x_1^2 + x_2^2 - 2x_1x_2 + 2x_1^3 + 0.5x_1^4$$

Find all stationary points of f and classify them.

Solve $\nabla f(x) = 0$, get three solutions:

$$\begin{aligned} (0, 0) &\quad \text{local minimum} \\ 1/2(-3-\sqrt{7}, -3-\sqrt{7}) &\quad \text{global minimum} \\ 1/2(-3+\sqrt{7}, -3+\sqrt{7}) &\quad \text{saddle point} \end{aligned}$$

To establish the type of point, we have to determine if the Hessian is positive definite and compare the values of the function at the points.

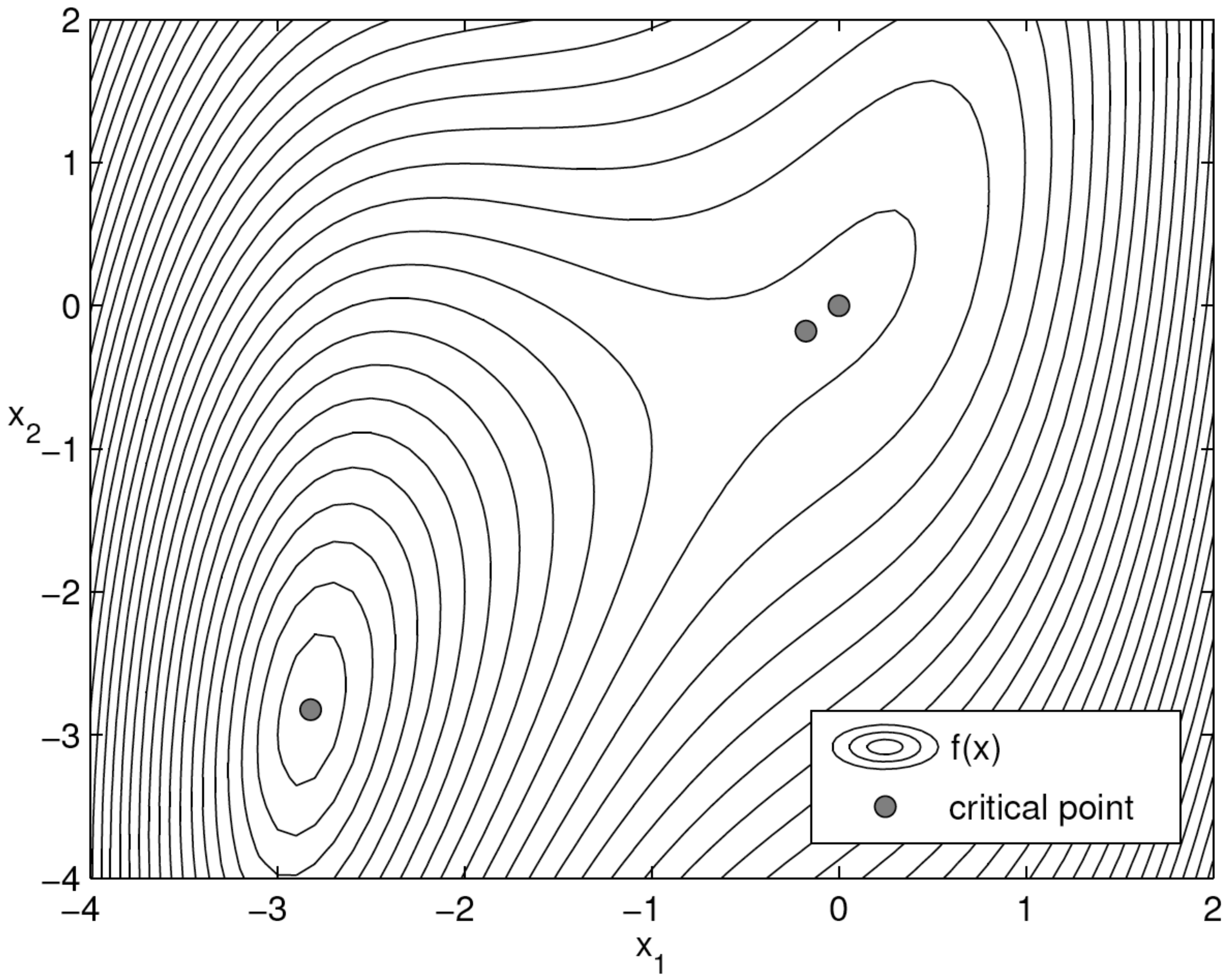


Figure 1: Critical points of $f(x) = 1.5x_1^2 + x_2^2 - 2x_1x_2 + 2x_1^3 + 0.5x_1^4$

1.3 Steepest Descent Method

The steepest descent method uses the gradient vector at each point as the search direction for each iteration. As mentioned previously, the gradient vector is orthogonal to the plane tangent to the isosurfaces of the function.

The gradient vector at a point, $g(x_k)$, is also the direction of maximum rate of change (maximum increase) of the function at that point. This rate of change is given by the norm, $\|g(x_k)\|$.

Steepest descent algorithm:

1. Select starting point x_0, and convergence parameters $\varepsilon_g, \varepsilon_a$ and ε_r.
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop. Otherwise, compute the normalized search direction to $p_k = -g(x_k)/\|g(x_k)\|$.
3. Perform line search to find step length α_k in the direction of p_k.
4. Update the current point, $x_{k+1} = x_k + \alpha p_k$.
5. Evaluate $f(x_{k+1})$. If the condition $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ is satisfied for two successive iterations then stop. Otherwise, set $k = k + 1$, $x_{k+1} = x_k + 1$ and return to step 2.

Here, $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_a + \varepsilon_r |F(x_k)|$ is a check for the successive reductions of f. ε_a is the absolute tolerance on the change in function value (usually small $\approx 10^{-6}$) and ε_r is the relative tolerance (usually set to 0.01).

If we use an exact line search, the steepest descent direction at each iteration is orthogonal to the previous one, i.e.,

$$\begin{aligned} \frac{df(x_{k+1})}{d\alpha} &= 0 \Rightarrow \frac{\partial f(x_{k+1})}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial \alpha} = 0 \Rightarrow \nabla^T f(x_{k+1}) p_k = 0 \Rightarrow \qquad (9) \\ -g^T(x_{k+1}) g(x_k) &= 0 \qquad (10) \end{aligned}$$

Therefore the method "zigzags" in the design space and is rather inefficient. Although a substantial decrease may be observed in the first few iterations, the method is usually very slow after that. In particular, while the algorithm is guaranteed to converge, it may take an infinite number of iterations. The rate of convergence is linear.

For steepest descent and other gradient methods that do not produce well-scaled search directions, we need to use other information to guess a step length.

One strategy is to assume that the first-order change in x_k will be the same as the one obtained in the previous step. i.e, that $\bar{\alpha} g_k^T p_k = \alpha_{k-1} g_{k-1}^T p_{k-1}$ and therefore:

$$\bar{\alpha} = \alpha_{k-1} \frac{g_{k-1}^T p_{k-1}}{g_k^T p_k}. \tag{11}$$

Example 1.2: Steepest Descent Applied to $f(x_1, x_2) = 1 - e^{-(10x_1^2 + x_2^2)}$

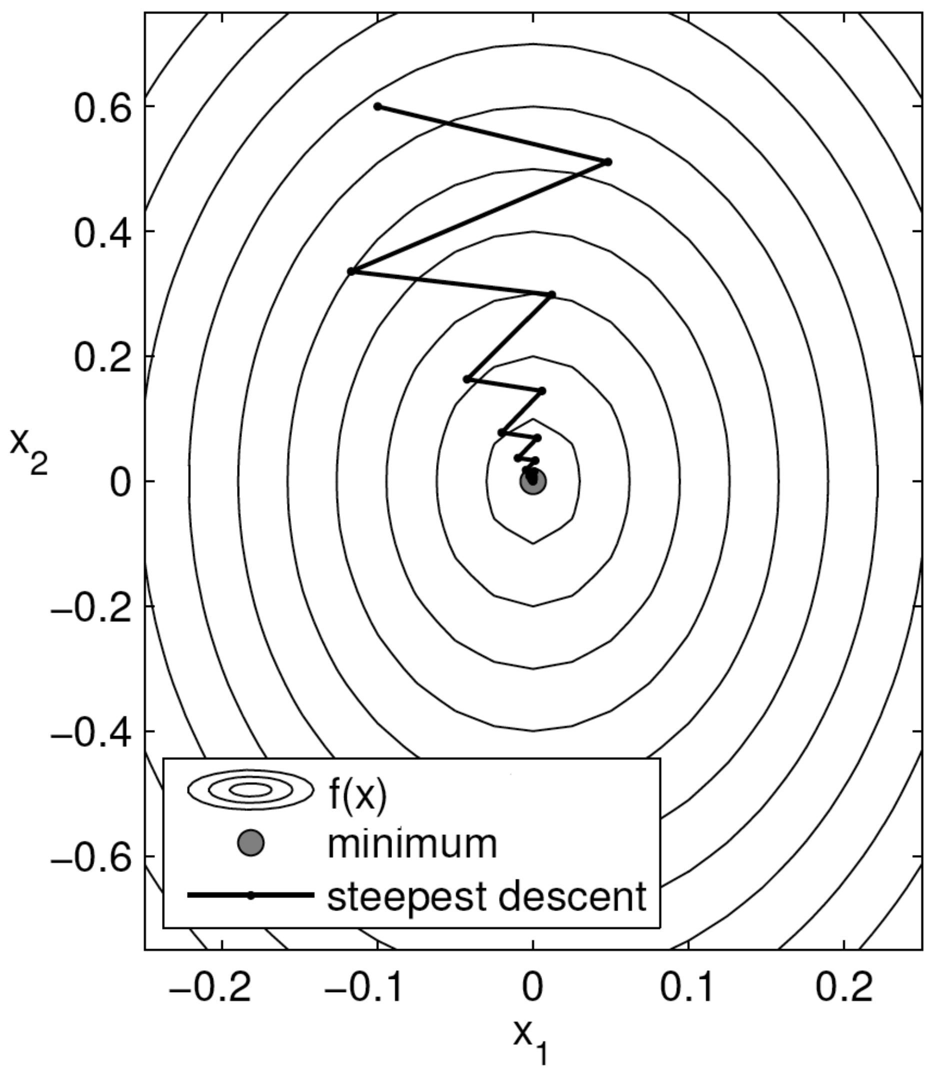


Figure 2: Solution path of the steepest descent method

1.4 Conjugate Gradient Method

A small modification to the steepest descent method takes into account the history of the gradients to move more directly towards the optimum.

Suppose we want to minimize a convex quadratic function

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x \tag{12}$$

where A is an $n \times n$ matrix that is symmetric and positive definite. Differentiating this with respect to x we obtain,

$$\nabla\phi(x) = Ax - b \equiv r(x). \tag{13}$$

Minimizing the quadratic is thus equivalent to solving the linear system,

$$Ax = b. \tag{14}$$

The conjugate gradient method is an iterative method for solving linear systems of equations such as this one.

A set of nonzero vectors $\{p_0, p_1, \ldots, p_{n-1}\}$ is *conjugate* with respect to A if

$$p_i^T Ap_j = 0, \quad \text{for all} \quad i \neq j. \tag{15}$$

Suppose that we start from a point x_0 and a set of directions $\{p_0, p_1, \ldots, p_{n-1}\}$ to generate a sequence $\{x_k\}$ where

$$x_{k+1} = x_k + \alpha_k p_k \tag{16}$$

where α_k is the minimizer of ϕ along $x_k + \alpha p_k$, given by

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \tag{17}$$

We will see that for any x_0 the sequence $\{x_k\}$ generated by the conjugate direction algorithm converges to the solution of the linear system in at most n steps.

Since conjugate directions are linearly independent, they span n-space. Therefore,

$$x^* - x_0 = \sigma_0 p_0 + \cdots + \sigma_{n-1} p_{n-1} \tag{18}$$

Premultiplying by $p_k^T A$ and using the conjugacy property we obtain

$$\sigma_k = \frac{p_k^T A (x^* - x_0)}{p_k^T A p_k} \tag{19}$$

Now we show that σ's are really the α's defined in (17).

A given iteration point can be written as a function of the search directions and step sizes so far,

$$x_k = x_0 + \alpha_0 p_0 + \cdots + \alpha_{k-1} p_{k-1}. \tag{20}$$

Premultiplying by $p_k^T A$ and using the conjugacy property we obtain

$$p_k^T A(x_k - x_0) = 0. \tag{21}$$

Therefore

$$p_k^T A(x^* - x_0) = P_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k \tag{22}$$

Dividing the leftmost term and the rightmost term by $p_k^T A p_k$ we get the equality,

$$\frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} = -\frac{p_k^T r_k}{p_k^T A p_k} \Rightarrow \sigma_k = \alpha_k. \tag{23}$$

There is a simple interpretation of the conjugate directions. If A is diagonal, the isosurfaces are ellipsoids with axes aligned with coordinate directions. We could then find minimum by performing univariate minimization along each coordinate direction in turn and this would result in convergence to the minimum in n iterations.

When A a positive-definite matrix that is not diagonal, its contours are still elliptical, but they are not aligned with the coordinate axes. Minimization along coordinate directions no longer leads to solution in n iterations (or even a finite n).

If we transform the variables using

$$\hat{x} = S^{-1}x, \tag{24}$$

where $S = \begin{bmatrix} p_0 & p_1 & \cdots & p_{n-1} \end{bmatrix}$, a matrix whose columns are the set of conjugate directions with respect to A. The quadratic now becomes

$$\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T \left(S^T A S\right) \hat{x} - \left(S^T b\right)^T \hat{x} \tag{25}$$

By conjugacy, $\left(S^T A S\right)$ is diagonal so we can do a sequence of n line minimizations along the coordinate directions of $\hat{x}$. Each univariate minimization determines a component of x^* correctly.

Nonlinear Conjugate Gradient Algorithm

(Also known as Fletcher–Reeves method)

1. Select starting point x_0, and convergence parameters $\varepsilon_g, \varepsilon_a$ and ε_r.
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop.
3. If $k = 0$, then go to step 5.
4. Compute the new conjugate gradient direction $p_k = -g_k + \beta_k p_{k-1}$, where $\beta = \left(\frac{\|g_k\|}{\|g_{k-1}\|}\right)^2 = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$.
5. Perform line search to find step length α_k in the direction of p_k.
6. Update the current point, $x_{k+1} = x_k + \alpha p_k$.
7. Evaluate $f(x_{k+1})$. If the condition $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_a + \varepsilon_r |f(x_k)|$ is satisfied for two successive iterations then stop. Otherwise, set $k = k + 1$ and return to step 3.

Usually, a *restart* is performed every n iterations for computational stability, i.e. we start with a steepest descent direction.

The conjugate gradient method does not produce well-scaled search directions, so we can use same strategy to choose the initial step size as for steepest descent.

Several variants of the Fletcher–Reeves CG method have been proposed. Most of these variants differ in their definition of β_k. For example, Dai and Yuan (SIAM Jounal on Optimization, 10, 1999, pp. 177–182) propose

$$\beta_k = \frac{\|g_k\|^2}{(g_k - g_{k-1})^T p_{k-1}}.$$

Example 1.3: Conjugate Gradient Applied to $f(x_1, x_2) = 1 - e^{-(10x_1^2 + x_2^2)}$

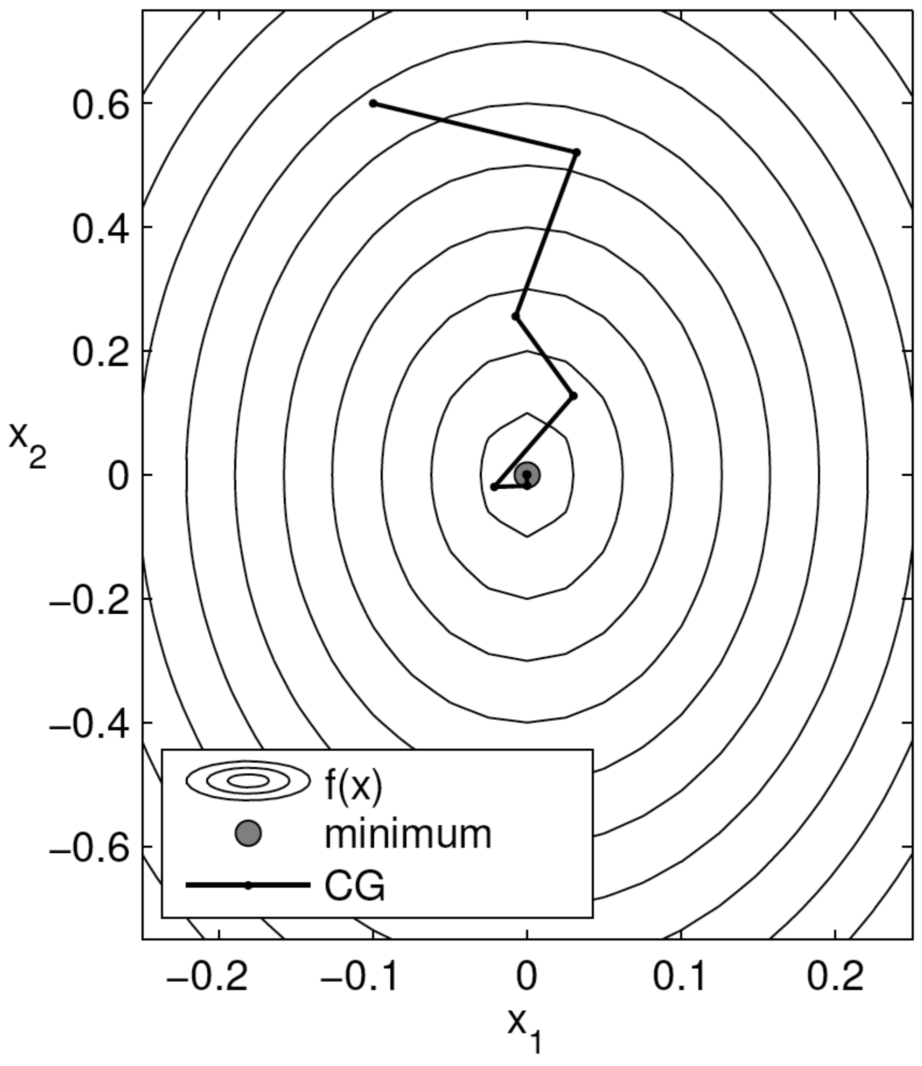


Figure 3: Solution path of the nonlinear conjugate gradient method

1.5 Newton's Method

The steepest descent and conjugate gradient methods only use first order information (the first derivative term in the Taylor series) to obtain a local model of the function.

Newton methods use a second-order Taylor series expansion of the function about the current design point, i.e. a quadratic model

$$f(x_k + s_k) \approx f_k + g_k^T s_k + \frac{1}{2} s_k^T H_k s_k, \tag{26}$$

where s_k is the step to the minimum. Differentiating this with respect to s_k and setting it to zero, we can obtain the step for that minimizes this quadratic,

$$H_k s_k = -g_k. \tag{27}$$

This is a linear system which yields the *Newton step*, s_k, as a solution.

If H_k is positive definite, only one iteration is required for a quadratic function, from any starting point. For a general nonlinear function, Newton's method converges quadratically if x_0 is sufficiently close to x^* and the Hessian is positive definite at x^*.

As in the single variable case, difficulties and even failure may occur when the quadratic model is a poor approximation of f far from the current point. If H_k is not positive definite, the quadratic model might not have a minimum or even a stationary point. For some nonlinear functions, the Newton step might be such that $f(x_k + s_k) > f(x_k)$ and the method is not guaranteed to converge.

Another disadvantage of Newton's method is the need to compute not only the gradient, but also the Hessian, which contains $n(n+1)/2$ second order derivatives.

Modified Newton's Method

A small modification to Newton's method is to perform a line search along the Newton direction, rather than accepting the step size that would minimize the quadratic model.

1. Select starting point x_0, and convergence parameter ε_g.
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop. Otherwise, continue.
3. Compute $H(x_k) \equiv \nabla^2 f(x_k)$ and the search direction, $p_k = -H^{-1} g_k$.
4. Perform line search to find step length α_k in the direction of p_k (start with $\alpha_k = 1$).
5. Update the current point, $x_{k+1} = x_k + \alpha_k p_k$ and return to step 2.

Although this modification increases the probability that $f(x_k + p_k) < f(x_k)$, it still vulnerable to the problem of having an Hessian that is not positive definite and has all the other disadvantages of the pure Newton's method.

We could also introduce a modification to use a symmetric positive definite matrix instead of the real Hessian to ensure descent:

$$B_k = H_k + \gamma I,$$

where γ is chosen such that all the eigenvalues are greater than a tolerance $\delta > 0$ (if δ is too close to zero, B_k might be ill-conditioned, which can lead to round of errors in the solution of the linear system).

When using Newton or quasi-Newton methods, the starting step length $\bar{\alpha}$ is usually set to 1, since Newton's method already provides a good guess for the step size.

The step size reduction ratio (ρ in the backtracking line search) sometimes varies during the optimization process and is such that $0 < \rho < 1$. In practice ρ is not set to be too close to 0 or 1.

Example 1.4: Modified Newton's Method Applied to $f(x_1, x_2) = 1 - e^{-(10x_1^2 + x_2^2)}$

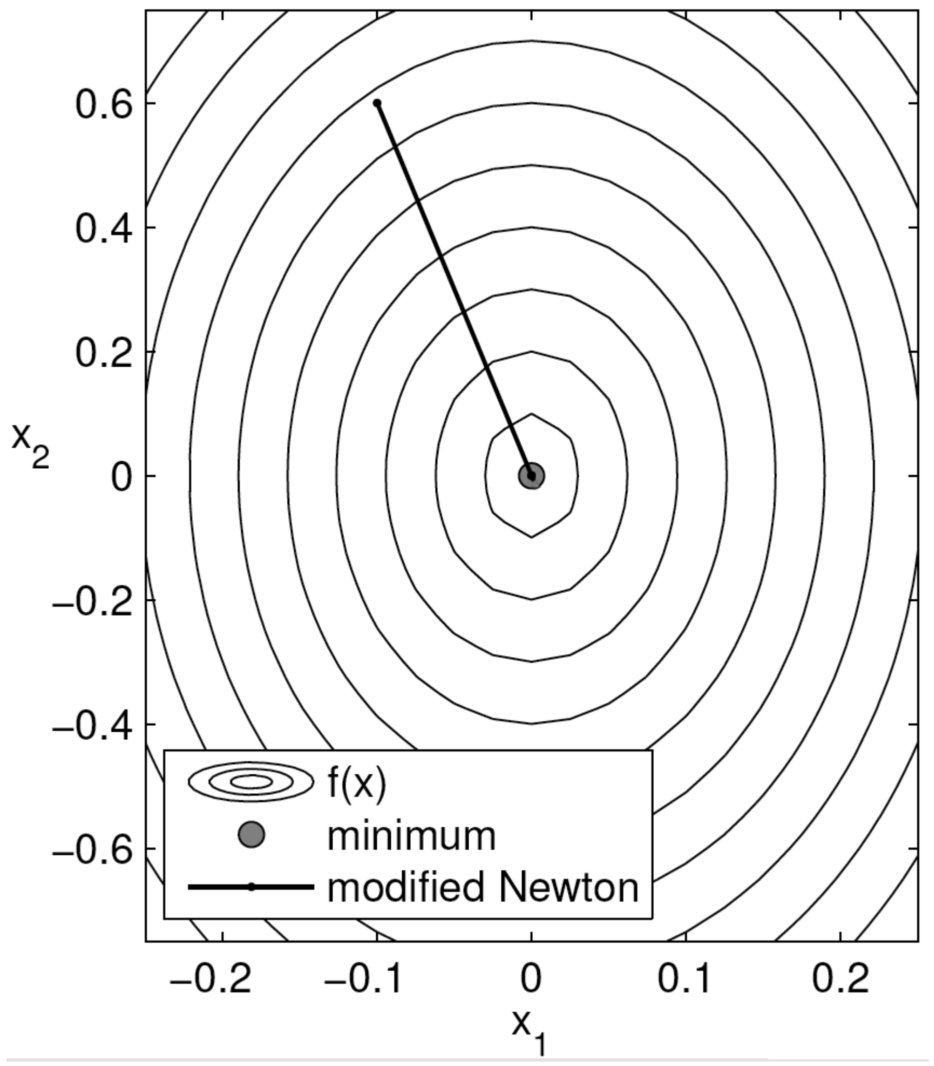


Figure 4: Solution path of the modified Newton's method

1.6 Quasi-Newton Methods

This class of methods uses first order information only, but build second order information — an approximate Hessian — based on the sequence of function values and gradients from previous iterations. Most of these methods also force the Hessian to be symmetric and positive definite, which can greatly improve their convergence properties.

When using quasi-Newton methods, we usually start with the Hessian initialized to the identity matrix and then update it at each iteration. Since what we actually need is the inverse of the Hessian, we will work with $V_k \approx H_k^{-1}$. The update at each iteration is written as $\hat{V}_k$ and is added to the current one,

$$V_{k+1} = V_k + \hat{V}_k. \tag{28}$$

Let s_k be the step taken from x_k, and consider the Taylor-series expansion of the gradient function about x_k,

$$g(x_k + s_k) = g_k + H_k s_k + \cdots \tag{29}$$

Truncating this series and setting the variation of the gradient to $y_k = g(x_k + s_k) - g_k$ yields

$$H_k s_k = y_k. \tag{30}$$

Then, the new approximate inverse of the Hessian, V_{k+1} must satisfy the *quasi-Newton condition*,

$$V_{k+1} y_k = s_k. \tag{31}$$

1.6.1 Davidon–Fletcher–Powell (DFP) Method

One of the first quasi-Newton methods was devised by Davidon (1959) and modified by Fletcher and Powell (1963).

Suppose we model the objective function as a quadratic at the current iterate x_k:

$$\phi_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p, \tag{32}$$

where B_k is an $n \times n$ symmetric positive definite matrix that is *updated* every iteration.

The minimizer p_k for this convex quadratic model can be written as,

$$p_k = -B_k^{-1} g_k. \tag{33}$$

This solution is used to compute the search direction to obtain the new iterate

$$x_{k+1} = x_k + \alpha_k p_k \tag{34}$$

where α_k is obtained using a line search.

This is the same procedure as the Newton method, except that we use an approximate Hessian B_k instead of the true Hessian.

Instead of computing B_k "from scratch" at every iteration, a quasi-Newton method updates it in a way that accounts for the curvature measured during the most recent step. We want to build the quadratic model,

$$\phi_{k+1}(p) = f_{k+1} + g_{k+1}^T p + \frac{1}{2} p^T B_{k+1} p. \tag{35}$$

What requirements should we impose on B_{k+1}, based on the new information from the last step? One would be that the gradient of ϕ_{k+1} should match the gradient of the actual function f at the latest two points x_k and x_{k+1}. This condition is satisfied by definition at x_{k+1}, since $\nabla \phi_{k+1}(0) = g_k$. To match the gradient at the previous point,

$$\nabla \phi_{k+1} \left(-\alpha_k p_k \right) = g_{k+1} - \alpha_k B_{k+1} p_k = g_k. \tag{36}$$

Rearranging,

$$B_{k+1} \alpha_k p_k = g_{k+1} - g_k. \tag{37}$$

Setting the variation of the gradient to $y_k = g_{k+1} - g_k$ yields

$$B_{k+1} s_k = y_k, \tag{38}$$

which is called the *secant condition*.

We have $n(n+1)/2$ unknowns and only n equations, so this system has an infinite number of solutions for B_{k+1}. To determine the solution uniquely, we impose a condition that among all the matrices that satisfy the secant condition, selects the B_{k+1} that is "closest" to the previous Hessian approximation B_k, i.e. we solve,

$$\begin{aligned} \text{minimize} \quad & \|B - B_k\| && (39) \\ \text{with respect to} \quad & B && (40) \\ \text{subject to} \quad & B = B^T, \quad B s_k = y_k. && (41) \end{aligned}$$

Using different matrix norms result in different quasi-Newton methods. One norm that makes it easy to solve this problem and possesses good numerical properties is the weighted Frobenius norm

$$\|A\|_W = \|W^{1/2} A W^{1/2}\|_F, \tag{42}$$

where the norm is defined as $\|C\|_F = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^2$. The weights W are chosen to satisfy certain favorable conditions.

For further details on the derivation of the quasi-Newton methods, see Dennis and Moré's review (SIAM Review, Vol. 19, No. 1, 1977).

Using this norm and weights, the unique solution of the norm minimization problem (39) is,

$$B_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) B_k \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) + \frac{y_k y_k^T}{y_k^T s_k}, \tag{43}$$

which is the *DFP updating formula* originally proposed by Davidon.

Using the inverse of B_k is usually more useful, since the search direction can then be obtained by matrix multiplication. Defining,

$$V_k = B_k^{-1}. \tag{44}$$

The DFP update for the inverse of the Hessian approximation can be shown to be

$$V_{k+1} = V_k - \frac{V_k y_k y_k^T V_k}{y_k^T V_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \tag{45}$$

The DFP Algorithm

1. Select starting point x_0, and convergence parameter ε_g. Set $k = 0$ and $V_0 = I$.
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop. Otherwise, continue.
3. Compute the search direction, $p_k = -V_k g_k$.
4. Perform line search to find step length α_k in the direction of p_k (start with $\alpha_k = 1$).
5. Update the current point, $x_{k+1} = x_k + \alpha_k p_k$, set $s_k = \alpha_k p_k$, and compute the change in the gradient, $y_k = g_{k+1} - g_k$.
6. Update V_{k+1} by computing

$$A_k = \frac{V_k y_k y_k^T V_k}{y_k^T V_k y_k} \qquad B_k = \frac{s_k s_k^T}{s_k^T y_k}$$

$$V_{k+1} = V_k - A_k + B_k$$

7. Set $k = k + 1$ and return to step 2.

1.6.2 Broyden–Fletcher–Goldfarb–Shanno (BFGS) Method

The DFP update was soon superseded by the BFGS formula, which is generally considered to be the most effective quasi-Newton updates. Instead of solving the norm minimization problem (39) of for B we now solve the same problem for its inverse V using equivalent conditions and and the solution is given by,

$$V_{k+1} = \left[I - \frac{s_k y_k^T}{s_k^T y_k}\right] V_k \left[I - \frac{y_k s_k^T}{s_k^T y_k}\right] + \frac{s_k s_k^T}{s_k^T y_k}. \tag{46}$$

The relative performance between the DFP and BFGS methods is problem dependent.

Example 1.5: BFGS Applied to $f(x_1, x_2) = 1 - e^{-(10x_1^2+x_2^2)}$

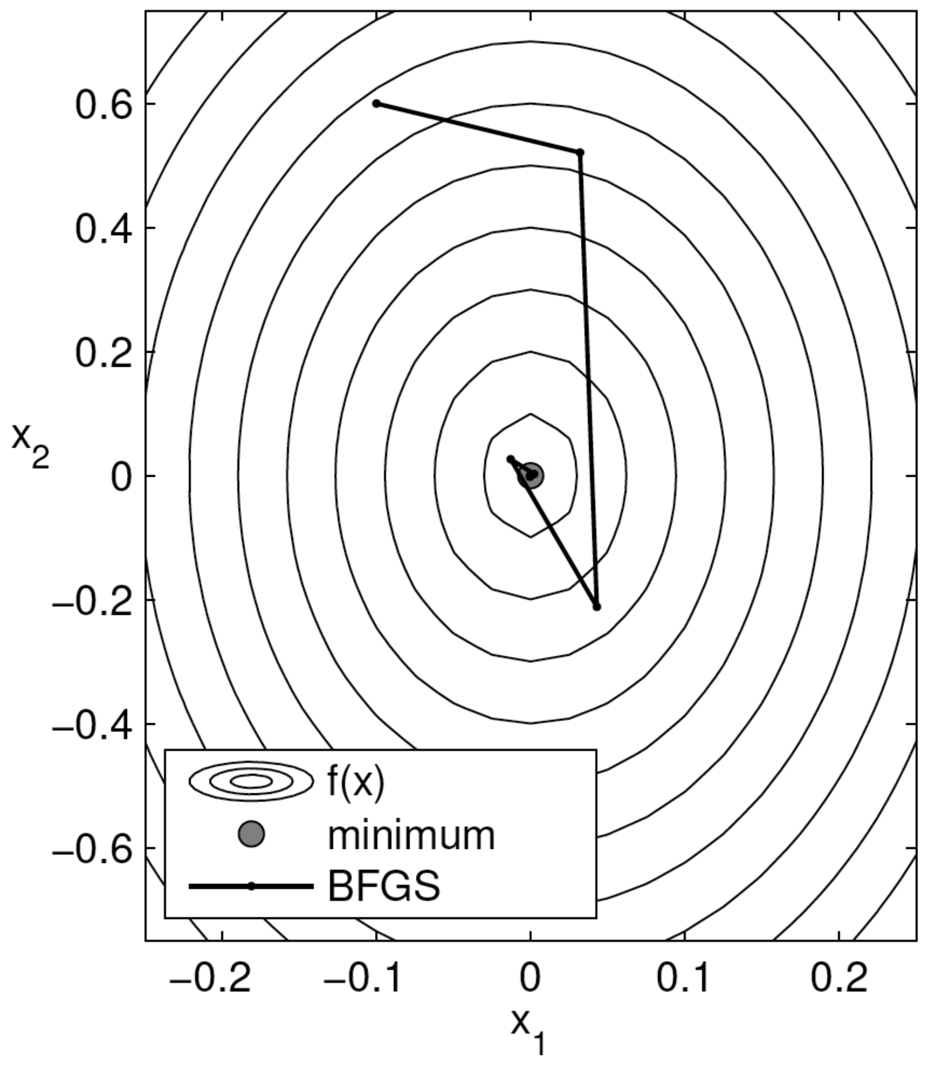


Figure 5: Solution path of the BFGS method

1.6.3 Symmetric Rank-1 Update Method (SR1)

If we drop the requirement that the approximate Hessian (or its inverse) be positive definite, we can derive a simple rank-1 update formula for B_k that maintains the symmetry of the matrix and satisfies the secant equation. Such a formula is given by the symmetric rank-1 update (SR1) method (we use the Hessian update here, and not the inverse V_k):

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

With this formula, we must consider the cases when the denominator vanishes, and add the necessary safe-guards:

1. if $y_k = B_k s_k$ then the only update that satisfies the secant equation is $B_{k+1} = B_k$ (i.e. do not change the matrix).
2. if $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$ then there is no symmetric rank-1 update that satisfies the secant equation.

To avoid the second case, we update the matrix only if the following condition is met:

$$|y_k^T (s_k - B_k y_k)| \geq r \|s_k\| \, \|y_k - B_k s_k\|,$$

where $r \in (0,1)$ is a small number (e.g. $r = 10^{-8}$). Hence, if this condition is not met, we use $B_{k+1} = B_k$.

Why would we be interested in a Hessian approximation that is potentially indefinite? In practice, the matrices produced by SR1 have been found to approximation the true Hessian matrix very well (often better than BFGS). This may be useful in trust-region methods (see next section) or constrained optimization problems; in the latter case the Hessian of the Lagrangian is often indefinte, even at the minimizer.

To use the SR1 method in practice, it may be necessary to add a diagonal matrix γI to B_k when calulating the serach direction, as was done in modified Newton's method. In addition, a simple back-tracking line search can be used, since the Wolfe conditions are not required as part of the update (unlike BFGS).

Example 1.6: Minimization of the Rosenbrock Function

Minimize Rosenbrock's function,

$$f(x) = 100 \left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 ,$$

starting from $x_0 = (-1.2, 1.0)^T$.

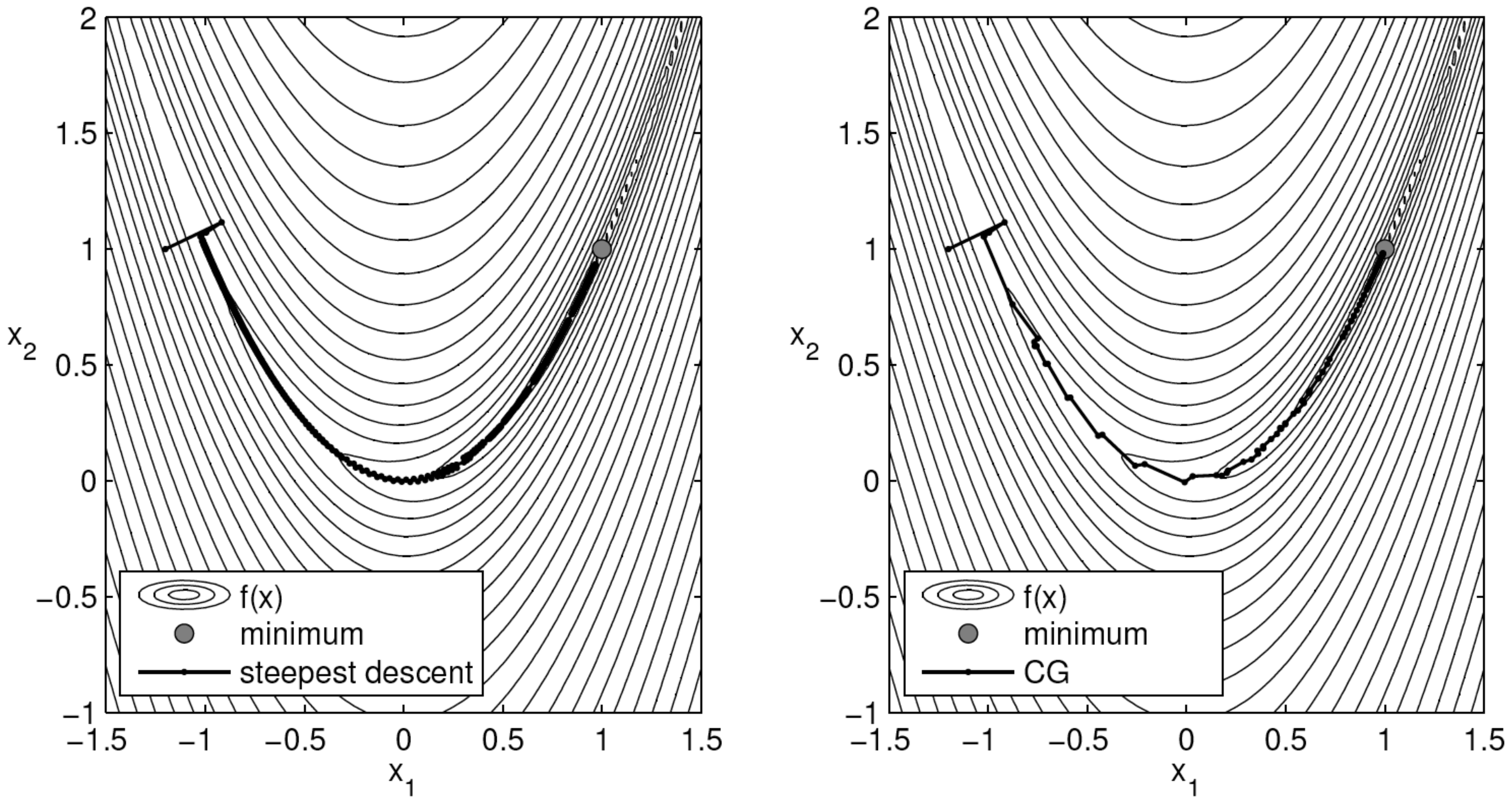


Figure 6: Solution path of the steepest descent and conjugate gradient methods

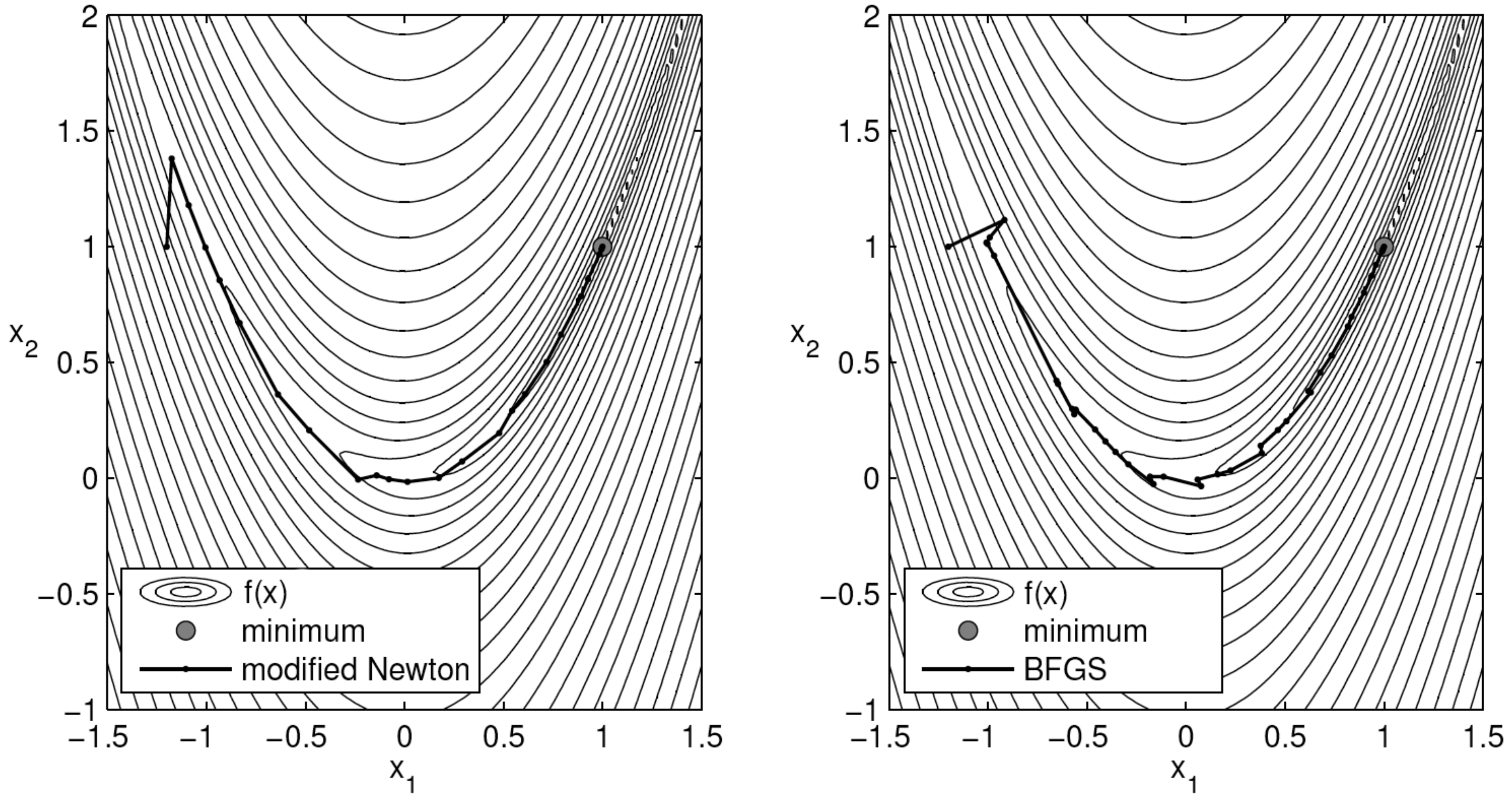


Figure 7: Solution path of the modified Newton and BFGS methods

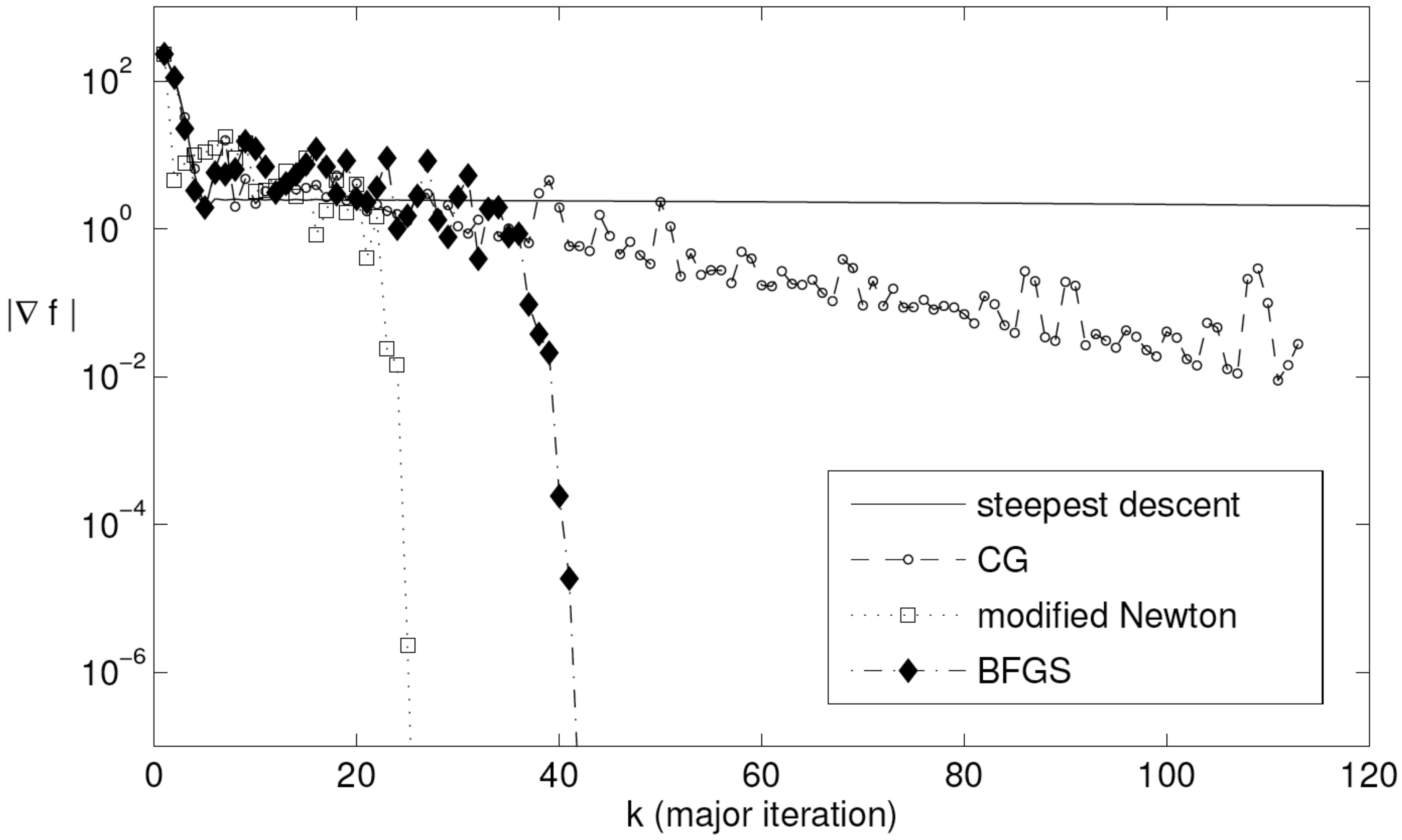


Figure 8: Comparison of convergence rates for the Rosenbrock function

1.7 Trust Region Methods

Trust region, or "restricted-step" methods are a different approach to resolving the weaknesses of the pure form of Newton's method, arising from an Hessian that is not positive definite or a highly nonlinear function.

One way to interpret these problems is to say that they arise from the fact that we are stepping outside a the region for which the quadratic approximation is reasonable. Thus we can overcome this difficulties by minimizing the quadratic function within a region around x_k within which we *trust* the quadratic model.

Trust Region Algorithm

1. Select starting point x_0, a convergence parameter ε_g and the initial size of the trust region, h_0.
2. Compute $g(x_k) \equiv \nabla f(x_k)$. If $\|g(x_k)\| \leq \varepsilon_g$ then stop. Otherwise, continue.
3. Compute $H(x_k) \equiv \nabla^2 f(x_k)$ and solve the *quadratic subproblem*

$$\text{minimize} \quad q(s_k) = f(x_k) + g(x_k)^T s_k + \tfrac{1}{2} s_k^T H(x_k) s_k \tag{47}$$

$$\text{w.r.t.} \quad s_k \tag{48}$$

$$\text{s.t.} \quad -h_k \leq s_k \leq h_k, \quad i = 1, \ldots, n \tag{49}$$

4. Evaluate $f(x_k + s_k)$ and compute the ratio that measures the accuracy of the quadratic model,
$$r_k = \frac{\Delta f}{\Delta q} = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - q(s_k)}.$$
5. Compute the size for the new trust region as follows:
$$\begin{aligned} h_{k+1} &= \frac{\|s_k\|}{4} \quad \text{if} \quad r_k < 0.25, && (50) \\ h_{k+1} &= 2h_k \quad \text{if} \quad r_k > 0.75 \quad \text{and} \quad h_k = \|s_k\|, && (51) \\ h_{k+1} &= h_k \quad \text{otherwise.} && (52) \end{aligned}$$
6. Determine the new point:
$$\begin{aligned} x_{k+1} &= x_k \quad \text{if} \quad r_k \leq 0 && (53) \\ x_{k+1} &= x_k + s_k \quad \text{otherwise.} && (54) \end{aligned}$$
7. Set $k = k + 1$ and return to 2.

The initial value of h is usually 1. The same stopping criteria used in other gradient-based methods are applicable.

References

[1] A. D. Belegundu and T. R. Chandrupatla. *Optimization Concepts and Applications in Engineering*, chapter 3. Prentice Hall, 1999.

[2] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[3] C. Onwubiko. *Introduction to Engineering Design Optimization*, chapter 4. Prentice Hall, 2000.