SI152: Numerical Optimization

Lecture 12: Quasi-Newton Method

Hao Wang
Email: haw309@gmail.com

ShanghaiTech University

November 6, 2025

Outline

Outline

$$\min_x \quad f(x)$$

Methods for unconstrained optimization are based on quadratic subproblems:

$$m_k(d) := f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T H_k d.$$

- Steepest descent is cheap, since only gradients are required. It, however, can be exceedingly slow, even in the neighborhood of a solution point.
- Newton's method is fast, but expensive, since it requires Hessians and the solution of a linear system to compute the search direction.

Quasi-Newton is one way to obtain fast convergence without second derivatives.

- Rather than compute $\nabla^2 f(x_k)$, we compute an approximation H_k.
- General goal is to approximate second derivatives with only first derivatives.
- Often, this is best done by updating H_k from one iteration to the next.
- An important consideration is whether to require positive definite approximations. Generally we do, but especially in a trust region framework, this is not necessary.

- Suppose that at iteration k we have the model

$$m_k(d) := f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T H_k d$$

with which we compute the search direction d_k and steplength α_k.

- We aim to compute H_{k+1} so that we have the new model

$$m_{k+1}(d) := f(x_{k+1}) + \nabla f(x_{k+1})^T d + \tfrac{1}{2} d^T H_{k+1} d$$

- What local information do we automatically have available to choose H_{k+1}?

values $\{f(x_k), f(x_{k+1})\}$ and gradients $\{\nabla f(x_k), \nabla f(x_{k+1})\}$.

- (We could imagine collecting information from other nearby points, but to have a cheap iteration, we focus only on using information we have already.)

Suppose we choose $m_{k+1}(d)$ to satisfy the following conditions:

$$m_{k+1}(0) = f(x_{k+1}); \quad \nabla m_{k+1}(0) = \nabla f(x_{k+1});$$

$$\nabla m_{k+1}(-\alpha_k d_k) = \nabla f(x_k).$$

That is,

- function value should match at x_{k+1};
- gradient values should match at x_{k+1} and x_k . (Note that with respect to x_{k+1}, we get to x_k with the step $-\alpha_k d_k$.
- The last is the only condition that depends on H_{k+1}. Rearranging

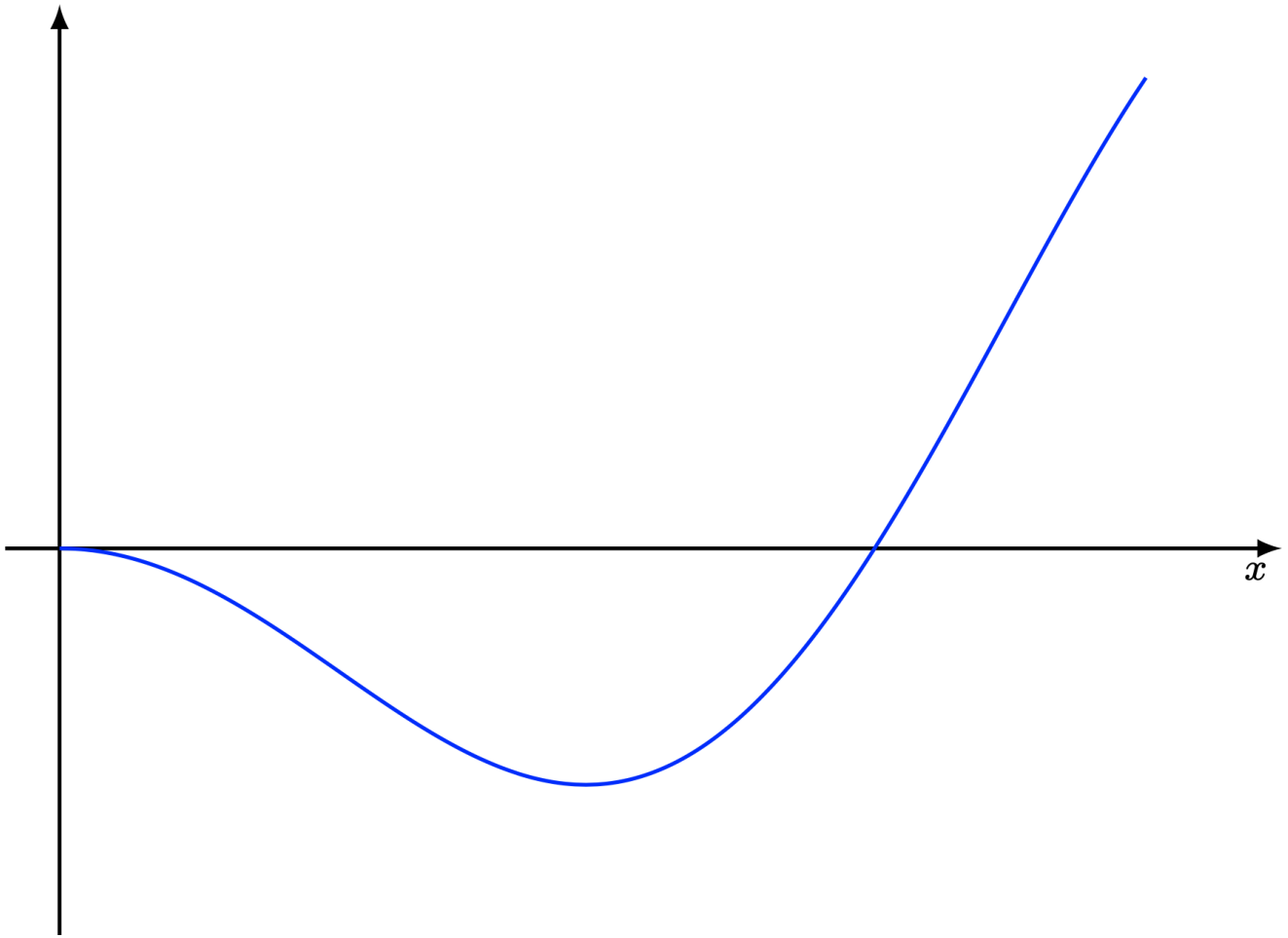
  $$\nabla f(x_k) = \nabla m_{k+1}(-\alpha_k d_k) = \nabla f(x_{k+1}) + H_{k+1}(-\alpha_k d_k),$$

  we obtain where

  $$H_{k+1} s_k = y_k \quad \text{(the "secant equation")},$$

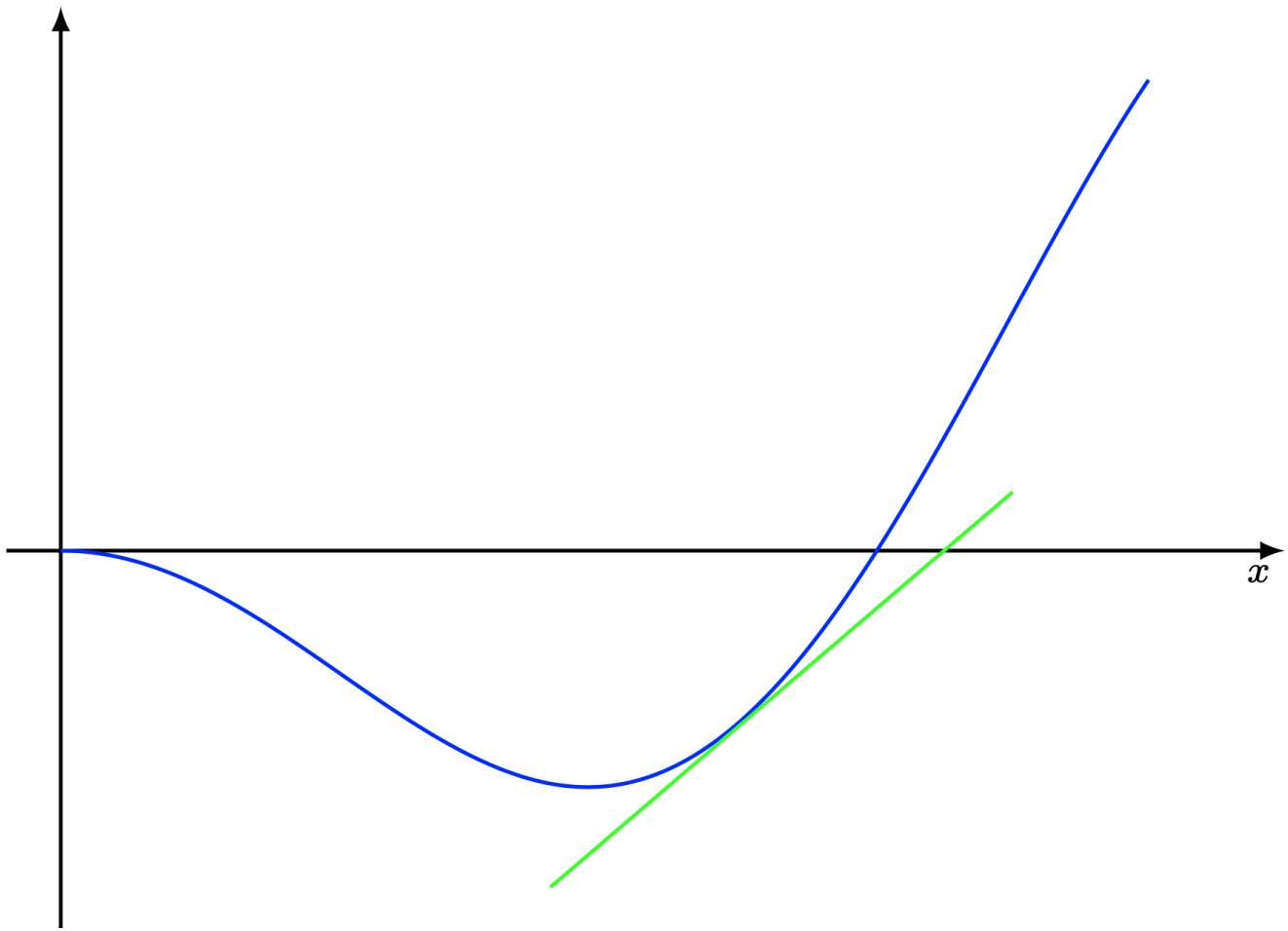
  $$s_k = x_{k+1} - x_k = \alpha_k d_k \text{ and } y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

Consider the following function:



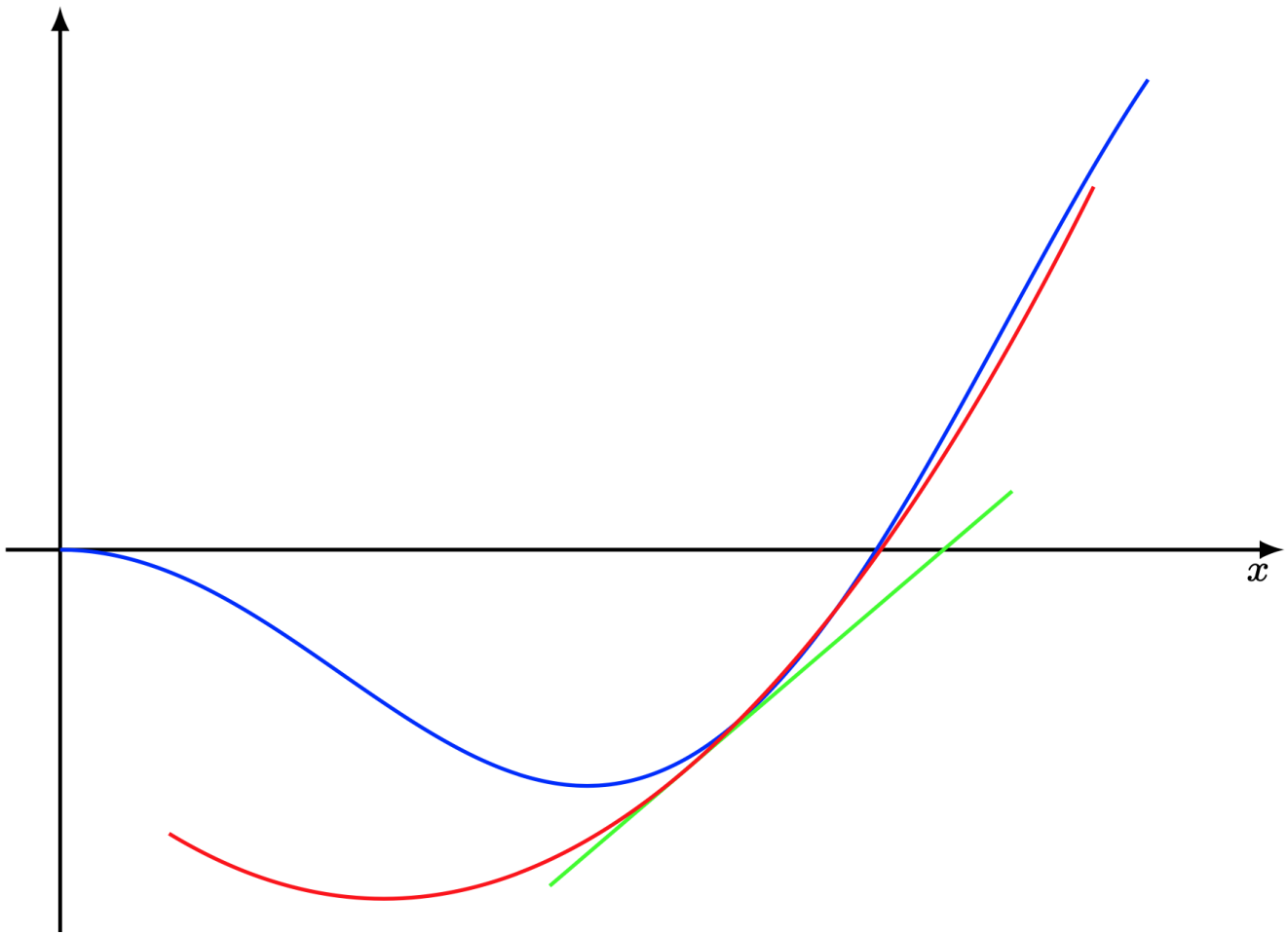
Example: Linear model at x_k

A linear model at x_k has the form $f(x_k) + \nabla f(x_k)^T d$:



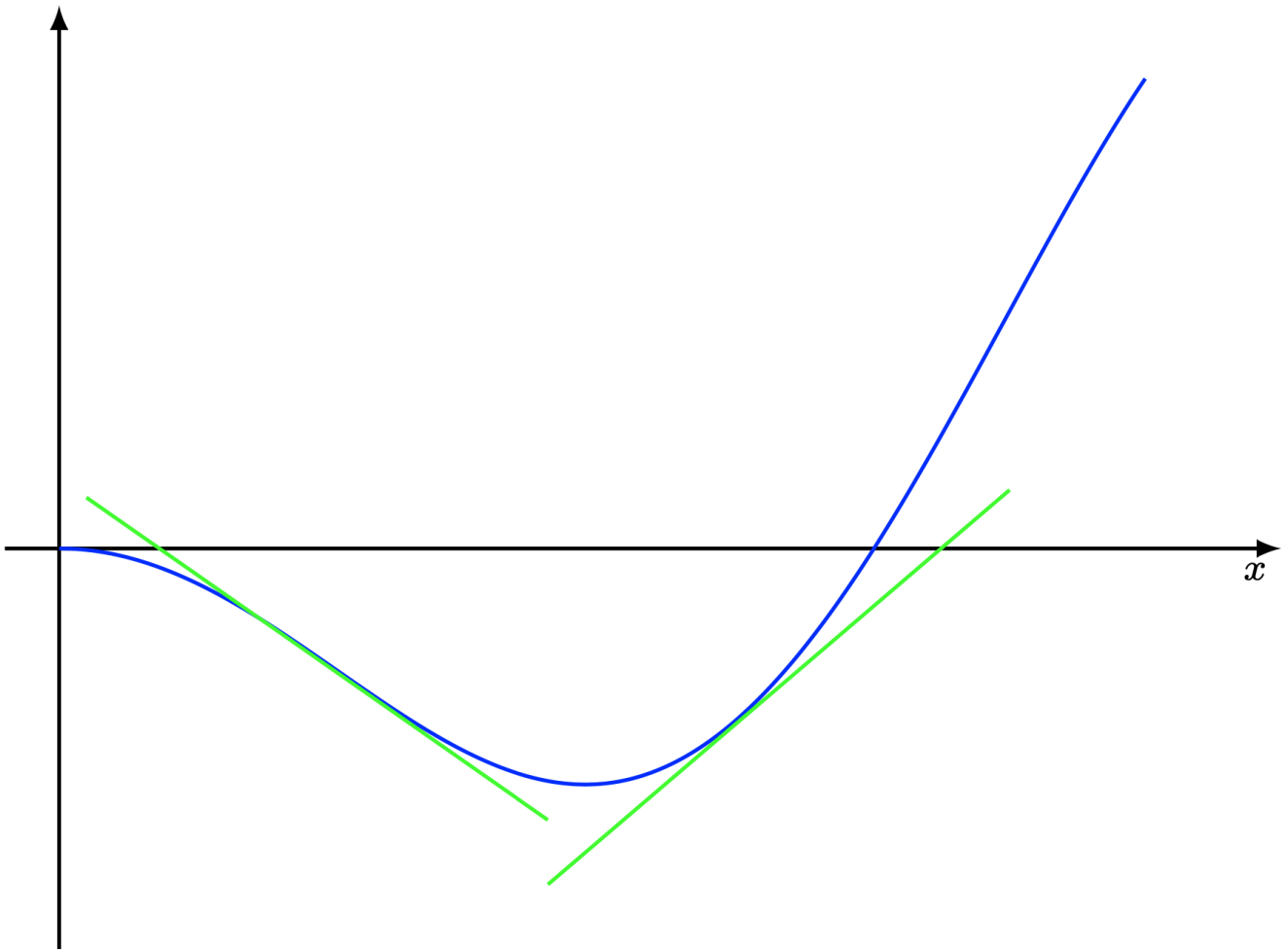
Example: Quadratic model at x_k

An approximation of the Hessian may give $m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d$:



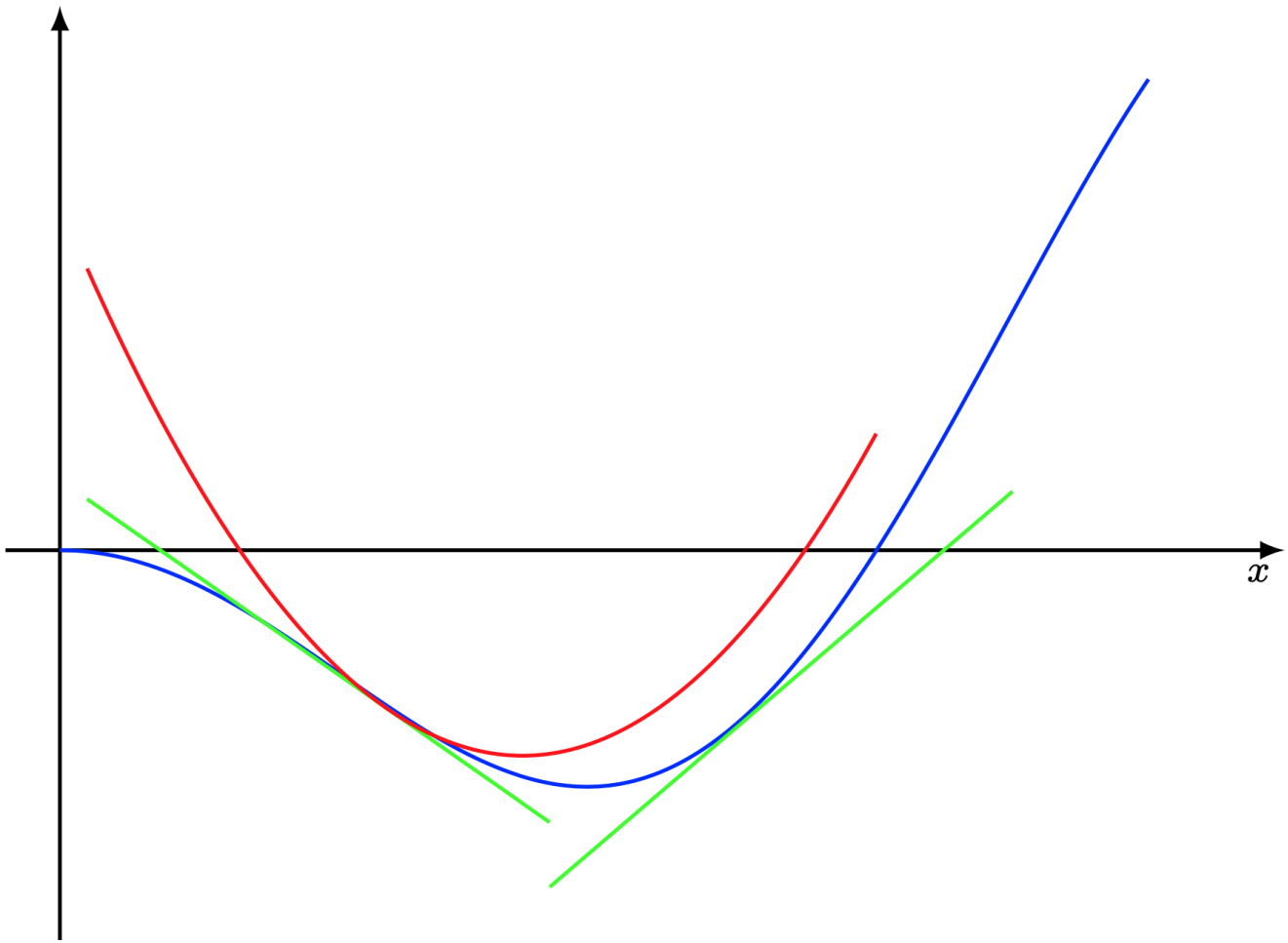
Example: Information available at x_{k+1}

At a new iterate, we observe function and gradient information at the last iterate:



Example: Quadratic model at x_{k+1}

We create a quadratic model that yields the same gradient value at the last iterate:



In 1D, the secant equation yields a unique Hessian. This is not true for $n > 1$.

- For an n dimensional problem, the square matrix H_{k+1} has n^2 entries.
- Requiring H_{k+1} to be symmetric reduces the degrees of freedom (d.o.f.) to

$$\frac{n(n+1)}{2}$$

- Requiring H_{k+1} to satisfy the secant equation reduces it to

$$\frac{n(n+1)}{2} - n$$

(In other words, $H_{k+1}s_k = y_k$ has an infinite number of solutions for $n \geq 2$.)

- (Optional) Requiring H_{k+1} to be positive definite may reduce it to as low as

$$\frac{n(n+1)}{2} - 2n$$

Even for small n, not all d.o.f. are absorbed, so we must choose between options.

Outline

The symmetric-rank-1 (SR1) method requires H_{k+1} to be symmetric and enforces: $H_{k+1}s_k = y_k$ (secant equation)

$$H_{k+1} = H_k + \sigma vv^T \text{ (rank 1 update);}$$

That is, the difference between H_k and H_{k+1} is σvv^T (symmetric with rank 1).

- Multiplying the rank 1 update equation by s_k and using the secant equation:
$$y_k = H_{k+1}s_k = H_k s_k + \sigma(v^T s_k)v.$$

- If $(y_k - H_k s_k)^T s^k \neq 0$, then for some δ we have $v = \delta(y_k - H_k s_k)$, so
$$y_k - H_k s_k = \sigma\delta^2((y_k - H_k s_k)^T s_k)(y_k - H_k s_k).$$

- One can show that the only solution to the above equation is
$$\sigma = \text{sign}((y_k - H_k s_k)^T s^k), \delta^2 = |(y_k - H_k s_k)^T s_k|^{-1}.$$

- The only symmetric-rank-1 update satisfying the secant equation is

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

- Applying the Sherman-Morrison formula, we also have the update

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

This latter update is particularly useful in a line search method since we want

$$d_{k+1} = -H_{k+1}^{-1} \nabla f(x_{k+1}),$$

(assuming $H_{k+1} \succ 0$) though we want the former in a trust region method.

Sherman–Morrison–Woodbury formula (SMW formula) :

$$(\mathbf{D} + \mathbf{V}\mathbf{V}^T)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^T\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^T\mathbf{D}^{-1}$$

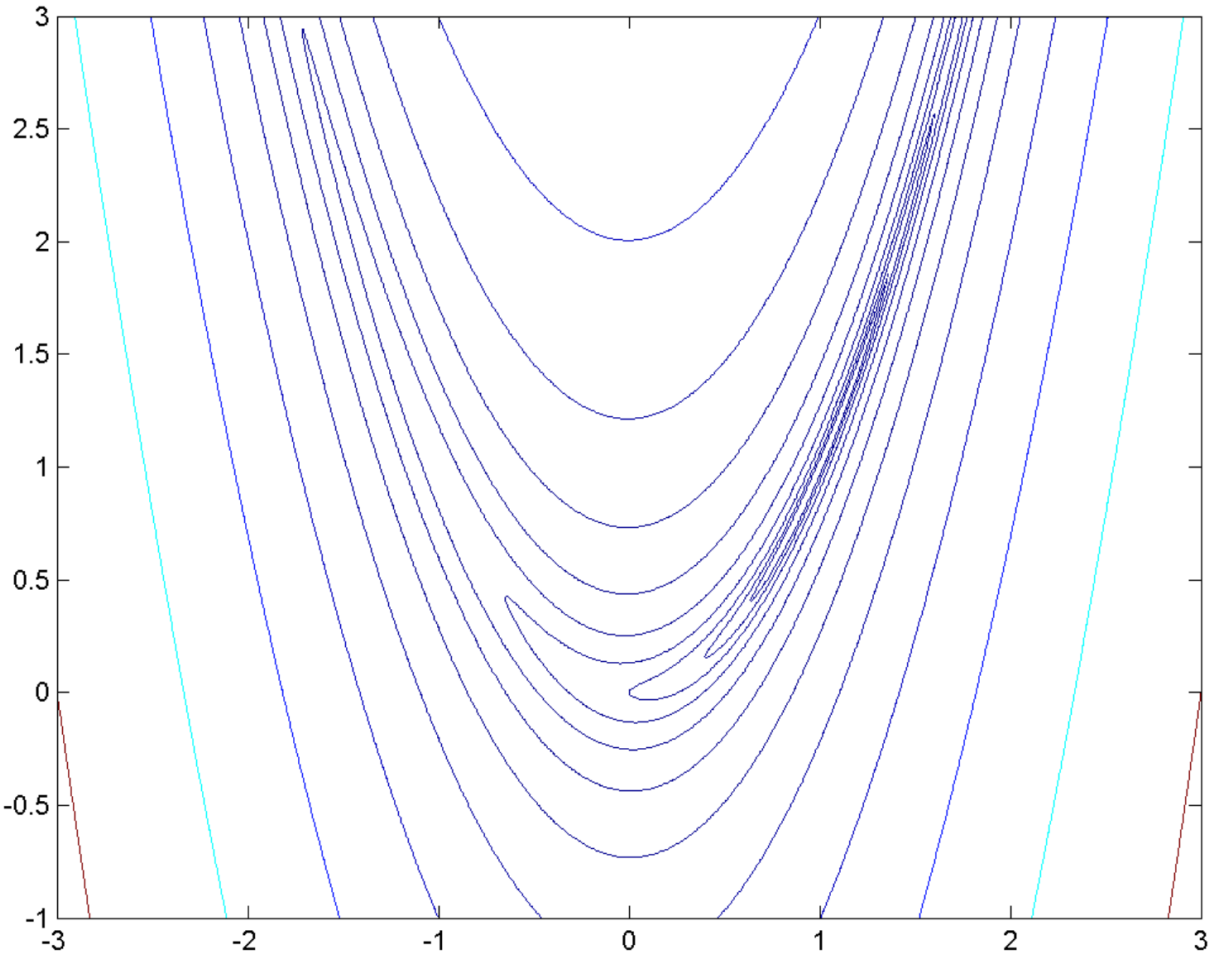
Consider the update:

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

- The method may breakdown unless we consider the following cases:
  - If $(y_k - H_k s_k)^T s_k \neq 0$, there is a unique update, as described.
  - If $y_k = H_k s_k$, only SR1 update satisfying the secant equation is
  $$H_{k+1} = H_k.$$
  - If $y_k \neq H_k s_k$, but $(y_k - H_k s_k)^T s_k = 0$, then the update is not well-defined.

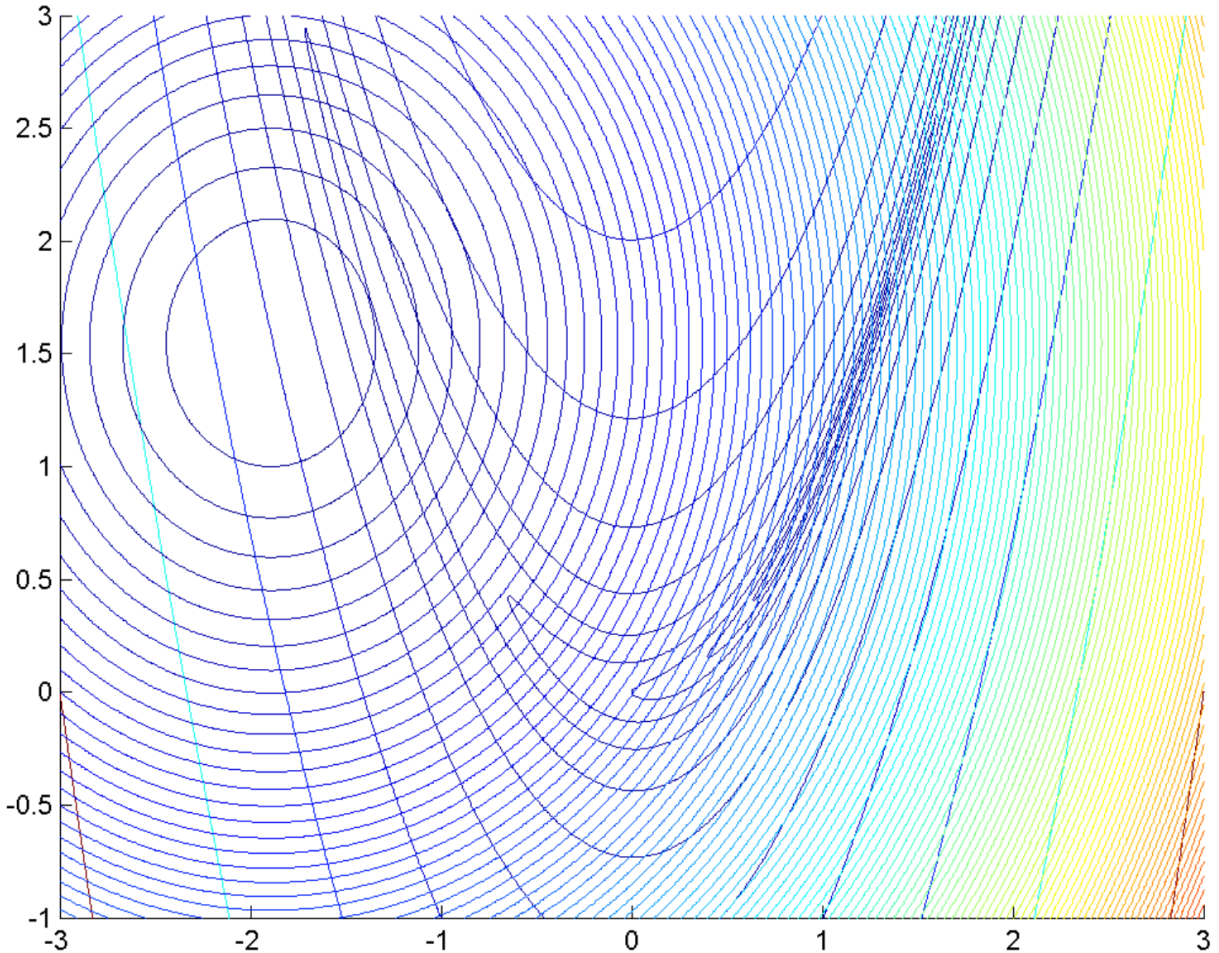
  Thus, it is common to skip the update (i.e., set $H_{k+1} = H_k$) unless
  $$|(y_k - H_k s_k)^T s_k| \geq c\|y_k - H_k s_k\|\|s_k\|$$
  where $c \in (0, 1)$ is a user-specified constant.
- Even if $H_k \succ 0$, there is no guarantee that the update will yield $H_{k+1} \succ 0$. Thus, we may either have to modify the matrix or use a trust region approach.

Recall the Rosenbrock function:

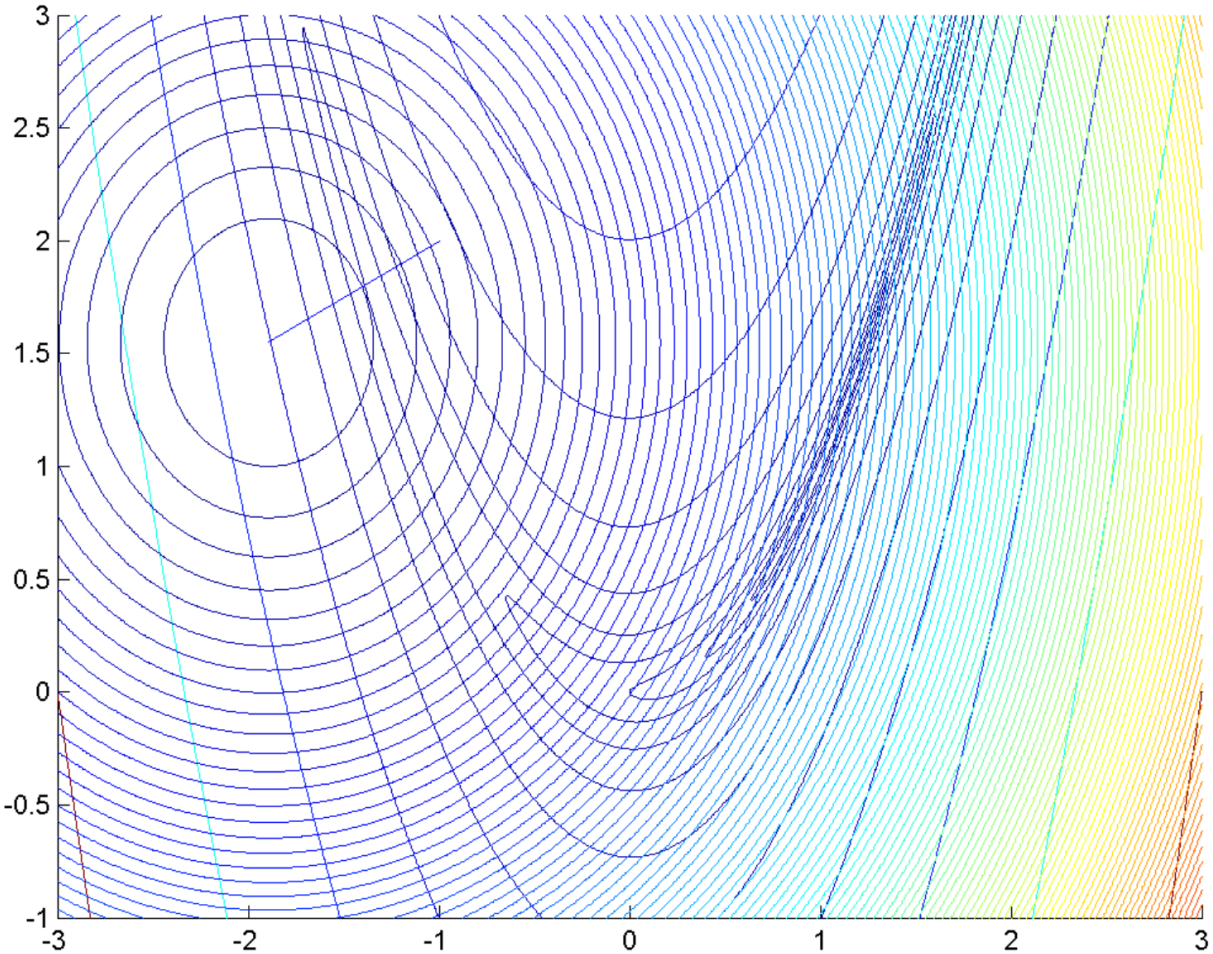


Example: Initial quadratic model

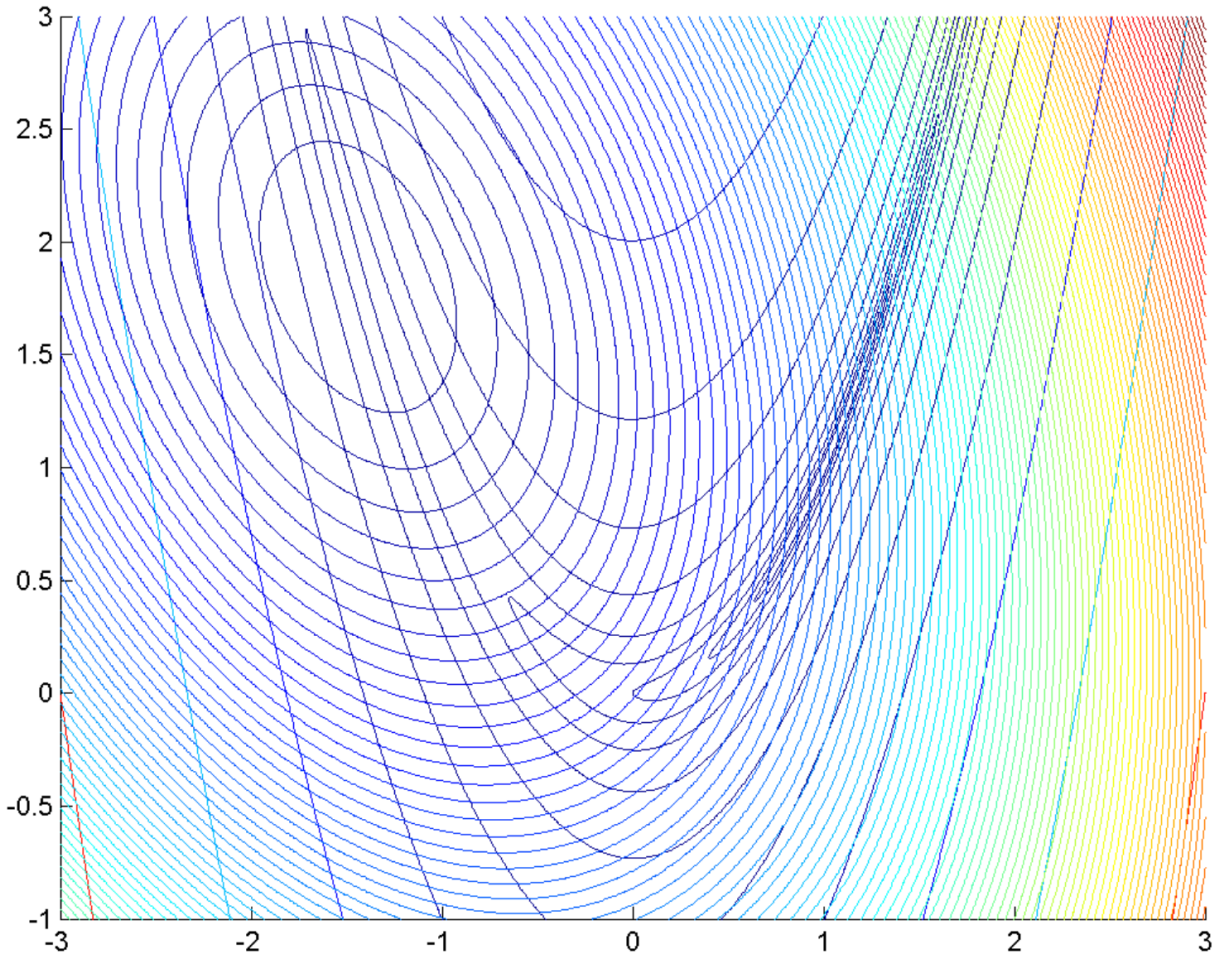
Suppose we start with $H_0 = \gamma I$ at $x_0 = (-1, 2)$:



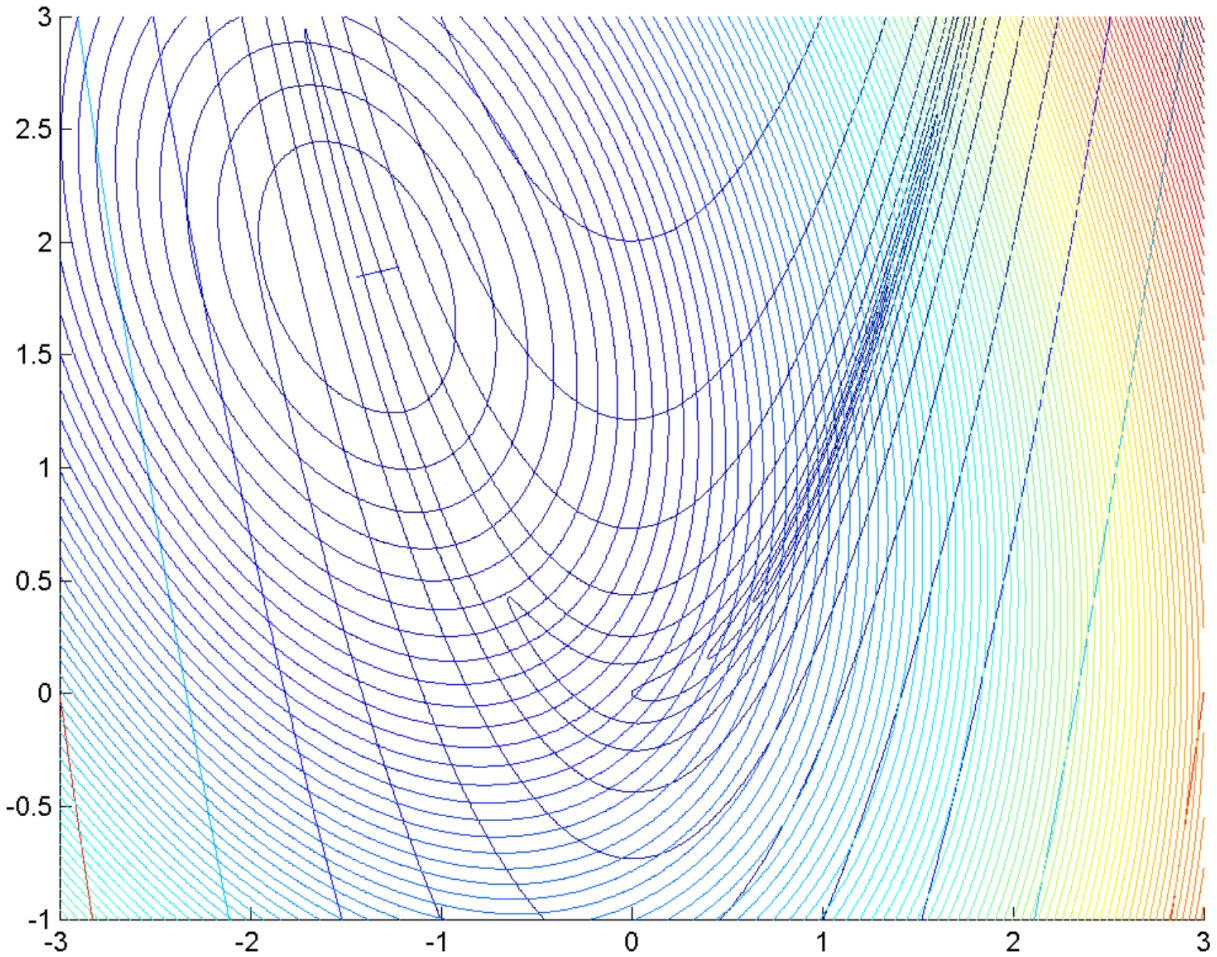
The step is obtained by minimizing the quadratic model:



After a line search, we have a new point and use the update to set H_1:



The step is again obtained by minimizing the quadratic model:



- SR1 is a reasonable update.
- However, there are better options.
- It can be shown to converge "superlinearly" (discussed later), but only if we never skip an update—something we cannot guarantee in general.

Outline

- SR1 defined the form of the update: adding a symmetric-rank-1 matrix.
- In this section, we discuss rank 2 updates.
- However, we come at the update from a different angle.
- In particular, we ask the following question after determining x_{k+1}: "Among all symmetric matrices satisfying the secant equation $H_{k+1}s_k = y_k$, which matrix is the closest to H_k?"
- In other words, we set H_{k+1} as the solution to the optimization problem:

$$\begin{aligned} \min_{H} \quad & \|H - H_k\| \\ \text{s.t.} \quad & H = H^T, Hs_k = y_k. \end{aligned}$$

- Desiring a matrix "close" to H_k helps ensure stability of the algorithm—so the Hessian matrices don't vary widely and erratically between iterations.

- Of course, "close" in vector spaces depends on the choice of norm.
- For many norms, the solution to this problem is difficult to obtain.
- However, for a certain weighted Frobenius norm, there is a unique solution:

$$H_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) + \frac{y_k y_k^T}{y_k^T s_k}.$$

  where y_k and s_k are defined as before. This is the DFP update.
- One can see that H_{k+1} is H_k plus a rank-2 matrix.
- Applying the Sherman-Morrison-Woodbury formula, we also have the update:

$$H_{k+1}^{-1} = H_k^{-1} - \frac{H_k^{-1} y_k y_k^T H_k^{-1}}{y_k^T H_k^{-1} y_k} + \frac{s_k s_k^T}{y_k^T s_k},$$

  which is most useful in line search methods.

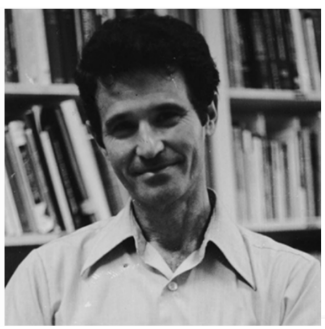
Sherman–Morrison–Woodbury formula (SMW formula) :

$$(\mathbf{D} + \mathbf{V}\mathbf{V}^T)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^T\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^T\mathbf{D}^{-1}$$

Figure 1. DFP: Bill Davidon, Roger Fletcher FRS and Mike, Trinity Hall, Cambridge 1981

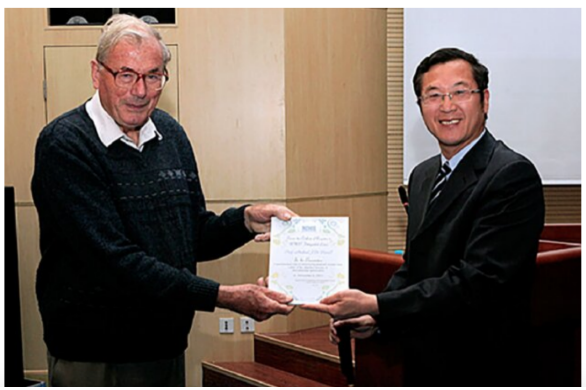
M. J. D. Powell

Michael J. D. Powell (1936–2015): Cambridge



William C. Davidon (1927–2013): Argonne->Haverford

Roger Fletcher (1939–2016): University of Dundee, Lagrange Prize



- Davidon can be viewed as the founder of quasi-Newton methods, and the DFP update was his creation (later analyzed by Fletcher and Powell).
- However, it was soon found that better practical results could be obtained by designing the update more directly for the inverse of the Hessian; i.e.,

$$\begin{aligned} \min_{H} \quad & \|H^{-1} - H_k^{-1}\| \\ \text{s.t.} \quad & H^{-1} = H^{-T}, H^{-1} y_k = s_k. \end{aligned}$$

- Using a same strategy as before, for a certain norm we get a simple solution:

$$H_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

  This is known as a BFGS update.

- Applying the Sherman-Morrison-Woodbury formula, we also have the update:

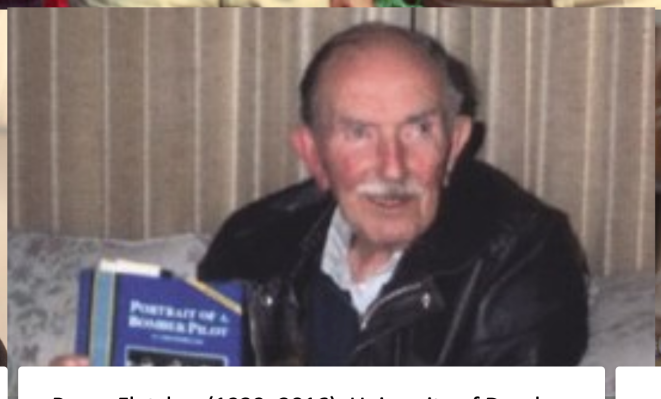
$$H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}.$$

- Again, the update takes the form of an added rank-2 matrix.

Broyden, Fletcher, Goldfarb, Shanno

Donald Goldfarb (1941–): co-founded NYU-CS, Columbia University

Charles G. Broyden (1933–2011): Aberystwyth->Essex->UK National Physical Lab->Bologna.



Roger Fletcher (1939–2016): University of Dundee, Lagrange Prize

David F. Shanno (1938–2016): Chicago->Toronto->Arizona->UCD->Rutgers University.

- The SR1 method is commonly only used in trust region methods.
- BFGS, however, works well for both line search and trust region methods.
- In the former case, we want to maintain a positive definite approximation.
- Fortunately, as long as $H_k \succ 0$ and the curvature condition

$$s_k^T y_k > 0$$

  is satisfied, then H_{k+1} will also be positive definite.
- But how to ensure that the curvature condition will hold?

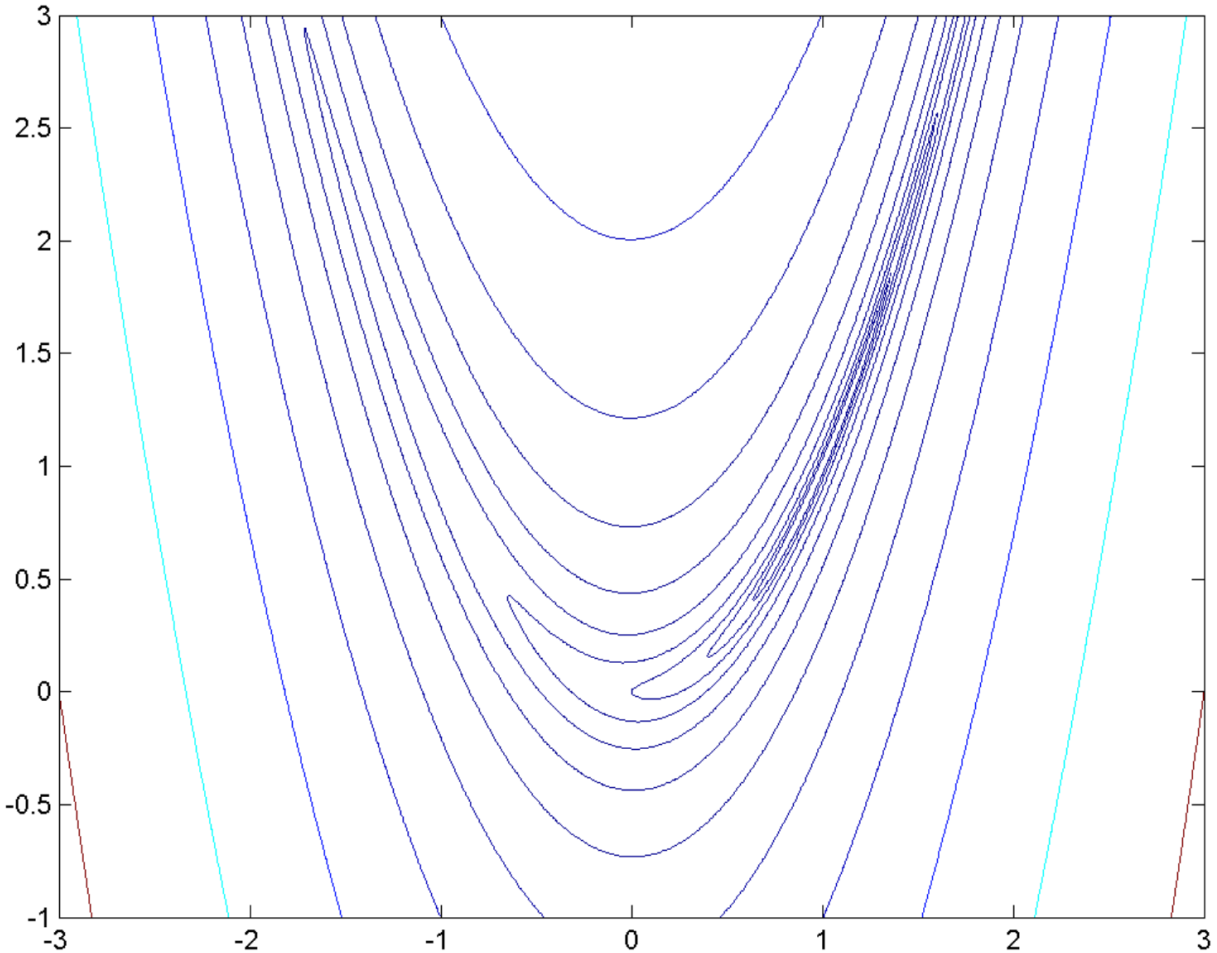
Using a Wolfe line search, we ensure that

$$\nabla f(x_{k+1})^T d_k \geq c_2 \nabla f(x_k)^T d_k.$$

Thus, (assuming $s_k \neq 0$)

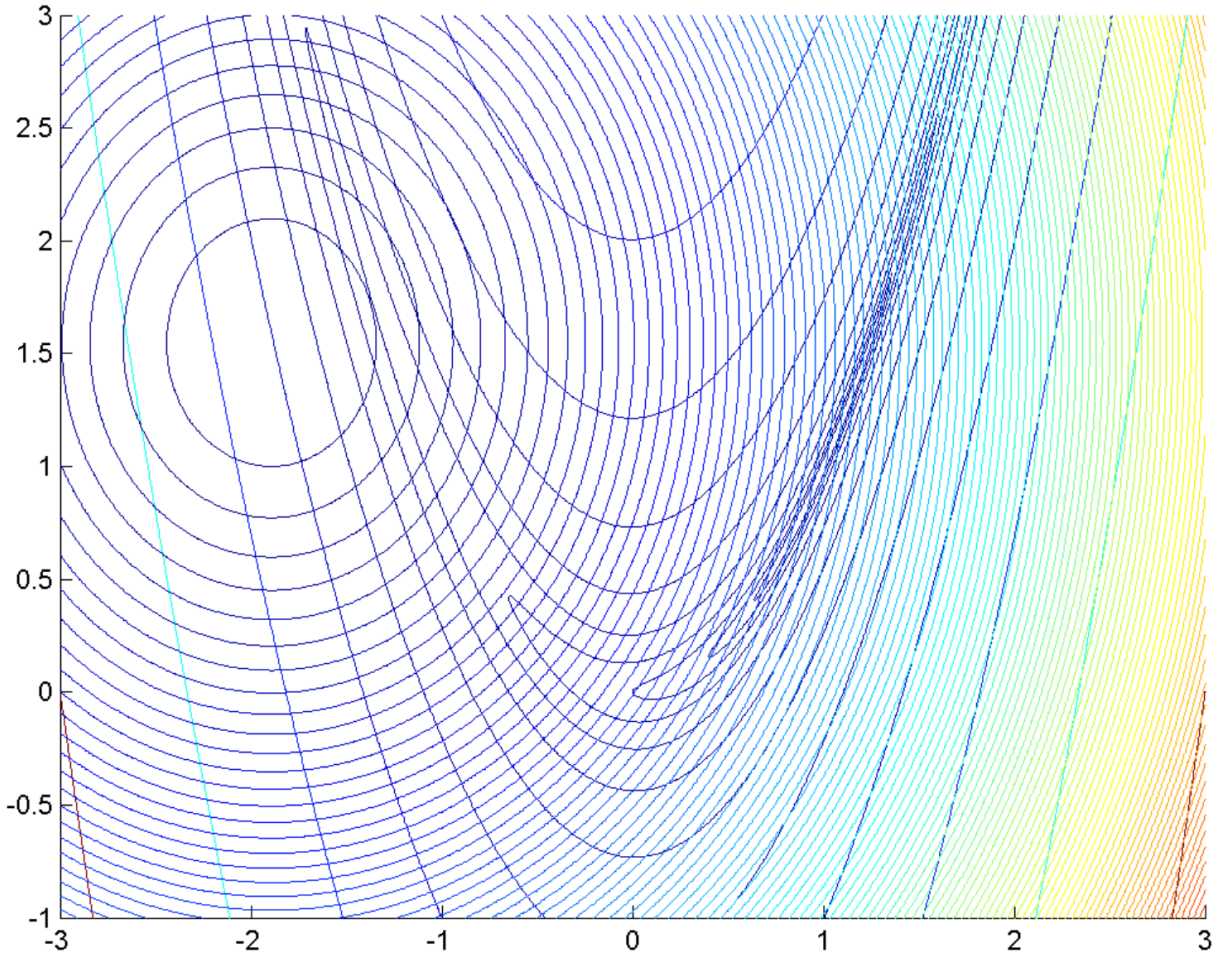
$$\begin{aligned} y_k^T s_k &= (\nabla f(x_{k+1}) - \nabla f(x_k))^T (\alpha_k d_k) \\ &= \alpha_k (\nabla f(x_{k+1})^T d_k - \nabla f(x_k)^T d_k) \\ &\geq \alpha_k (c_2 - 1) \nabla f(x_k)^T d_k > 0. \end{aligned}$$

Recall the Rosenbrock function:

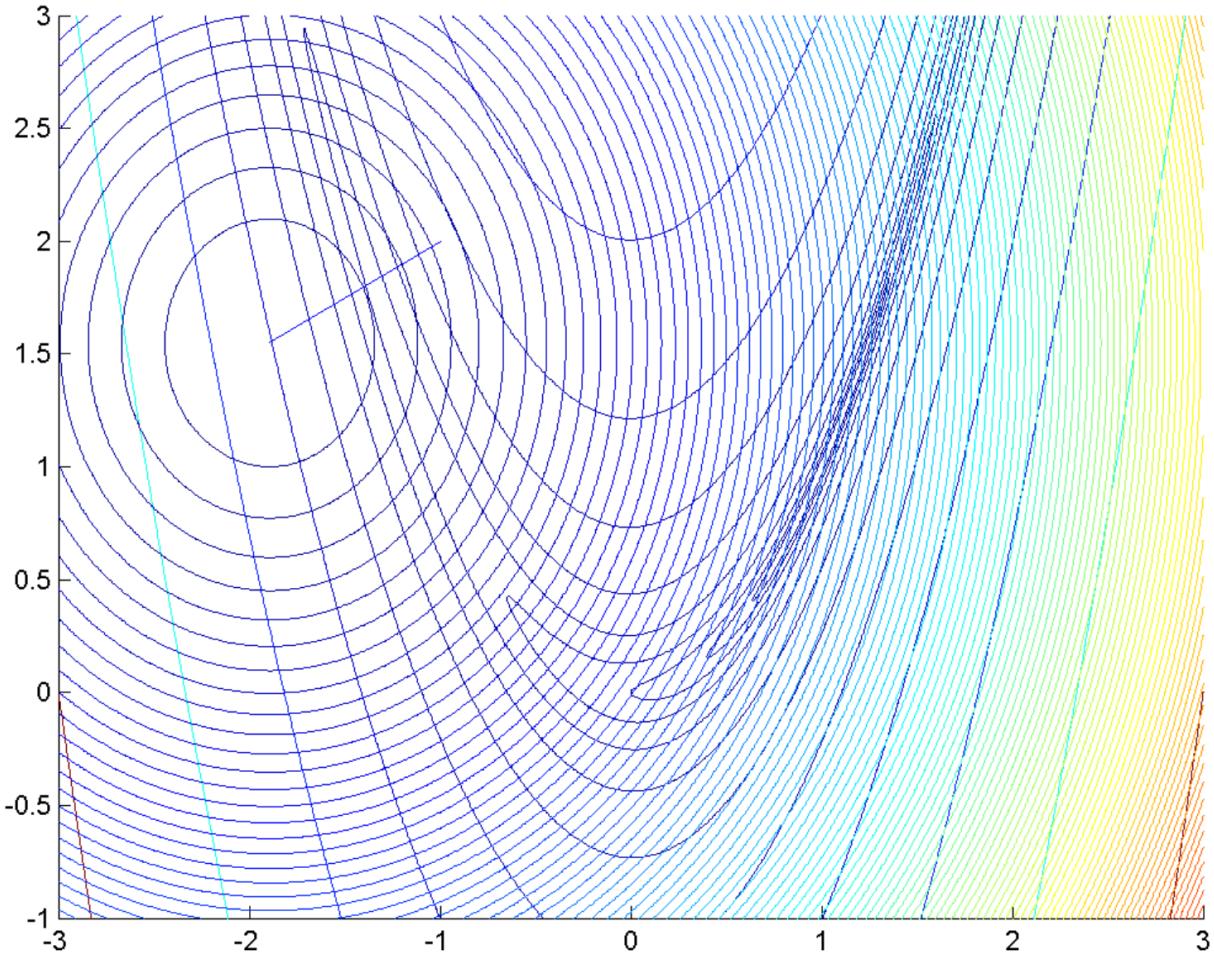


Example: Initial quadratic model

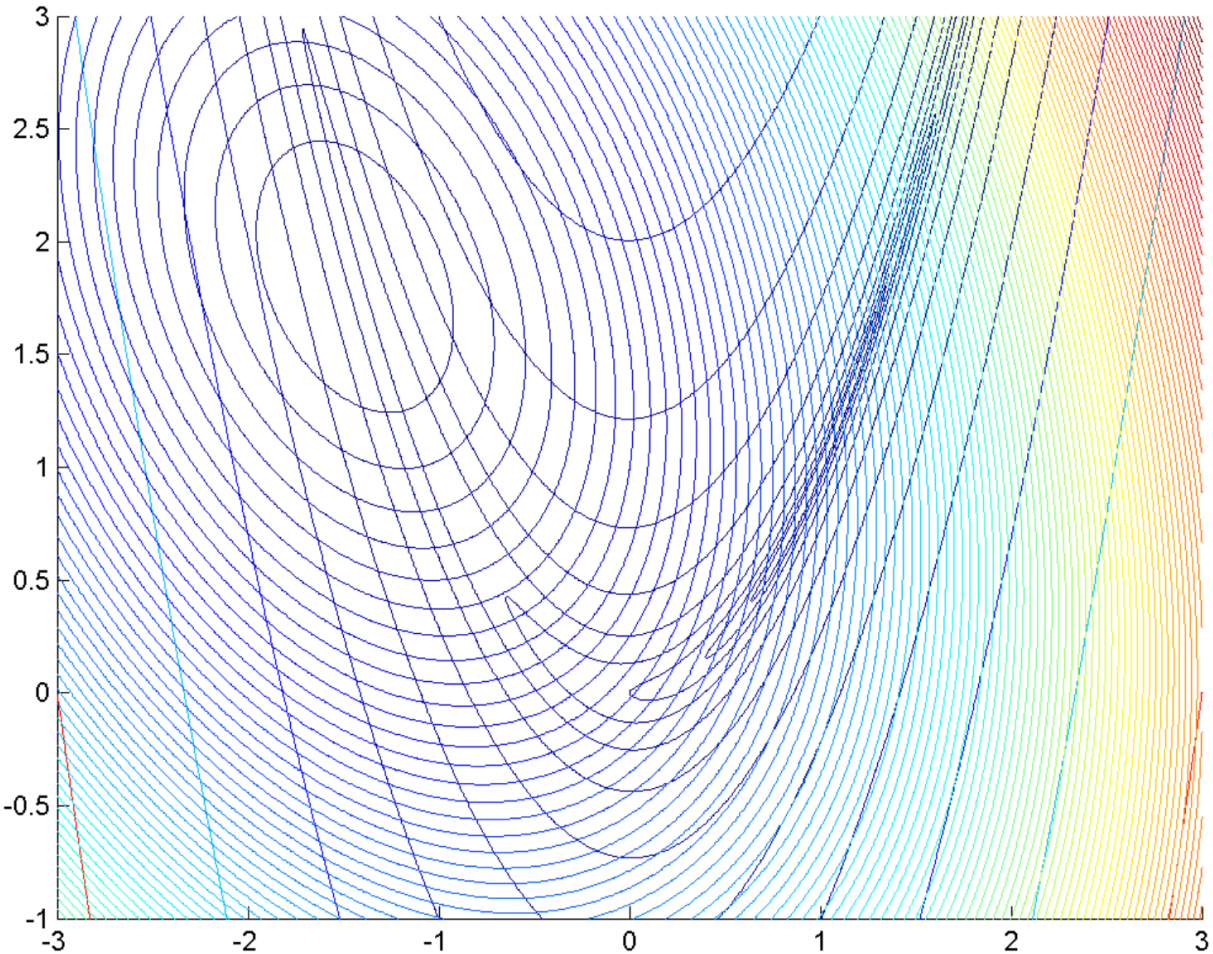
Suppose we start with $H_0 = \gamma I$ at $x_0 = (-1, 2)$:



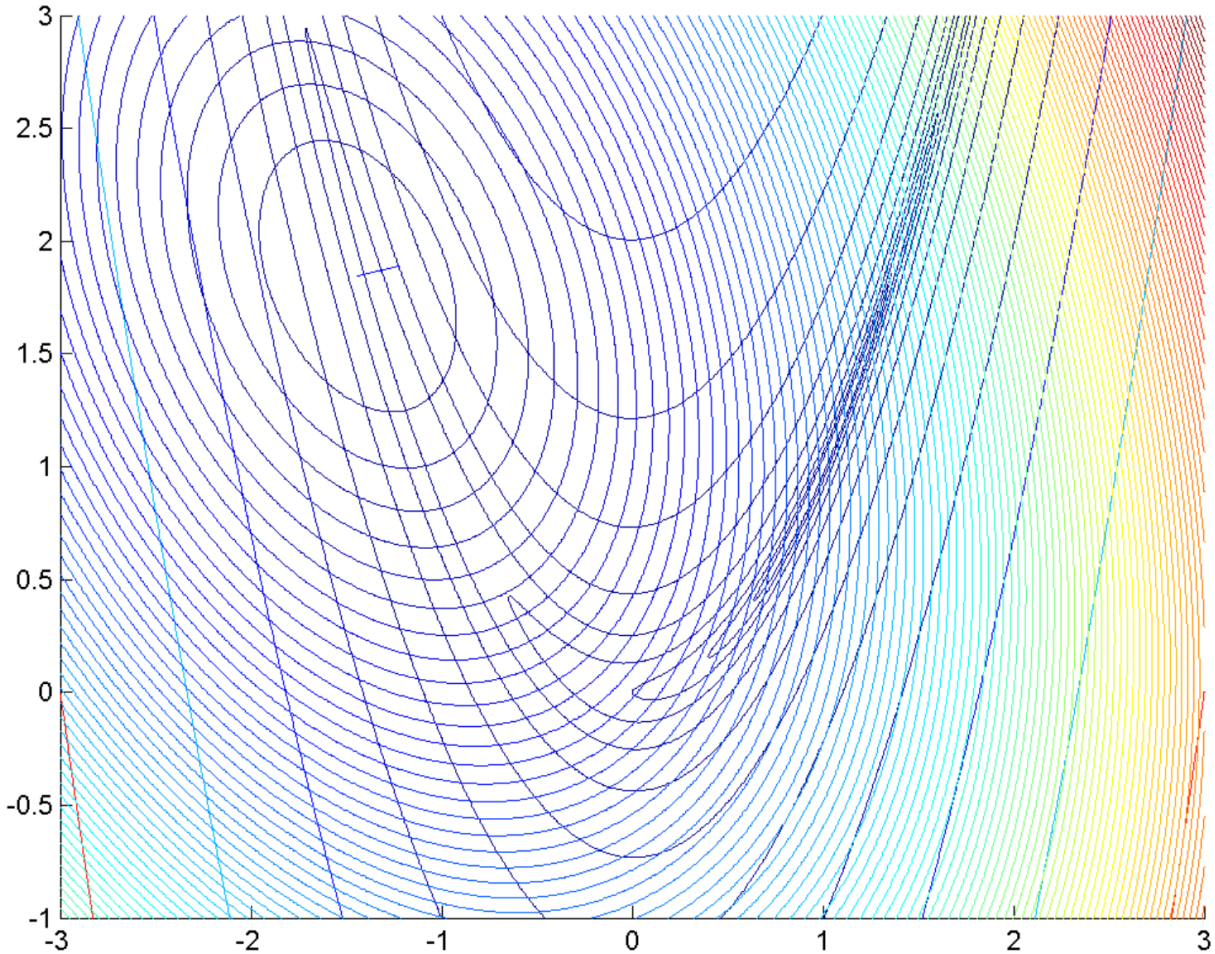
The step is obtained by minimizing the quadratic model:



After a line search, we have a new point and use the update to set H_1:



The step is again obtained by minimizing the quadratic model:



- DFP and BFGS are both reasonable updates.
- The general consensus is that BFGS is the best known quasi-Newton update.
- It can be shown to converge superlinearly (discussed later).
- There are practical issues to consider, such as the following:
  - Without a Wolfe line search, the curvature condition may fail to hold. The update can be skipped in such cases, but this is ill-advised as it will slow convergence.
  - If $y_k^T s_k \approx 0$, there can be numerical issues.
  - The Hessian approximations may become poorly conditioned.
  - Results are highly dependent on the choice of H_0 and if one poor approximation is encountered, then the approximations may continue to be bad for many iterations.
  - The Hessian approximations are dense.

  However, with careful implementations, these issues can be overcome.

If $y_k^T s_k$ is not sufficiently positive, i.e., if we do not have $y_k^T s_k \geq \theta s_k^T H_k s_k$ for some $\theta \in (0, 1)$, then instead of skipping the update, we can modify it.

- Define $s_k = \alpha_k d_k$ and $r_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, similar to before, then set

$$y_k = \theta_k r_k + (1 - \theta_k) H_k s_k,$$

where

$$\theta_k = \begin{cases} 1 & \text{if } r_k^T s_k \geq 0.2 s_k^T H_k s_k \\ \frac{0.8 s_k^T H_k s_k}{s_k^T H_k s_k - r_k^T s_k} & \text{otherwise.} \end{cases}$$

- We have redefined y_k so that we have now guaranteed (assuming $s_k = 0$)

$$y_k^T s_k \geq 0.2 s_k^T H_k s_k > 0.$$

- For general nonlinear function, the true Hessian changes between iterations.
- However, due to the result in the last section, we can expect a quasi-Newton approximation to be relatively accurate in the neighborhood of a solution point (since many iterates will remain in the neighborhood, thus improving the model in that region).
- In particular, for fast local convergence we find that the model is sufficiently accurate along the search path to yield superlinear convergence.

Theorem 1

Suppose the following conditions (which imply a unique minimizer) hold:

- f *is twice continuously differentiable;*
- *The level set* $\mathcal{L} := \{x : f(x) \leq f(x_0)\}$ *is convex;*
- *There exist constants* $m > 0$ *and* $M > 0$ *such that*

$$m\|d\|^2 \leq d^T \nabla^2 f(x) d \leq M\|d\|^2$$

for all $x \in \mathcal{L}$ *and* $d \in \mathbb{R}^n$.

Then, with any x_0 *and* H_0*, the sequence generated by the BFGS method with a Wolfe line search converges to the minimizer* x_* *of* f.

Theorem 2

Suppose the following conditions hold:

- f *is twice continuously differentiable;*
- *BFGS converges to a minimizer* x_* *at which* $\nabla^2 f$ *is Lipschitz continuous;*
- *The iterates satisfy*

$$\sum_{k=1}^{\infty} \|x_k - x_*\| < \infty.$$

(This can be proved under the conditions of the previous theorem.)

Then, $x_k \to x_*$ *at a superlinear rate.*

The BFGS method computes inverse Hessian approximations of the form

$$H_{k+1}^{-1} = V_k^T H_k^{-1} V_k + \rho_k s_k s_k^T,$$

where

$$\rho_k = \frac{1}{y_k^T s_k} \text{ and } V_k = I - \rho_k y_k s_k^T$$

and

$$s_k = x_{k+1} - x_k \text{ and } y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

This means that at iteration k, the matrix H_{k+1}^{-1} has been built from:

$$H_0^{-1}, s_0, y_0, s_1, y_1, ..., s_k, y_k.$$

Moreover, H_{k+1}^{-1} will generally be a dense matrix, which may be a lot to store.

We can

1. reduce the memory requirements and
2. "flush out" old information

by forming the updating based solely on an initial matrix $\tilde{H}_{k+1}$ (often simple) and the m most recent pairs of $\{(s_j, y_j)\}$ values:

$$\tilde{H}_{k+1}, s_{k+1-m}, y_{k+1-m}, \ldots, s_k, y_k.$$

The resulting update is actually simple to calculate and is quite popular (see the discussion of L-BFGS in §7.2 of the textbook).