Quasi-Newton methods require only the gradient (like steepest descent) of the objective to be computed at each iterate.

By successive measurements of the gradient, Quasi-Newton methods build a quadratic model of the objective function which is sufficiently good that superlinear convergence is achieved.

Quasi-Newton methods are much faster than steepest descent (and coordinate descent) methods.

Since second derivatives (the Hessian) are not required, quasi-Newton methods are sometimes more efficient (as measured by total work / “wall-clock computational time”) than Newton methods, especially when Hessian evaluation is slow/expensive.

The BFGS method is named for its discoverers: BROYDEN-FLETCHER-GOLDFARB-SHANNO, and is the most popular quasi-Newton method.

We first derive the DFP method (a close relative; named after DAVIDON-FLETCHER-Powell) and then the BFGS method; and look at some properties and practical implementation details.

The derivation starts with the quadratic model

\[ m_k(\bar{p}) = f(\bar{x}_k) + \nabla f(\bar{x}_k)^T \bar{p} + \frac{1}{2} \bar{p}^T B_k \bar{p} \]

at the current iterate \( \bar{x}_k \). \( B_k \) is a symmetric positive definite matrix (model Hessian) that will be updated in every iteration.
The BFGS Method: Introduction

Given this convex quadratic model, we can write down the minimizer $\mathbf{p}_k$ explicitly as

$$
\mathbf{p}_k = -B_k^{-1}\nabla f(\bar{x}_k).
$$

We can compute the search direction $\mathbf{p}_k$ using e.g. the Cholesky factorization, or a (P)CG-iteration; once we have $\mathbf{p}_k$ we find the new iterate:

$$
\bar{x}_{k+1} = \bar{x}_k + \alpha_k \mathbf{p}_k,
$$

where we require that the step length $\alpha_k$ satisfies e.g. the Wolfe conditions:

$$
\begin{align*}
  f(\bar{x}_k + \alpha \mathbf{p}_k) &\leq f(\bar{x}_k) + \alpha c_1 \mathbf{p}_k^T \nabla f(\bar{x}) , & c_1 \in (0,1) \\
  \mathbf{p}_k^T \nabla f(\bar{x}_k + \alpha \mathbf{p}_k) &\geq c_2 \mathbf{p}_k^T \nabla f(\bar{x}_k), & c_2 \in (1,1).
\end{align*}
$$

The BFGS Method: Conditions on $B_{k+1}$

We impose two conditions on the new model $m_{k+1}(\mathbf{p})$:

$$
[1,2] \quad m_{k+1}(\mathbf{p}) \text{ must match the gradient of the objective function in } \bar{x}_k \text{ and } \bar{x}_{k+1}.
$$

The second condition is satisfied by construction, since

$$
\nabla m_{k+1}(\mathbf{0}) = \nabla f(\bar{x}_{k+1}).
$$

The first condition gives us

$$
\nabla m_{k+1}(-\alpha_k \mathbf{p}_k) = \nabla f(\bar{x}_{k+1}) - \alpha_k B_{k+1} \mathbf{p}_k = \nabla f(\bar{x}_k).
$$

With a little bit of re-arrangement we get

$$
\alpha_k B_{k+1} \mathbf{p}_k = \nabla f(\bar{x}_{k+1}) - \nabla f(\bar{x}_k).
$$

We clean up the notation by introducing $(\bar{x}_k, \bar{y}_k)$:

$$
\begin{align*}
\bar{s}_k &= \bar{x}_{k+1} - \bar{x}_k \quad \Rightarrow \quad \alpha_k \mathbf{p}_k \\
\bar{y}_k &= \nabla f(\bar{x}_{k+1}) - \nabla f(\bar{x}_k).
\end{align*}
$$

We can now express the condition on $B_{k+1}$ in terms of $\bar{s}_k$ and $\bar{y}_k$:

**Secant Equation**

$$
B_{k+1} \bar{s}_k = \bar{y}_k.
$$

By pre-multiplying the secant equation by $\bar{s}_k^T$ we get the

**Curvature Condition**

$$
\bar{s}_k^T B_{k+1} \bar{s}_k = \bar{s}_k^T \bar{y}_k \Rightarrow \bar{s}_k^T \bar{y}_k > 0.
$$
The BFGS Method: Conditions on $B_{k+1}$

If we impose the Wolfe, or strong Wolfe condition on the line search procedure, the curvature condition will always hold, since

$$\nabla f(\bar{x}_{k+1})^T \bar{s}_k \geq c_2 \nabla f(\bar{x}_k)^T \bar{s}_k,$$

by the (curvature) Wolfe condition, and therefore

$$\bar{y}_k^T \bar{s}_k \geq (c_2 - 1) \alpha_k \nabla f(\bar{x}_k)^T \bar{p}_k,$$

where the right-hand-side is positive since $c_2 < 1$ and $\bar{p}_k$ is a descent direction.

When the curvature condition is satisfied, the secant equation always has at least one solution $B_{k+1}$.

---

The BFGS Method: More Conditions on $B_{k+1}$

It turns out that there are infinitely many symmetric positive definite matrices $B_{k+1}$ which satisfy the secant equation.

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>Conditions Imposed</th>
</tr>
</thead>
</table>
| $n(n+1)/2$ — Symmetric | $n$ — The Secant Equation
|                    | $n$ — Principal minors positive (PD) |

To determine $B_{k+1}$ uniquely we must impose additional conditions — we will select the $B_{k+1}$ that is closest to $B_k$ in some sense:

**Matrix-Minimization-Problem**

$$B_{k+1} = \arg \min_B \| B - B_k \|_{\text{some-norm}}$$

subject to

$$B = B^T, \quad Bs_k = \bar{y}_k.$$

---

Square Roots of SPD Matrices

- A positive semi-definite matrix, $M$ has a unique positive semi-definite square root, $R = M^{1/2}$.

- When $M = XX^{-1} = Q\Lambda Q^T$, let $R = QSQ^T$, and

  $$R^2 = (QSQ^T)^2 = QSQ^TQSQ^T = QSSQ^T = QS^2Q^T = M,$$

  showing that

  $$S = \Lambda^{1/2}, \quad \text{and therefore} \quad R = QA^{1/2}Q^T$$

- $\exists$ other approaches.
The DFP Method

With this weighting matrix and norm, the unique solution of the MMP is

\[ B_{k+1} = \left( I - \gamma_k \bar{y}_k \bar{s}_k^T \right) B_k \left( I - \gamma_k \bar{s}_k \bar{y}_k^T \right) + \gamma_k \bar{y}_k \bar{y}_k^T, \quad \gamma_k = \frac{1}{\bar{y}_k^T \bar{s}_k}. \]

Note that \( \gamma_k \) is a scalar, and \( \bar{y}_k \bar{s}_k^T, \bar{s}_k \bar{y}_k^T, \) and \( \bar{y}_k \bar{y}_k^T \) are rank-one matrices.

This is the original Davidon-Fletcher-Powell (DFP) method suggested by W.C. Davidon in 1959.

The original paper describing this revolutionary idea — the first quasi-Newton method — was not accepted for publication. It later appeared in 1991 in the first issue of the SIAM Journal on Optimization.

Fletcher and Powell demonstrated that this algorithm was much faster and more reliable than existing methods (at the time). This revolutionized the field of non-linear optimization.

The DFP method is quite effective, but once the quasi-Newton idea was accepted by the optimization community is was quickly superseded by the BFGS method.

The updated approximation must be symmetric positive definite, and must satisfy the secant equation in the form

\[ H_{k+1} \bar{y}_k = \bar{s}_k, \quad \text{compare:} \quad B_{k+1} \bar{s}_k = \bar{y}_k. \]

We get a slightly different matrix minimization problem...
The BFGS Matrix Minimization Problem

Matrix-Minimization-Problem (BFGS)

\[ H_{k+1} = \arg \min_{H} \|H - H_k\|_{\text{some-norm}} \]

subject to \( H = H^T, \ H \bar{y}_k = \bar{s}_k \)

If we again choose the weighted Frobenius norm (with the same weight), then we get the unique update

\[ H_{k+1} = \left( I - \rho_k \bar{s}_k \bar{y}_k^T \right) H_k \left( I - \rho_k \bar{y}_k \bar{s}_k^T \right) + \rho_k \bar{s}_k \bar{s}_k^T, \quad \rho_k = \frac{1}{\bar{y}_k^T \bar{s}_k}, \]

which translated back to the Hessian approximation yields

\[ B_{k+1} = B_k - \frac{B_k \bar{s}_k \bar{s}_k^T B_k + \bar{y}_k \bar{y}_k^T}{\bar{s}_k^T B_k \bar{s}_k + \bar{y}_k^T \bar{y}_k}. \]

The BFGS Method: Starting — \( H_0 = ??? \)

The initial value for the iteration can be selected in different ways

- A finite difference approximation at \( \bar{x}_0 \).
- \( H_0 = I \), the identity matrix.
- \( H_0 = \text{diag}(s_1, s_2, \ldots, s_n) \), where \( \bar{s} \) captures the scaling of the variables (if known).

Algorithm: The BFGS Method

Given starting point \( \bar{x}_0 \), convergence tolerance \( \epsilon > 0 \), and initial inverse Hessian approximation \( H_0 \):

\[ k = 0 \]

while ( \( \|\nabla f(\bar{x}_k)\| > \epsilon \) )

\[ \bar{p}_k = -H_k \nabla f(\bar{x}_k) \]

\[ \bar{x}_{k+1} = \text{linesearch}(\bar{p}_k, \ldots) \]

\[ \bar{s}_k = \bar{x}_{k+1} - \bar{x}_k \]

\[ \bar{y}_k = \nabla f(\bar{x}_{k+1}) - \nabla f(\bar{x}_k) \]

\[ \rho_k = \frac{1}{\bar{y}_k^T \bar{s}_k} \]

\[ H_{k+1} = \left( I - \rho_k \bar{s}_k \bar{y}_k^T \right) H_k \left( I - \rho_k \bar{y}_k \bar{s}_k^T \right) + \rho_k \bar{s}_k \bar{s}_k^T \]

\[ k = k + 1 \]

end-while
The BFGS Method: Summary

The cost per iteration is

- \( O(n^2) \) arithmetic operations
- function evaluation
- gradient evaluation

The convergence rate is

- Super-linear

Newton’s method converges quadratically, but the cost per iteration is higher — it requires the solution of a linear system. In addition Newton’s method requires the calculation of second derivatives whereas the BFGS method does not.

The BFGS Method: Stability and Self-Correction 1 of 2

It has been shown that the BFGS method has self-correcting properties. — If \( H_k \) incorrectly estimates the curvature of the objective function, and if this estimate slows down the iteration, then the Hessian approximation will tend to correct itself within a few steps.

If at some point \( \rho_k = 1 / \bar{y}_k^T \bar{s}_k \) becomes large, i.e. \( \bar{y}_k^T \bar{s}_k \sim 0 \), then from the update formula

\[
H_{k+1} = \left( I - \rho_k \bar{s}_k \bar{y}_k^T \right) H_k \left( I - \rho_k \bar{y}_k \bar{s}_k^T \right) + \rho_k \bar{s}_k \bar{s}_k^T
\]

we see that \( H_{k+1} \) becomes large.

If for this, or some other, reason \( H_k \) becomes a poor approximation of \( [\nabla^2 f(\bar{x}_k)]^{-1} \) for some \( k \), is there any hope of correcting it?

The BFGS Method: Stability and Self-Correction 2 of 2

The self-correcting properties stand and fall with the quality of the line search! — The Wolfe conditions ensure that the model captures appropriate curvature information.

The DFP method is less effective at self-correcting bad Hessian approximations.

Practical Implementation Details:

- The linesearch should always test \( \alpha = 1 \) first, because this step length will eventually be accepted, thus creating super-linear convergence.
- The linesearch can be somewhat “sloppy”: \( c_1 = 10^{-4} \) and \( c_2 = 0.9 \) are commonly used values in the Wolfe conditions.
- The initial matrix \( H_0 \) should not be too large, if \( H_0 = \beta I \), then the first step is \( \bar{p}_0 = -\beta \nabla f(\bar{x}_0) \) which may be too long if \( \beta \) is large, often \( H_0 \) is rescaled before the update \( H_1 \) is computed:

\[
H_0 \leftarrow \frac{\bar{y}_k^T \bar{s}_k}{\bar{y}_k^T \bar{y}_k} I.
\]

L-BFGS

Forming the \( n \times n \) dense matrix \( H_k \) can be quite expensive for large problems. L-BFGS stores a limited history of the BFGS update vectors \( \bar{s}_k \) and \( \bar{y}_k \) (which are size \( n \)), and use these to “implicitly” form the matrix operations.

In standard BFGS, the current \( H_k \) contains updates all the way back to initial step \( \{ \bar{s}_j, \bar{y}_j \}_{j=0}^{k-1} \), whereas L-BFGS only uses a limited number of “recent” updates; so that the action of \( \tilde{H}_k \) is formed by application of \( \{ \bar{s}_j, \bar{y}_j \}_{j=\max(0,k-m)}^{k-1} \).
Given a local initial positive definite model for the Hessian, $\tilde{H}_k$:

1. $\bar{v} = \nabla f(\bar{x}_k)$
2. $\alpha_j = \rho_j \bar{s}_j^T \bar{v}$, $\bar{v} = \bar{v} - \alpha_j \bar{y}_j$, $j = k - 1, \ldots, k - m$.
3. $\bar{w} = \tilde{H}_k \bar{v}$
4. $\beta_j = \rho_j \bar{y}_j^T \bar{w}$, $\bar{w} = \bar{w} + \bar{s}_j (\alpha_j - \beta_j)$, $j = k - m, \ldots, k - 1$
5. Now, use $\bar{p}_k = -\bar{w} \approx -H_k \nabla f(\bar{x}_k)$.

References: