The BFGS Method

Choose an initial guess $\boldsymbol{x}_0$ and approximate $\boldsymbol{B}_0$ (e.g., $\boldsymbol{B}_0 = \boldsymbol{I}$)
while (criterion)
  Calculate $\boldsymbol{s}_k$ by solving $\boldsymbol{B}_k \boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$
  Find an optimal step size β_k by a line search method
  Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \beta_k \boldsymbol{s}_k$
  Calculate $\boldsymbol{u}_k, \boldsymbol{v}_k$ and update $\boldsymbol{B}_{k+1}$ using (5.3) and (5.4)
end for while
Set $k = k + 1$

Figure 5.1: The pseudocode of the BFGS method.

where $\boldsymbol{B}_k$ is the approximation to the Hessian matrix at kth iteration. Then, a line search is performed to find the optimal stepsize β_k so that the new trial solution is determined by

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_k + \beta_k \boldsymbol{s}_k. \tag{5.2}$$

Introducing two new variables

$$\boldsymbol{u}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \beta_k \boldsymbol{s}_k, \qquad \boldsymbol{v}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k), \tag{5.3}$$

we can update the new estimate as

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{\boldsymbol{v}_k^T \boldsymbol{u}_k} - \frac{(\boldsymbol{B}_k \boldsymbol{u}_k)(\boldsymbol{B}_k \boldsymbol{u}_k)^T}{\boldsymbol{u}_k^T \boldsymbol{B}_k \boldsymbol{u}_k}. \tag{5.4}$$

The procedure of the BFGS method is outlined in Figure 5.1.

5.2 NELDER-MEAD METHOD

5.2.1 A Simplex

In the n-dimensional space, a simplex, which is a generalization of a triangle on a plane, is a convex hull with $n+1$ distinct points. For simplicity, a simplex in the n-dimension space is referred to as n-simplex. Therefore, 1-simplex is a line segment, 2-simplex is a triangle, a 3-simplex is a tetrahedron (see Figure 5.2), and so on.

5.2.2 Nelder-Mead Downhill Simplex

The Nelder-Mead method is a downhill simplex algorithm for unconstrained optimization without using derivatives, and it was first developed by J. A. Nelder and R. Mead in 1965. This is one of the most widely used methods

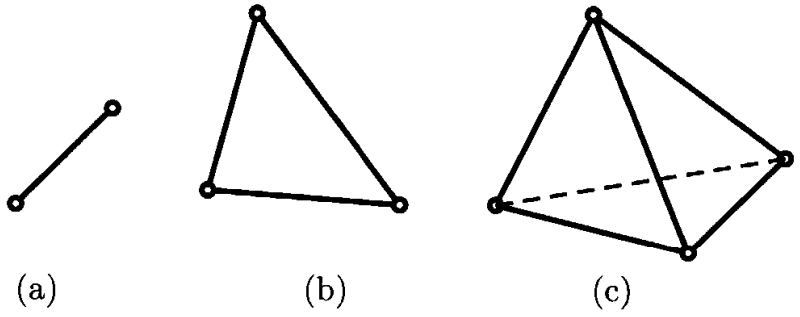

Figure 5.2: The concept of a simplex: (a) 1-simplex, (b) 2-simplex, and (c) 3-simplex.

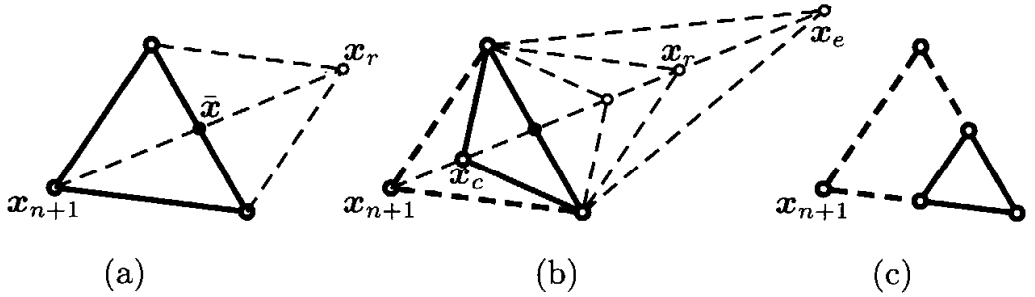

Figure 5.3: Simplex manipulations: (a) reflection with fixed volume (area), (b) expansion or contraction along the line of reflection, (c) reduction.

since its computational effort is relatively small and is something to get a quick grasp of the optimization problem. The basic idea of this method is to use the flexibility of the constructed simplex via amoeba-style manipulations by reflection, expansion, contraction and reduction (see Figure 5.3). In some books such as the best known *Numerical Recipes*, it is also called the 'Amoeba Algorithm'. It is worth pointing out that this downhill simplex method has nothing to do with the simplex method for linear programming.

There are a few variants of the algorithm that use slightly different ways of constructing initial simplex and convergence criteria. However, the fundamental procedure is the same (see Figure 5.4).

The first step is to construct an initial n-simplex with $n+1$ vertices and to evaluate the objective function at the vertices. Then, by ranking the objective values and re-ordering the vertices, we have an ordered set so that

$$f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2) \leq ... \leq f(\boldsymbol{x}_{n+1}), \tag{5.5}$$

at $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{n+1}$, respectively. As the downhill simplex method is for minimization, we use the convention that $\boldsymbol{x}_{n+1}$ is the worse point (solution), and $\boldsymbol{x}_1$ is the best solution. Then, at each iteration, similar ranking manipulations are carried out.

Then, we have to calculate the centroid $\boldsymbol{x}$ of simplex excluding the worst vertex $\boldsymbol{x}_{n+1}$

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i. \tag{5.6}$$

Using the centroid as the basis point, we try to find the reflection of the worse point $\boldsymbol{x}_{n+1}$ by

$$\boldsymbol{x}_r = \bar{\boldsymbol{x}} + \alpha(\bar{x} - \boldsymbol{x}_{n+1}), \qquad (\alpha > 0), \tag{5.7}$$

though the typical value of $\alpha = 1$ is often used.

Whether the new trial solution is accepted or not and how to update the new vertex, depends on the objective function at $\boldsymbol{x}_r$. There are three possibilities:

- If $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_r) < f(\boldsymbol{x}_n)$, then replace the worst vertex $\boldsymbol{x}_{n+1}$ by $\boldsymbol{x}_r$, that is $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_r$.

- If $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_1)$ which means the objective improves, we then seek a more bold move to see if we can improve the objective even further by moving/expanding the vertex further along the line of reflection to a new trial solution

  $$\boldsymbol{x}_e = \boldsymbol{x}_r + \beta(\boldsymbol{x}_r - \bar{\boldsymbol{x}}), \tag{5.8}$$

  where $\beta = 2$. Now we have to test if $f(\boldsymbol{x}_e)$ improves even better. If $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$, we accept it and update $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_e$; otherwise, we can use the result of the reflection. That is $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_r$.

- If there is no improvement or $f(\boldsymbol{x}_r) > f(\boldsymbol{x}_n)$, we have to reduce the size of the simplex whiling maintaining the best sides. This is contraction

  $$\boldsymbol{x}_c = \boldsymbol{x}_{n+1} + \gamma(\bar{\boldsymbol{x}} - \boldsymbol{x}_{n+1}), \tag{5.9}$$

  where $0 < \gamma < 1$, though $\gamma = 1/2$ is usually used. If $f(\boldsymbol{x}_c) < f(\boldsymbol{x}_{n+1})$ is true, we then update $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_c$.

If all the above steps fail, we should reduce the size of the simplex towards the best vertex $\boldsymbol{x}_1$. This is the reduction step

$$\boldsymbol{x}_i = \boldsymbol{x}_1 + \delta(\boldsymbol{x}_i - \boldsymbol{x}_1), \qquad (i = 2, 3, ..., n+1). \tag{5.10}$$

Then, we go to the first step of the iteration process and start over again.

5.3 TRUST-REGION METHOD

The fundamental ideas of the trust-region have developed over many years with many seminal papers by a dozen of pioneers. A good history review of the trust-region methods can be found in the book by Conn, Gould and Toint (2000). Briefly speaking, the first important work was due to Levenberg

Nelder-Mead (Downhill Simplex) Method

Initialize a simplex with $n+1$ vertices in n dimension.
while (stop criterion is not true)
(1) Re-order the points such that $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2) \leq ... \leq f(\boldsymbol{x}_{n+1})$
with $\boldsymbol{x}_1$ being the best and $\boldsymbol{x}_{n+1}$ being the worse (highest value)
(2) Find the centroid $\bar{\boldsymbol{x}}$ using $\bar{\boldsymbol{x}} = \sum_{i=1}^{n} \boldsymbol{x}_i/n$ excluding $\boldsymbol{x}_{n+1}$.
(3) Generate a trial point via the reflection of the worse vertex
$\boldsymbol{x}_r = \bar{\boldsymbol{x}} + \alpha(\bar{\boldsymbol{x}} - \boldsymbol{x}_{n+1})$ where $\alpha > 0$ (typically $\alpha = 1$)
(a) **if** $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_r) < f(\boldsymbol{x}_n)$, $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_r$; **end**
(b) **if** $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_1)$,
Expand in the direction of reflection $\boldsymbol{x}_e = \boldsymbol{x}_r + \beta(\boldsymbol{x}_r - \bar{\boldsymbol{x}})$
if $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$, $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_e$; **else** $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_r$; **end**
end
(c) **if** $f(\boldsymbol{x}_r) > f(\boldsymbol{x}_n)$, Contract by $\boldsymbol{x}_c = \boldsymbol{x}_{n+1} + \gamma(\bar{\boldsymbol{x}} - \boldsymbol{x}_{n+1})$;
if $f(\boldsymbol{x}_c) < f(\boldsymbol{x}_{n+1})$, $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_c$;
else Reduction by $\boldsymbol{x}_i = \boldsymbol{x}_1 + \delta(\boldsymbol{x}_i - \boldsymbol{x}_1), (i = 2, ..., n+1)$; **end**
end
end

Figure 5.4: Pseudocode of Nelder-Mead's downhill simplex method.

in 1944, which proposed the usage of addition of a multiple of the identity matrix to the Hessian matrix as a damping measure to stabilize the solution procedure for nonlinear least-squares problems. Later, Marquardt in 1963 independently pointed out the link between such damping in the Hessian and the reduction of the step size in a restricted region. Slightly later in 1966, Goldfelt, Quandt and Trotter essentially set the stage for trust-region methods by introducing an explicit updating formula for the maximum step size. Then, in 1970, Powell proved the global convergence for the trust-region method, though it is believed that the term 'trust region' was coined by Dennis in 1978, as earlier literature used various terminologies such as region of validity, confidence region, and restricted step method.

In the trust-region algorithm, a fundamental step is to approximate the nonlinear objective function by using truncated Taylor expansions, often the quadratic form

$$\phi_k(\boldsymbol{x}) \approx f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T \boldsymbol{u} + \frac{1}{2}\boldsymbol{u}^T \boldsymbol{H}_k \boldsymbol{u}, \tag{5.11}$$

in a so-called trust region Ω_k defined by

$$\Omega_k = \{\boldsymbol{x} \in \Re^d \Big| \|\boldsymbol{\Gamma}(\boldsymbol{x} - \boldsymbol{x}_k)\| \leq \Delta_k\}, \tag{5.12}$$

where Δ_k is the trust-region radius. Here $\boldsymbol{H}_k$ is the local Hessian matrix. $\boldsymbol{\Gamma}$ is a diagonal scaling matrix that is related to the scalings of the optimization