# Numerical Optimization

## Lecture 14: Newton Method & Quasi-Newton Method

**王浩**

**信息科学与技术学院**

**Email: wanghao1@shanghaitech.edu.cn**

# Newton Methods

# Systems of equations

Suppose we aim to solve the system of nonlinear equation(s):

$$F(x) = 0.$$

Examples of such systems abound.

- Pedagogical example: Calculate $\sqrt{z}$ by solving $x^2 - z = 0$.
- More examples: Practically all modern optimization methods in some way reduce to Newton's method for solving systems of nonlinear equations!

# Newton's method

We can motivate Newton's Method in 3 ways (that are basically all the same).

- At the current point $(x_k, F(x_k))$, draw a tangent line until it hits the $x$-axis; call that point $x_{k+1}$.
- Create an affine model of $F$ at $x_k$:

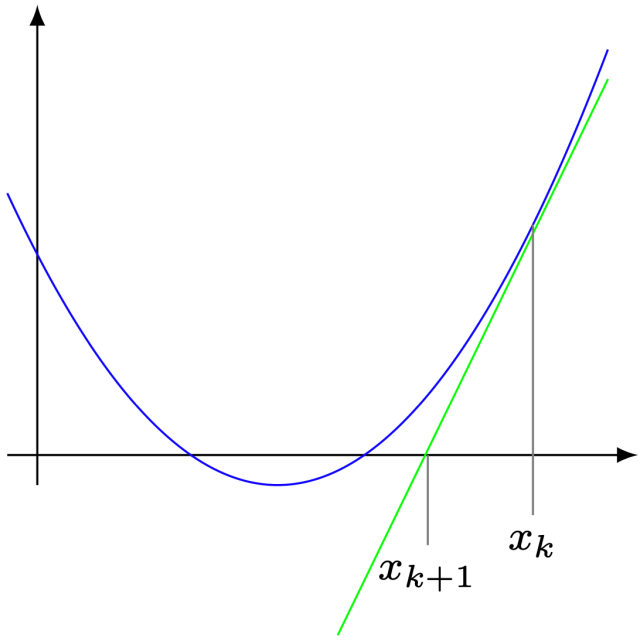
$$m_k(x) = F(x_k) + F'(x_k)(x - x_k);$$

call $x_{k+1}$ the solution to $m_k(x) = 0$.

- Write the Taylor series of $F$ at $x_k$:

$$F(x) = F(x_k) + F'(x_k)(x - x_k) + \tfrac{1}{2}F''(x_k)(x - x_k)^2 + \cdots;$$

approximate $F(x)$ with the affine portion, solve the resulting affine equation, and call the solution $x_{k+1}$.



# Newton's method

The "affine model" is perhaps the best interpretation.

- We can write an equation, and then approximate an integral:

$$\begin{aligned} F(x) = &\ F(x_k) + \int_{x_k}^{x} F'(z)dz \\ \approx &\ F(x_k) + F'(x_k)(x - x_k) =: m_k(x). \end{aligned}$$

- Solving for $x$ in $m_k(x) = 0$ yields the familiar formula:

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}.$$

Will it work??? Yes! (sometimes)

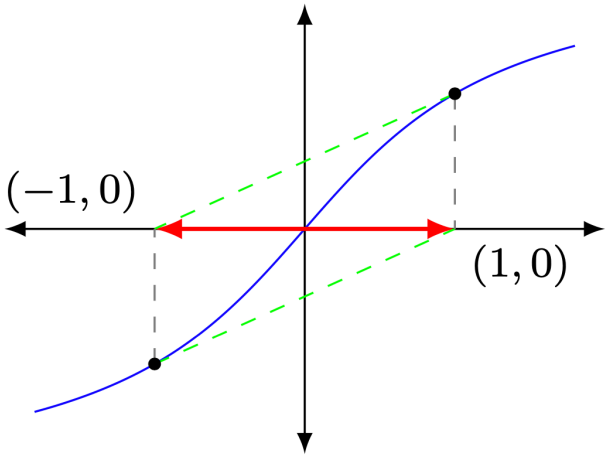
# Failures of Newton's method

Newton's method can fail in many ways:

- Certain starting points can lead to cycling and even divergence.
- May have $F'(x_k) = 0$. (So what?)
- $F(x_k)$ may be undefined/imaginary.

Also, it might not fail, but in some situations it can converge very slooooooowly...

Still, it is very powerful.



# Newton's method (multivariable)

Suppose we have $F : \mathbb{R}^n \to \mathbb{R}^n$.

- Form an affine model of the function at $x_k$:

$$\begin{aligned} F(x) &= F(x_k) + \int_{[x_k, x]} \nabla F(z)^T \cdot dz \\ &\approx F(x_k) + \nabla F(x_k)^T (x - x_k). \end{aligned}$$

- Solve for the step/displacement $d_k$ (i.e., let $d = x - x_k$):

$$\nabla F(x_k)^T d = -F(x_k).$$

- Update the iterate

$$x_{k+1} = x_k + d_k.$$

- We can think about this as simultaneously solving individual affine models for each of the components of $F$:

$$F^i(x) \approx F^i(x_k) + \nabla F^i(x_k)^T d \quad \Rightarrow \quad \nabla F^i(x_k)^T d = -F^i(x_k), \ i = 1, \ldots, n.$$

# Example

Suppose we aim to solve

$$F(x) = \begin{bmatrix} x_1^2 + x_1 x_2 \\ e^{x_1} - x_2 \end{bmatrix} = 0.$$

(Can we solve this analytically? How many solutions does it have?)

- Pick a starting point, say $x_0 = (1, 1)$.
- Evaluate $F(x)$ and $\nabla F(x)^T$:

$$F(x) = \begin{bmatrix} x_1^2 + x_1 x_2 \\ e^{x_1} - x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ e - 1 \end{bmatrix}$$

$$\text{and} \quad \nabla F(x)^T = \begin{bmatrix} 2x_1 + x_2 & x_1 \\ e^{x_1} & -1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ e & -1 \end{bmatrix}.$$

- Solve the Newton system:

$$\begin{bmatrix} 3 & 1 \\ e & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = - \begin{bmatrix} 2 \\ e - 1 \end{bmatrix}.$$

- Update $x_1 = x_0 + d_0$, reevaluate, solve, update, reevaluate, solve....

# Example results

```
>> x = newton('example',[1;1]);
================================
  k      ||F(x)||        ||d||
================================
   0   2.6368e+000   6.5211e-001
   1   6.5261e-001   2.9418e-001
   2   8.5037e-002   5.8346e-002
   3   4.1779e-003   4.2638e-003
   4   2.2801e-005   2.6147e-005
   5   8.2119e-010   -----------
================================
>> x

x =

    0.0000
    1.0000
```

```
>> x = newton('example',[-1;1]);
================================
  k      ||F(x)||        ||d||
================================
   0   6.3212e-001   6.5353e-001
   1   4.6100e-002   4.1159e-002
   2   2.4495e-004   2.2103e-004
   3   6.9278e-009   -----------
================================
>> x

x =

   -0.5671
    0.5671
```

# Example

Suppose we aim to solve

$$F(x) = \arctan(x) = 0.$$

(Can we solve this analytically? How many solutions does it have?)

- Pick a starting point, say $x_0 = 1$.
- Evaluate $F(x)$ and $\nabla F(x)^T$:

$$F(x) = \arctan(x) \approx 0.7854$$

$$\text{and} \quad \nabla F(x)^T = \frac{1}{1+x^2} = \frac{1}{2}.$$

- Solve the Newton system:

$$\frac{1}{2} d = -0.7854.$$

- Update $x_1 = x_0 + d_0$, reevaluate, solve, update, reevaluate, solve....

# Example results

```
>> x = newton('example2',1.391);
==============================
  k     ||F(x)||       ||d||
==============================
   0  9.4749e-001  2.7808e+000
   1  9.4708e-001  2.7763e+000
   2  9.4598e-001  2.7647e+000
   3  9.4308e-001  2.7342e+000
   4  9.3539e-001  2.6555e+000
   5  9.1489e-001  2.4597e+000
   6  8.5946e-001  2.0165e+000
   7  7.0810e-001  1.2272e+000
   8  3.5527e-001  4.0417e-001
   9  3.3148e-002  3.3184e-002
  10  2.4303e-005  2.4303e-005
  11  9.5692e-015  -----------
==============================
>> x

x =

 -9.5692e-015
```

```
>> x = newton('example2',1.392);
==============================
  k     ||F(x)||       ||d||
==============================
   0  9.4783e-001  2.7844e+000
   1  9.4798e-001  2.7859e+000
   2  9.4835e-001  2.7899e+000
   3  9.4934e-001  2.8006e+000
   4  9.5194e-001  2.8288e+000
   5  9.5878e-001  2.9047e+000
   6  9.7661e-001  3.1160e+000
   7  1.0221e+000  3.7576e+000
   8  1.1304e+000  6.2188e+000
   9  1.3314e+000  2.3681e+001
  10  1.5198e+000  5.8438e+002
  11  1.5690e+000  5.0052e+005
  12  1.5708e+000  3.9262e+011
  13  1.5708e+000  2.4214e+023
  14  1.5708e+000  9.2101e+046
  15  1.5708e+000  1.3324e+094
  16  1.5708e+000  2.7888e+188
  17  1.5708e+000  Inf
  18  1.5708e+000  Inf
  19  NaN  -----------
==============================
>> x

x =

   NaN
```

# Quadratic convergence of Newton's method

- Newton's method converges quadratically! (under nice assumptions)

```
>> x = newton('example',[-1;1]);
================================
  k      ||F(x)||       ||d||
================================
  0   6.3212e-001   6.5353e-001
  1   4.6100e-002   4.1159e-002
  2   2.4495e-004   2.2103e-004
  3   6.9278e-009   -----------
================================
>> x

x =

   -0.5671
    0.5671
```

- (Why else do you think something so old is still around!)
- We will go through the entire proof.
- First, we need to discuss our assumptions...

# Assumptions

## Assumption

*For some point $x_* \in \mathbb{R}^n$ such that $F(x_*) = 0$, the following hold:*

- *$F$ is continuously differentiable in an open convex set $\mathcal{X} \subseteq \mathbb{R}^n$ with $x_* \in \mathcal{X}$.*
- *The Jacobian of $F$ at $x_*$ is invertible and is bounded in norm by $M > 0$, i.e.,*

$$\|(\nabla F(x_*)^T)^{-1}\|_2 \leq M.$$

- *For some neighborhood of $x_*$ with radius $r > 0$ contained in $\mathcal{X}$, i.e.,*

$$\mathbb{B}(x_*, r) := \{x \in \mathbb{R}^n \mid \|x - x_*\|_2 \leq r\} \in \mathcal{X},$$

*the Jacobian of $F(x)$ is Lipschitz continuous with constant $L$ in $\mathbb{B}(x_*, r)$.*

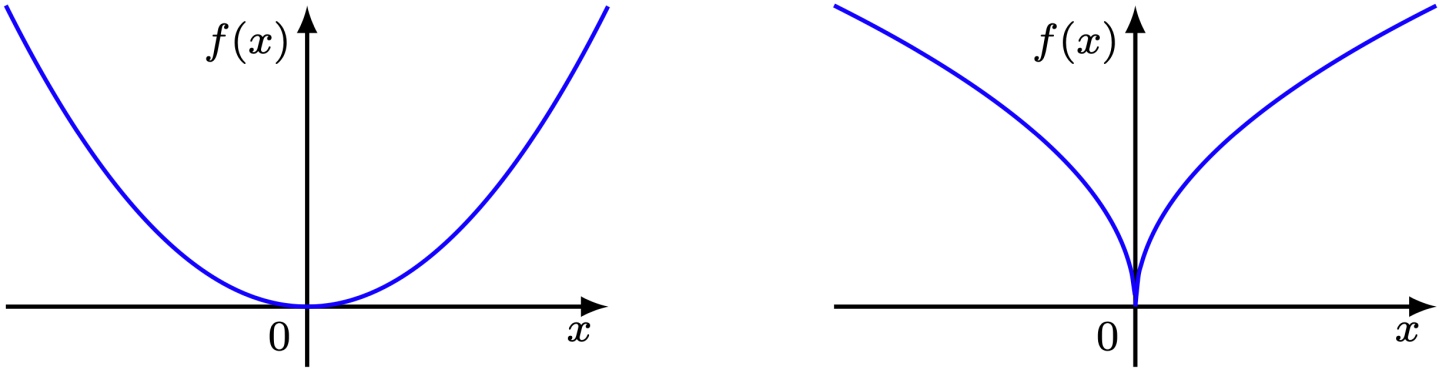
# Lipschitz continuity

**Definition**

A function $G$ is Lipschitz continuous with constant $L \geq 0$ in $\mathcal{X}$ if

$$\|G(x_1) - G(x_2)\|_2 \leq L\|x_1 - x_2\|_2 \quad \text{for all} \quad \{x_1, x_2\} \subseteq \mathcal{X}.$$

- What does it mean? There is a limit to how fast $G$ changes.




- Thus, in our proof, we assume there is a limit to how fast the Jacobian of $F(x)$ changes (in norm) in a neighborhood of $x_*$.

# Quadratic convergence of Newton's method

**Theorem** (Quadratic convergence of Newton's method)

*There exists $\epsilon > 0$ such that for all $x_0 \in \mathbb{B}(x_*, \epsilon)$, the sequence defined by*

$$\begin{aligned} \nabla F(x_k)^T d_k &= -F(x_k) \\ x_{k+1} &= x_k + d_k, \quad k = 0, 1, 2, \ldots \end{aligned}$$

*is well-defined, converges to $x_*$, and for some $c > 0$ satisfies*

$$\|x_{k+1} - x_*\|_2 \leq c\|x_k - x_*\|_2^2.$$

# Quadratic convergence of Newton's method

## Proof, part 1.

Consider $\bar{\epsilon} > 0$ such that, with $\|x_0 - x_*\|_2 \leq \bar{\epsilon}$, the Jacobian $\nabla F(x_0)^T$ is nonsingular. This guarantees that the iteration is well-defined at $x_0$. Let

$$\epsilon := \min\{\bar{\epsilon}, \tfrac{1}{2ML}\} > 0.$$

Recall that if $A$ is nonsingular and $\|A^{-1}(B - A)\|_2 < 1$, then $B$ is nonsingular and

$$\|B^{-1}\|_2 \leq \frac{\|A^{-1}\|_2}{1 - \|A^{-1}(B - A)\|_2}.$$

Thus, since $\nabla F(x_*)^T$ is nonsingular and

$$\begin{aligned}\|\nabla F(x_*)^{-T}(\nabla F(x_0)^T - \nabla F(x_*)^T)\|_2 \leq & \ \|\nabla F(x_*)^{-T}\|_2 \|\nabla F(x_0)^T - \nabla F(x_*)^T\|_2 \\ \leq & \ ML\|x_0 - x_*\|_2 \leq ML\epsilon \leq \tfrac{1}{2},\end{aligned}$$

we know that $\nabla F(x_0)^T$ is nonsingular and

$$\|\nabla F(x_0)^{-T}\|_2 \leq \frac{\|\nabla F(x_*)^{-T}\|_2}{1 - \|\nabla F(x_*)^{-T}(\nabla F(x_0)^T - \nabla F(x_*)^T)\|_2} \leq 2M.$$

# Quadratic convergence of Newton's method

## Proof, part 2.

We now show that for some $c > 0$ we have $\|x_1 - x_*\|_2 \leq c\|x_0 - x_*\|_2^2$. (This is the relationship we need for quadratic convergence, which will be nice if we can show that we actually converge!) The difference between $x_1$ and $x_*$ can be written as

$$\begin{aligned} x_1 - x_* &= x_0 - x_* - \nabla F(x_0)^{-T} F(x_0) \\ &= x_0 - x_* - \nabla F(x_0)^{-T}(F(x_0) - F(x_*)) \\ &= \nabla F(x_0)^{-T}(F(x_*) - \underbrace{F(x_0) - \nabla F(x_0)^T(x_* - x_0)}_{\text{affine model of } F(x) \text{ at } x_0}). \end{aligned}$$

Recalling a result from multivariable calculus, we then find

$$\begin{aligned} \|x_1 - x_*\|_2 &\leq \|\nabla F(x_0)^{-T}\|_2 \|F(x_*) - F(x_0) - \nabla F(x_0)^T(x_* - x_0)\|_2 \\ &\leq (2M)(\tfrac{1}{2}L\|x_0 - x_*\|_2^2) \\ &\leq ML\|x_0 - x_*\|_2^2. \end{aligned}$$

Thus, the result holds for $c = ML$.

# Quadratic convergence of Newton's method

## Proof, part 3.

Finally, we show that $\|x_1 - x_*\|_2 \leq \frac{1}{2}\|x_0 - x_*\|_2$. (This shows that $x_1 \in \mathbb{B}(x_*, \epsilon)$, so all of our results so far will continue to hold for $k = 1, 2, \ldots$, meaning that we converge and do so quadratically!) We have shown already that

$$\|x_1 - x_*\|_2 \leq ML\|x_0 - x_*\|_2^2,$$

and since $\|x_0 - x_*\|_2 \leq \epsilon \leq (2ML)^{-1}$ (by definition of $\epsilon$), we have

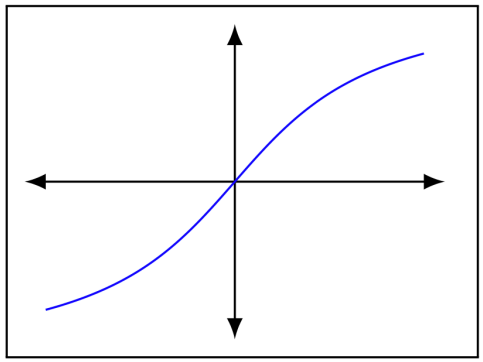
$$\|x_1 - x_*\|_2 \leq \frac{1}{2}\|x_0 - x_*\|_2.$$

# Summary

- Naturally, we need to be close enough to $x_*$ so that the function will be differentiable at all iterates; otherwise, the algorithm won't be well-defined.
- The other two constants, $L$ and $M$, play crucial roles in the proof, but also have great significance in terms of
  - how close we need to be in order to converge quadratically (i.e., $\epsilon$), and
  - how quickly we will converge once we are there (i.e., $c$).
- If $L$ is large, then the gradients of the functions change rapidly.
- If $M$ is large, then following the gradient may send us far away.

# Example

Consider $F(x) = \arctan(x)$:



- For $x_* = 0$, we have $F(x_*) = 0$.
- $F(x)$ is continuously differentiable.
- The inverse of the Jacobian at $x_*$ is

$$\nabla F(x_*)^{-T} = \left(\frac{1}{1+x_*^2}\right)^{-1} = 1 = M.$$

- The Jacobian has $L = 1$.
- Thus, the theorem says we will converge quadratically for $\epsilon = \frac{1}{2}$...

```
>> x = newton('example2',1.391);
==============================
  k      ||F(x)||       ||d||
==============================
   0  9.4749e-001  2.7808e+000
   1  9.4708e-001  2.7763e+000
   2  9.4598e-001  2.7647e+000
   3  9.4308e-001  2.7342e+000
   4  9.3539e-001  2.6555e+000
   5  9.1489e-001  2.4597e+000
   6  8.5946e-001  2.0165e+000
   7  7.0810e-001  1.2272e+000
   8  3.5527e-001  4.0417e-001
   9  3.3148e-002  3.3184e-002
  10  2.4303e-005  2.4303e-005
  11  9.5692e-015  -----------
==============================
>> x

x =

 -9.5692e-015
```

# Quasi-Newton Methods

# Quadratic subproblems

Methods for unconstrained optimization are based on quadratic subproblems:

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

$H_k = I$ corresponds to steepest descent and $H_k = \nabla^2 f(x_k)$ to Newton's method.

- 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.
- (CG is one way to reduce the expense of computing the step, but the issue if whether or not to compute second derivatives remains.)

# Quasi-Newton methods

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.

# Hessian updates

- 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 $\alpha_k$.
- 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.)

## Secant equation

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

$$\begin{aligned} m_{k+1}(0) &= f(x_{k+1}); \\ \nabla m_{k+1}(0) &= \nabla f(x_{k+1}); \\ \nabla m_{k+1}(-\alpha_k d_k) &= \nabla f(x_k). \end{aligned}$$

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

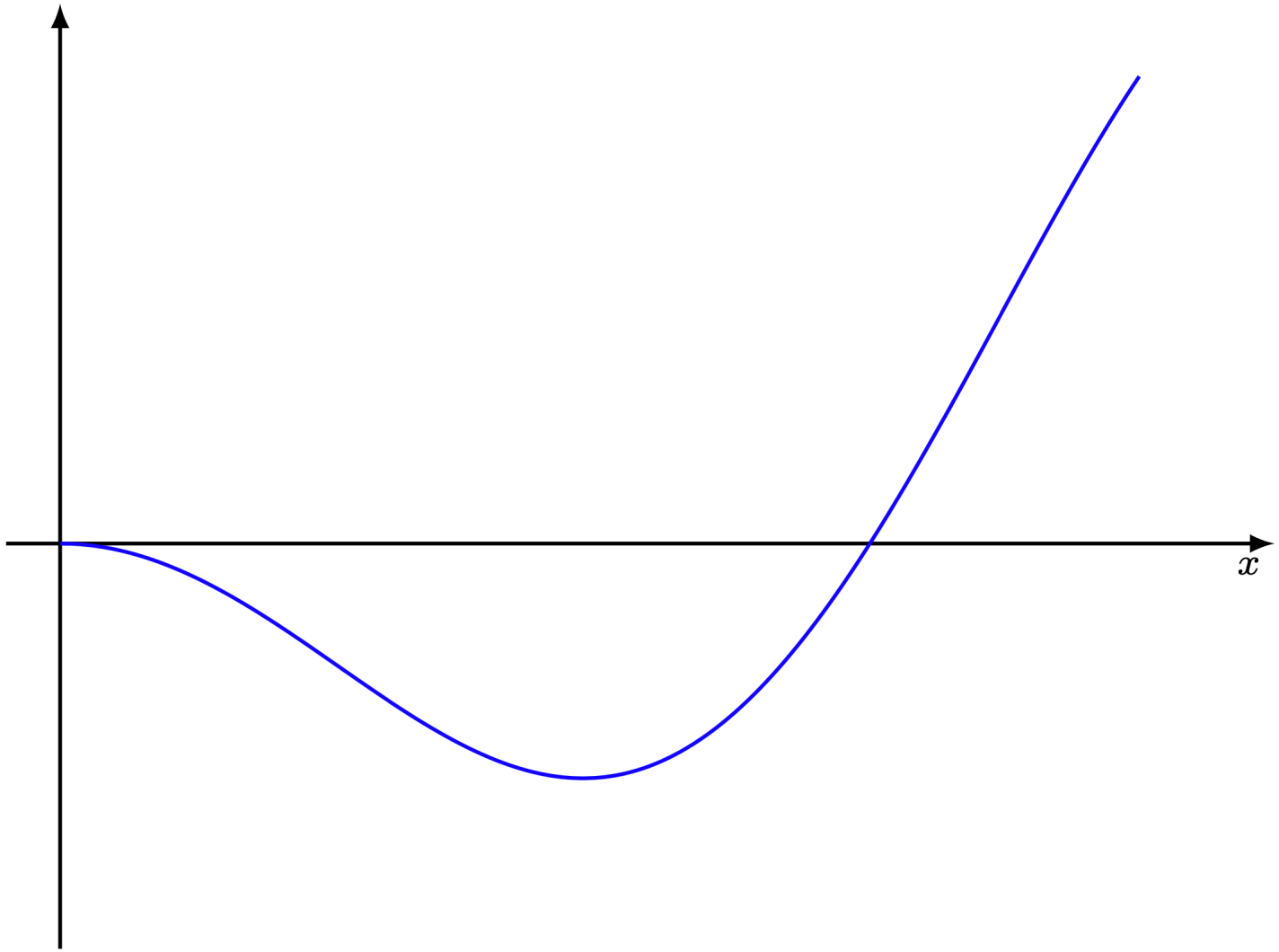
$$H_{k+1} s_k = y_k \text{ (the "secant equation")},$$

where

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

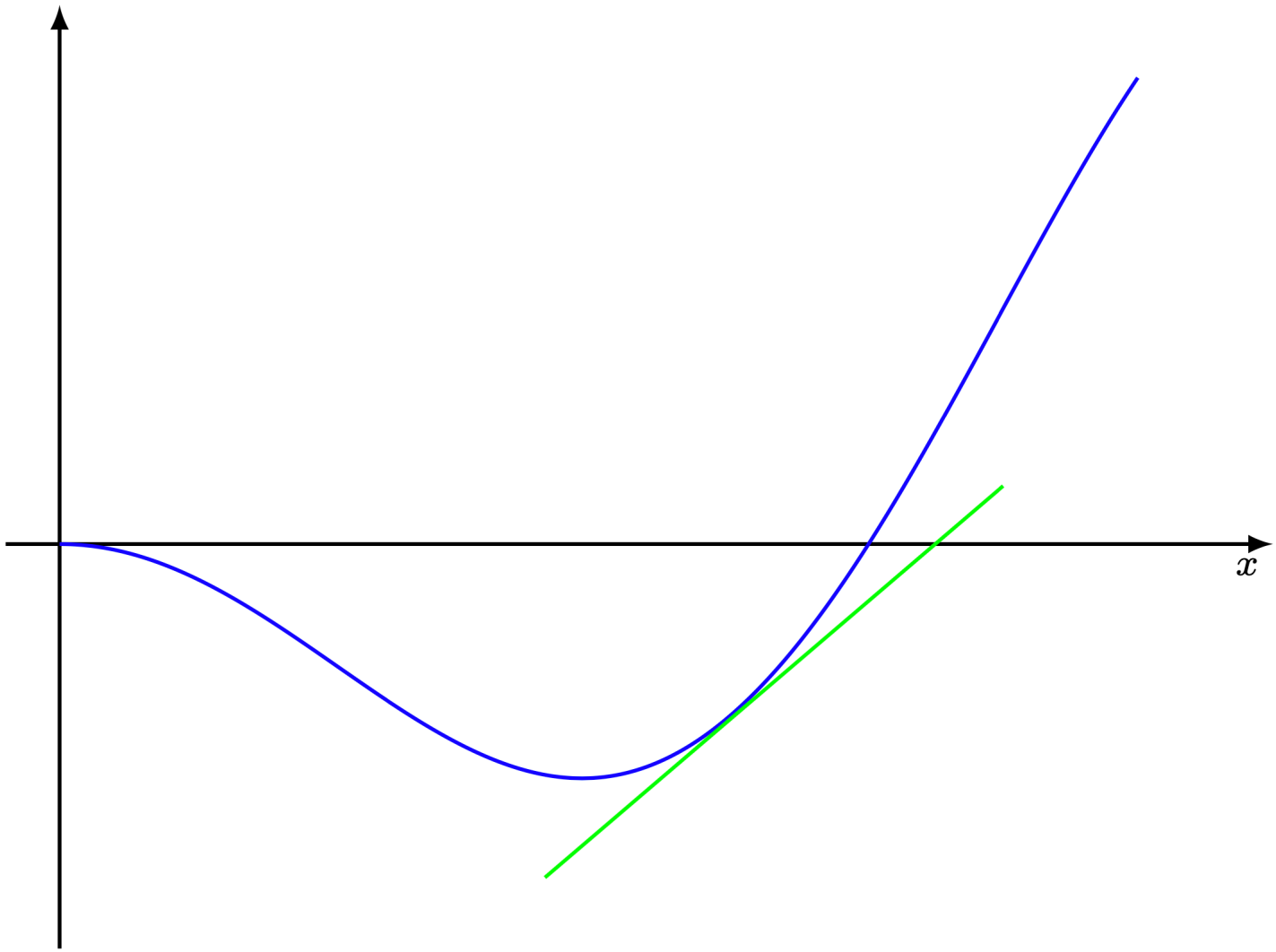
# Example

Consider the following function:



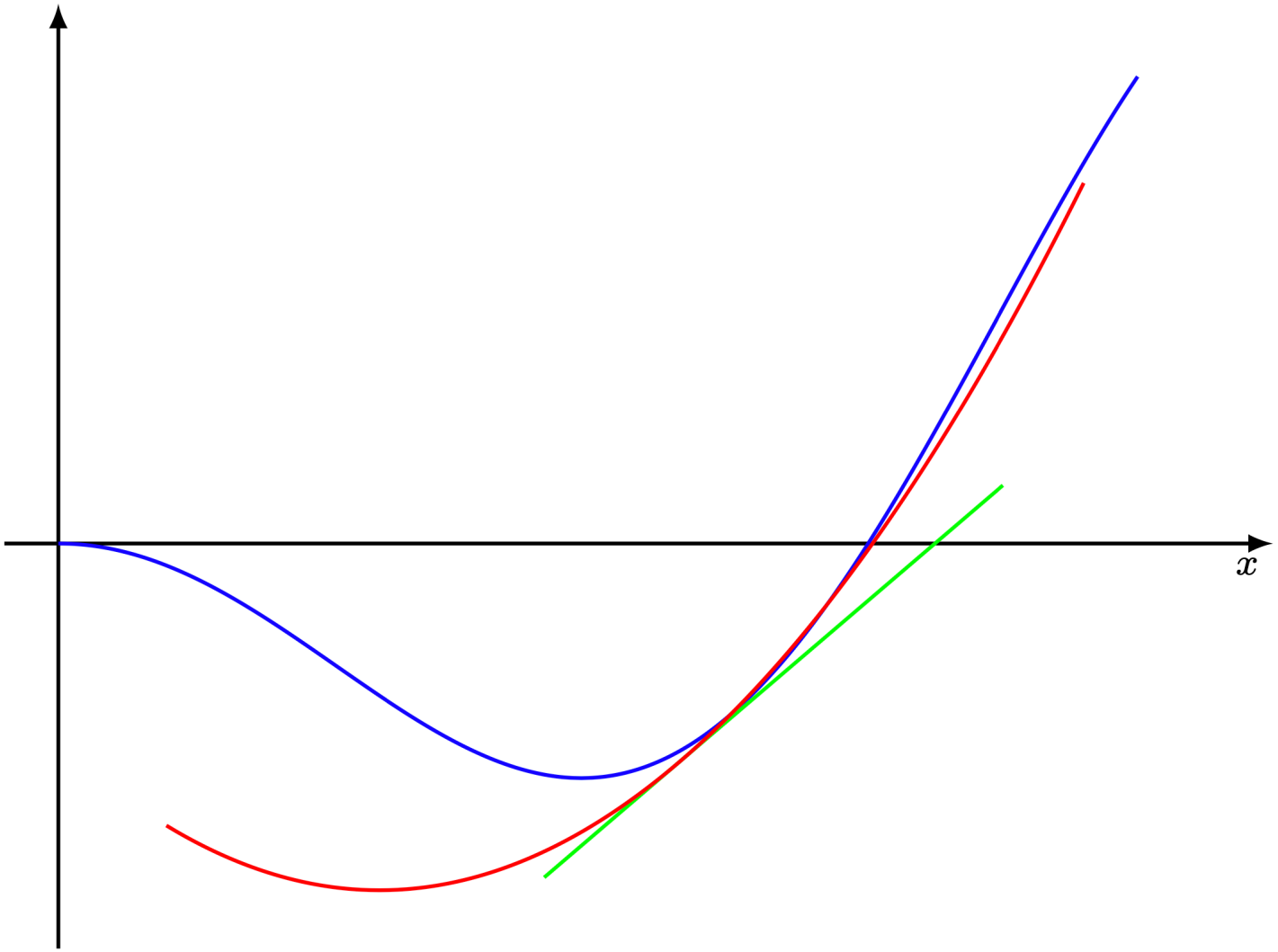
# Example: Linear model at $x_k$

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



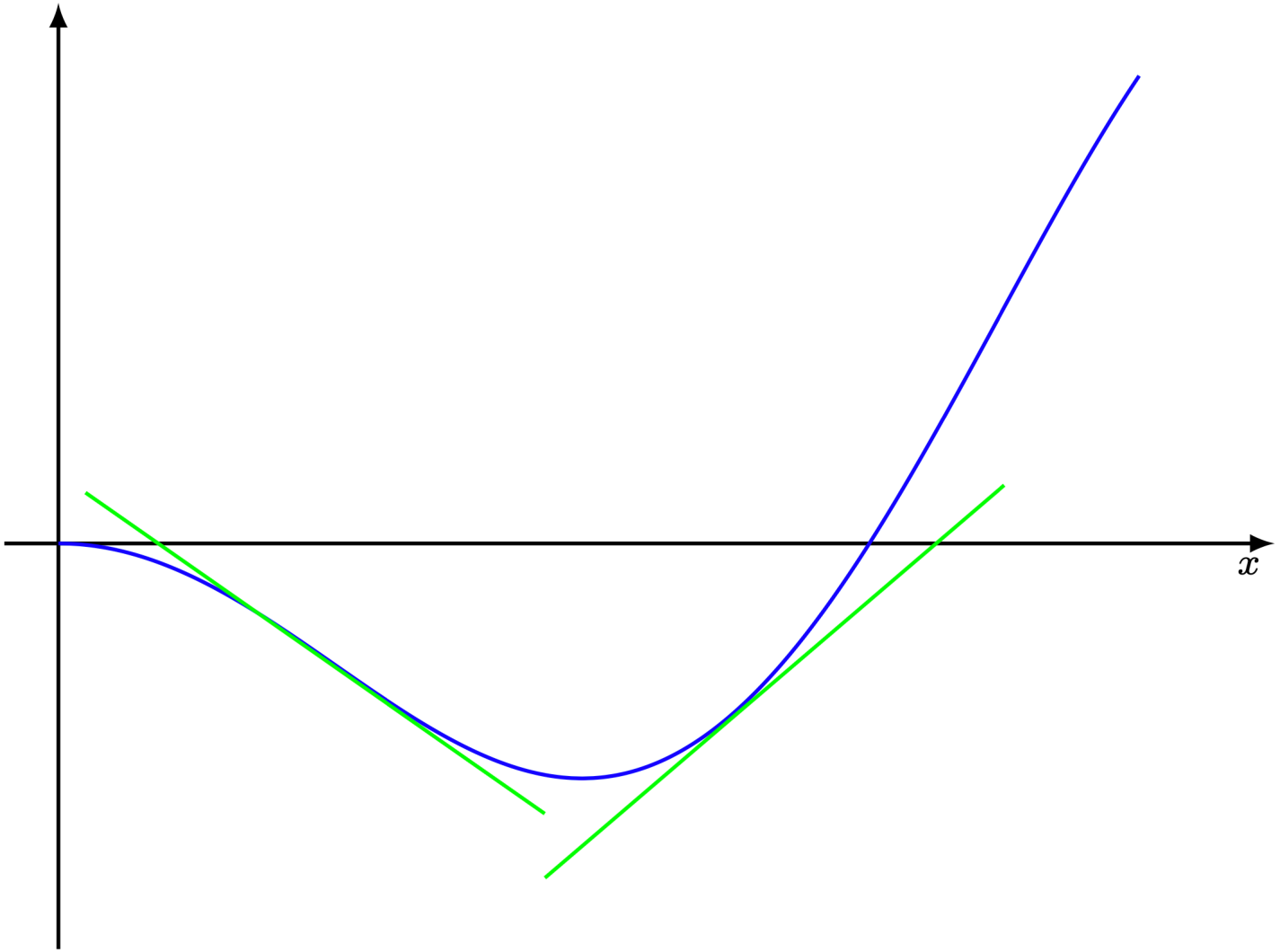
# Example: Quadratic model at $x_k$

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



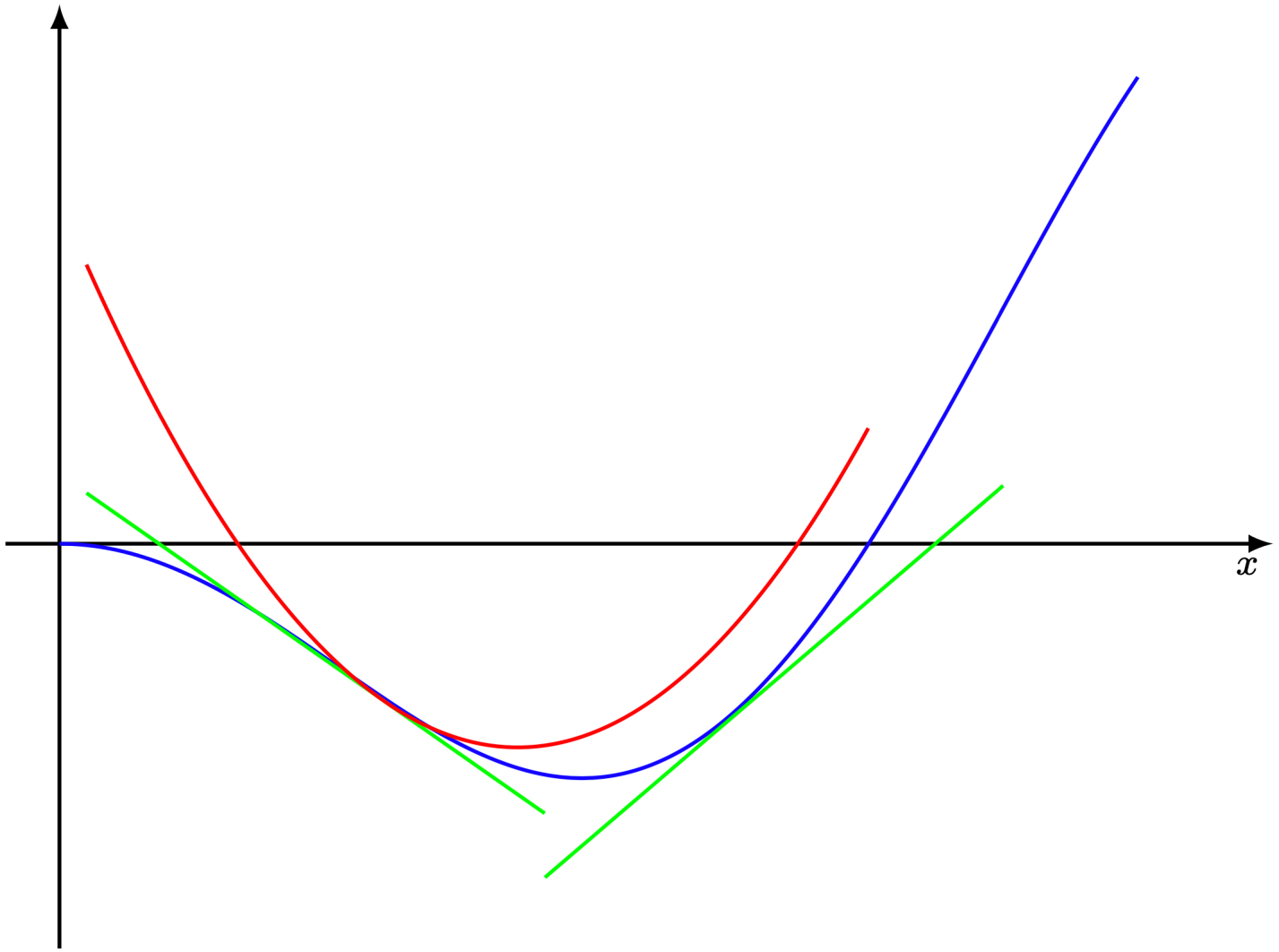
# Example: Information available at $x_{k+1}$

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:



# Quasi-Newton variations

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.

# Symmetric-rank-1 updates

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

$$\begin{aligned} H_{k+1}s_k &= y_k && \text{(secant equation)} \\ H_{k+1} &= H_k + \sigma vv^T && \text{(rank 1 update);} \end{aligned}$$

That is, the difference between $H_k$ and $H_{k+1}$ is $\sigma vv^T$ (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 $\delta$ 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

$$\sigma = \text{sign}((y_k - H_k s_k)^T s_k), \quad \delta^2 = |(y_k - H_k s_k)^T s_k|^{-1}.$$

## SR1 update

(**Sherman-Morrison公式**)假设$A \in \mathbb{R}^{n\times n}$为可逆矩阵，$u, v \in \mathbb{R}^n$为列向量，则$A + uv^T$可逆当且仅当$1 + v^T A^{-1} u \neq 0$, 且当$A + uv^T$可逆时，该逆矩阵由以下公式给出：

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}.$$

- 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

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

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.

# Observations of SR1 updating

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

  $$|(y_k - H_k s_k)^T s_k| \geq c\|y_k - H_k s_k\|\|s_k\|$$

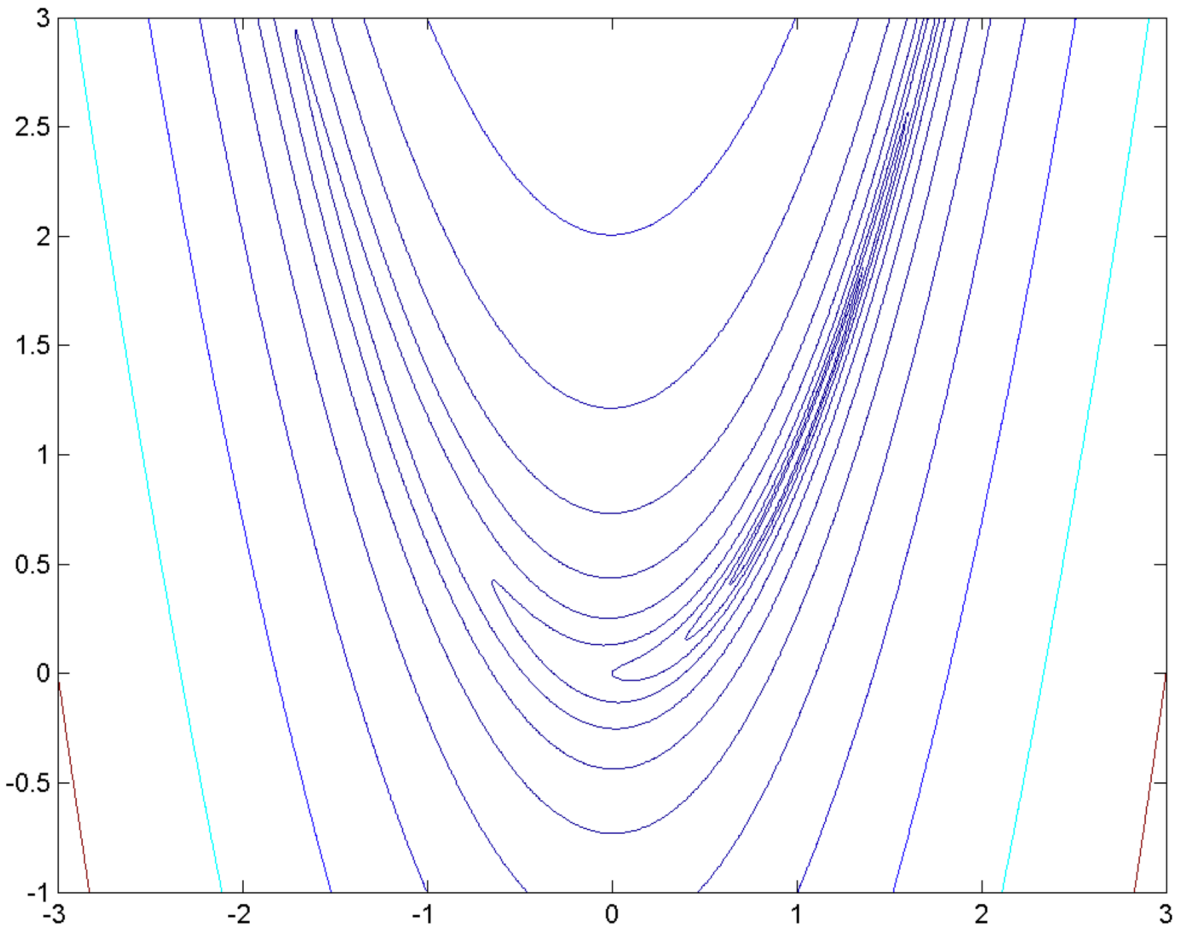
  where $c \in (0, 1)$ is a user-specified constant.
- Even if $H_k \succ 0$, there is no guarantee that the update will yield

  $$H_{k+1} \succ 0.$$

  Thus, we may either have to modify the matrix or use a trust region approach.

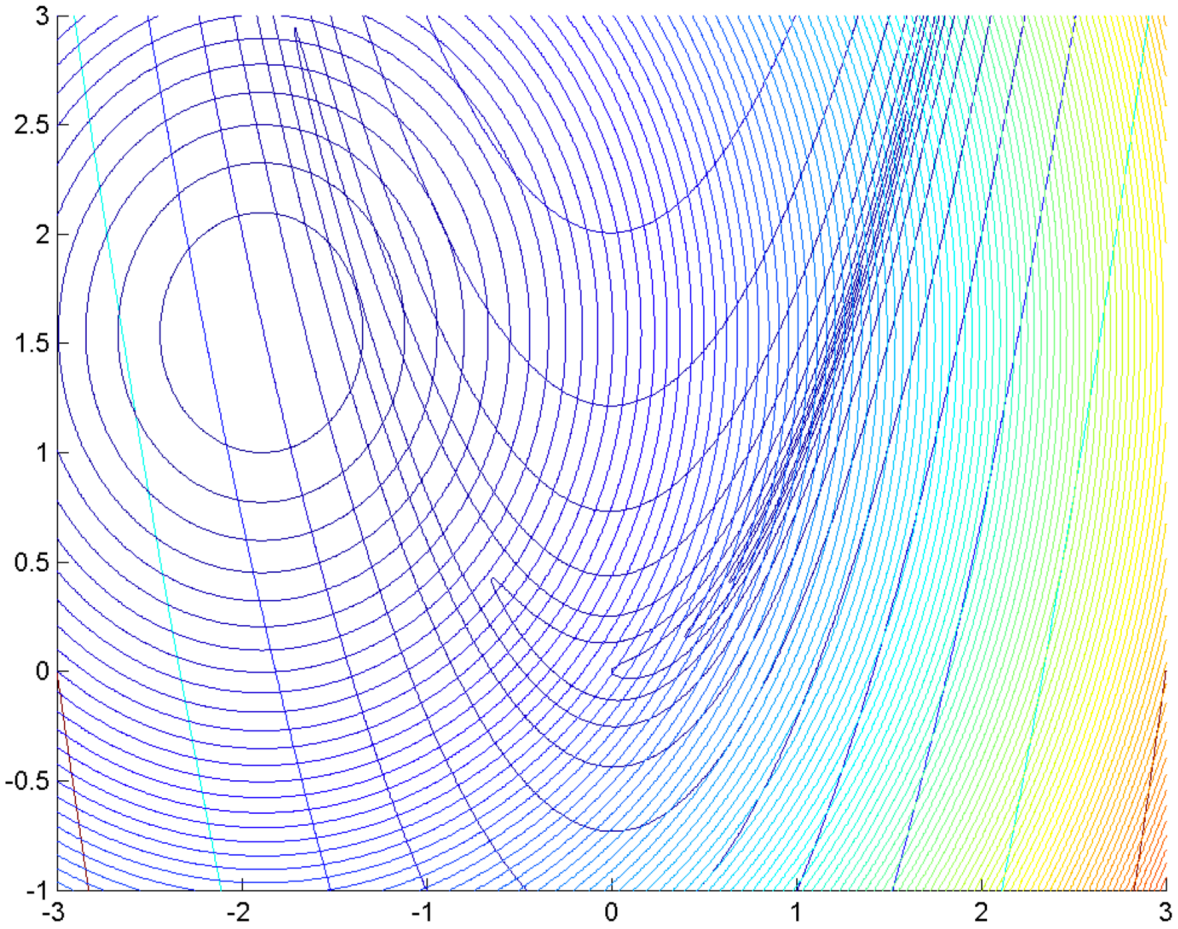
# Example

Recall the Rosenbrock function:



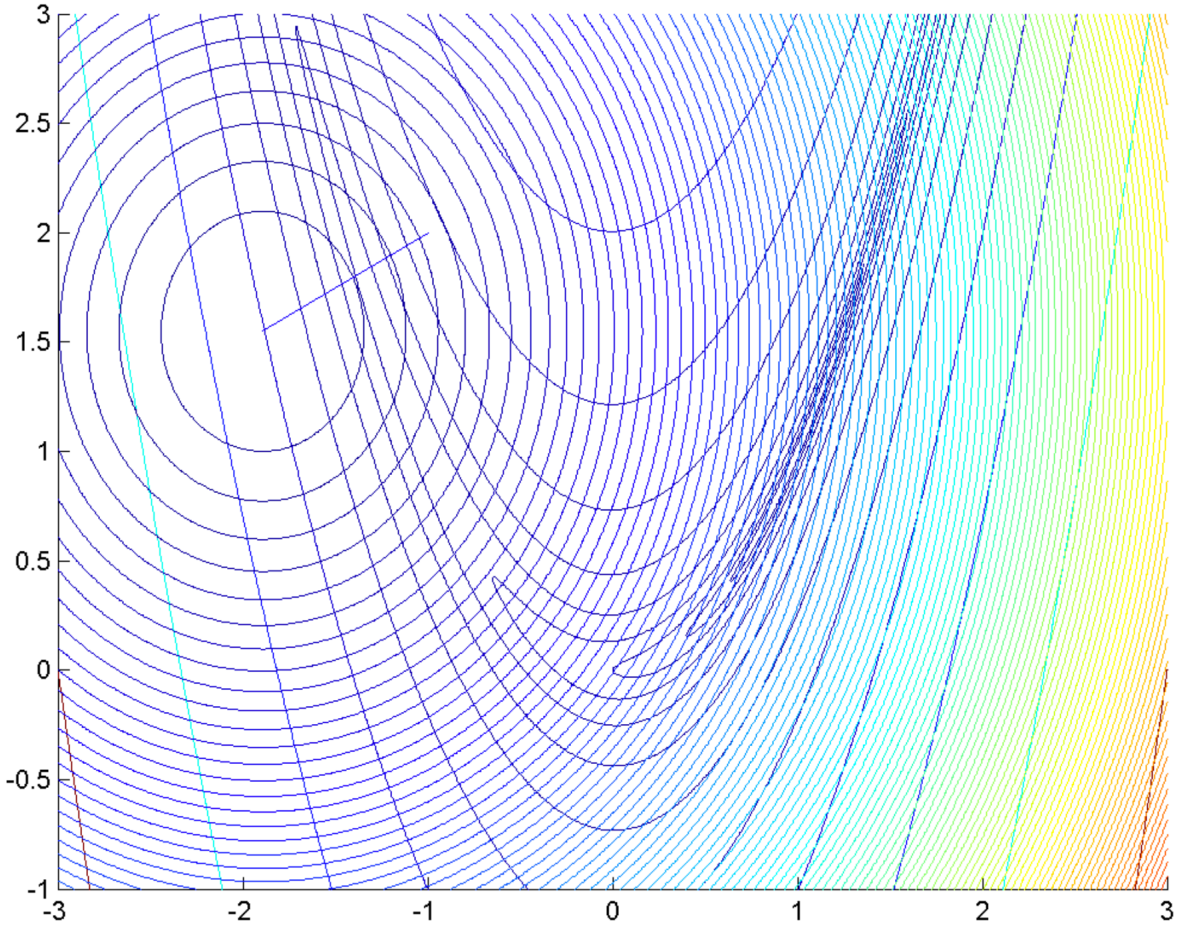
# Example: Initial quadratic model

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



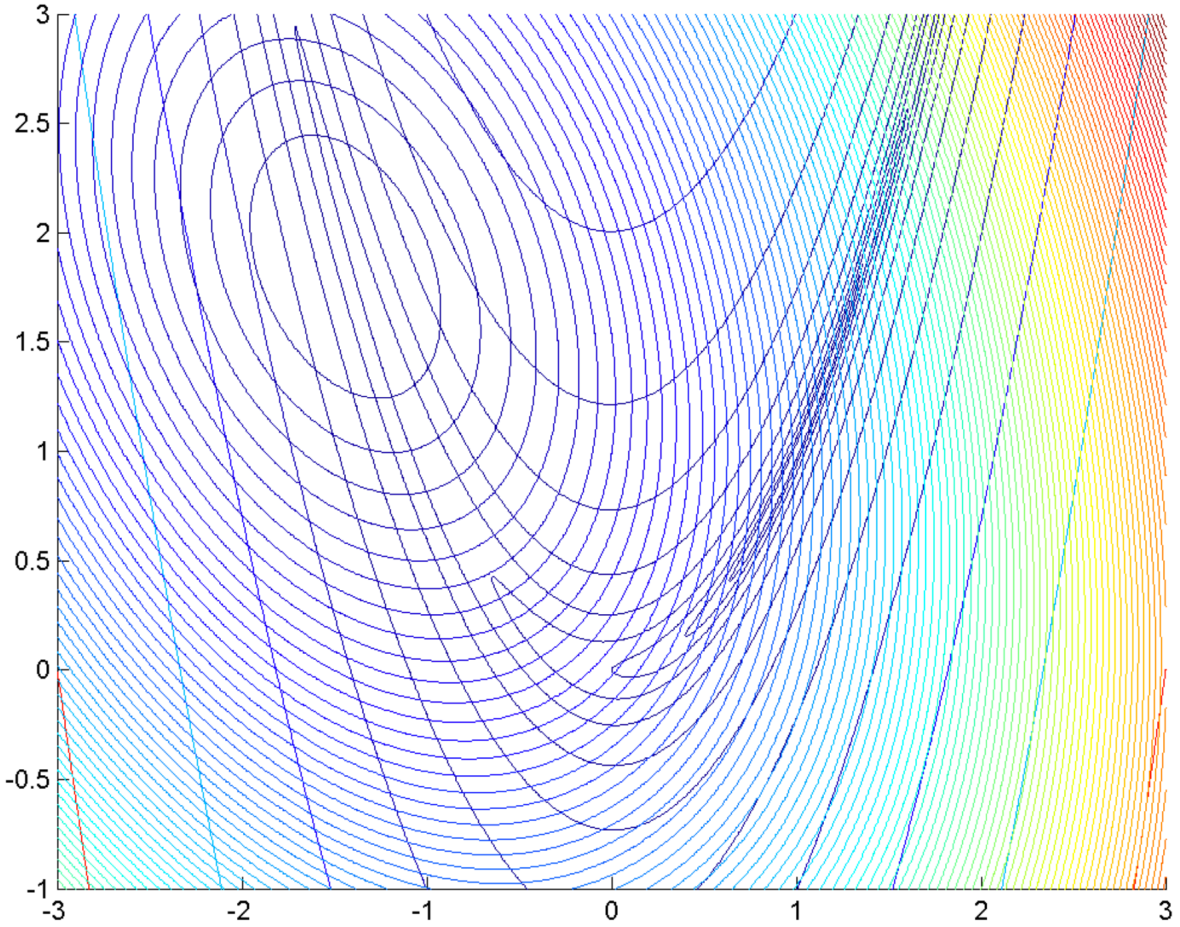
# Example

The step is obtained by minimizing the quadratic model:



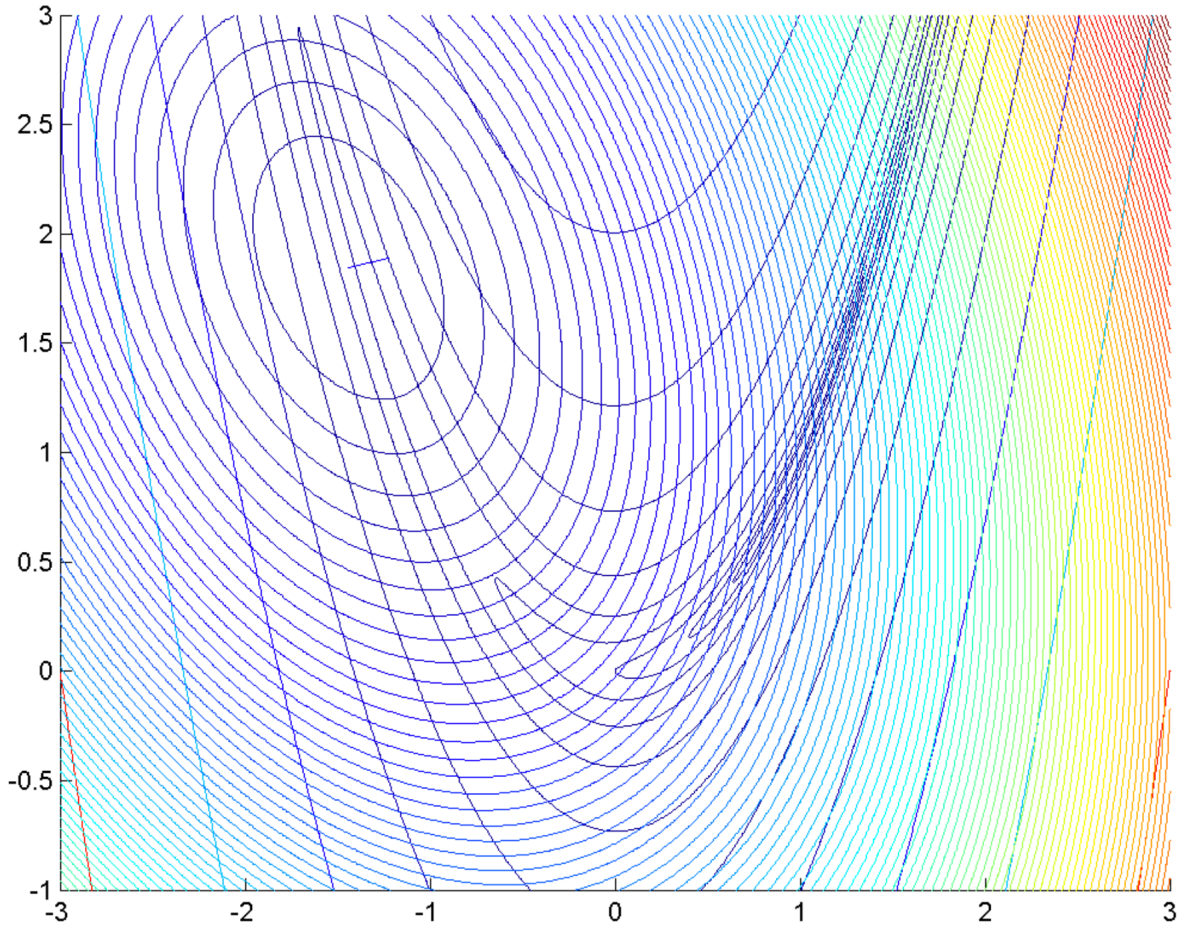
# Example

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



# Example

The step is again obtained by minimizing the quadratic model:



# Motivation

- 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:

$$\min_{H} \|H - H_k\|$$
$$\text{s.t. } H = H^T, Hs_k = y_k.$$

- 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.

# Davidon-Fletcher-Powell update

- 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）：

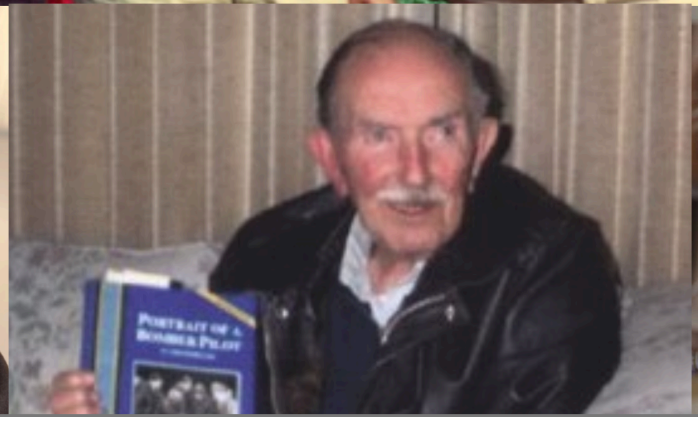
$$(\mathbf{D} + \mathbf{V}\mathbf{V}^T)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^T\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^T\mathbf{D}^{-1}$$

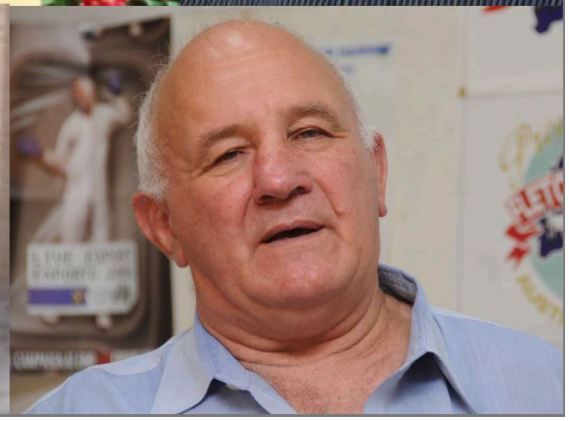
# Broyden-Fletcher-Goldfarb-Shanno update

- 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.,

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

## Broyden, Fletcher, Goldfarb, Shanno





# Broyden-Fletcher-Goldfarb-Shanno update

- 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.

# Positive definiteness

- 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?

# Benefit of a Wolfe line search

Using a Wolfe line search, we ensure that

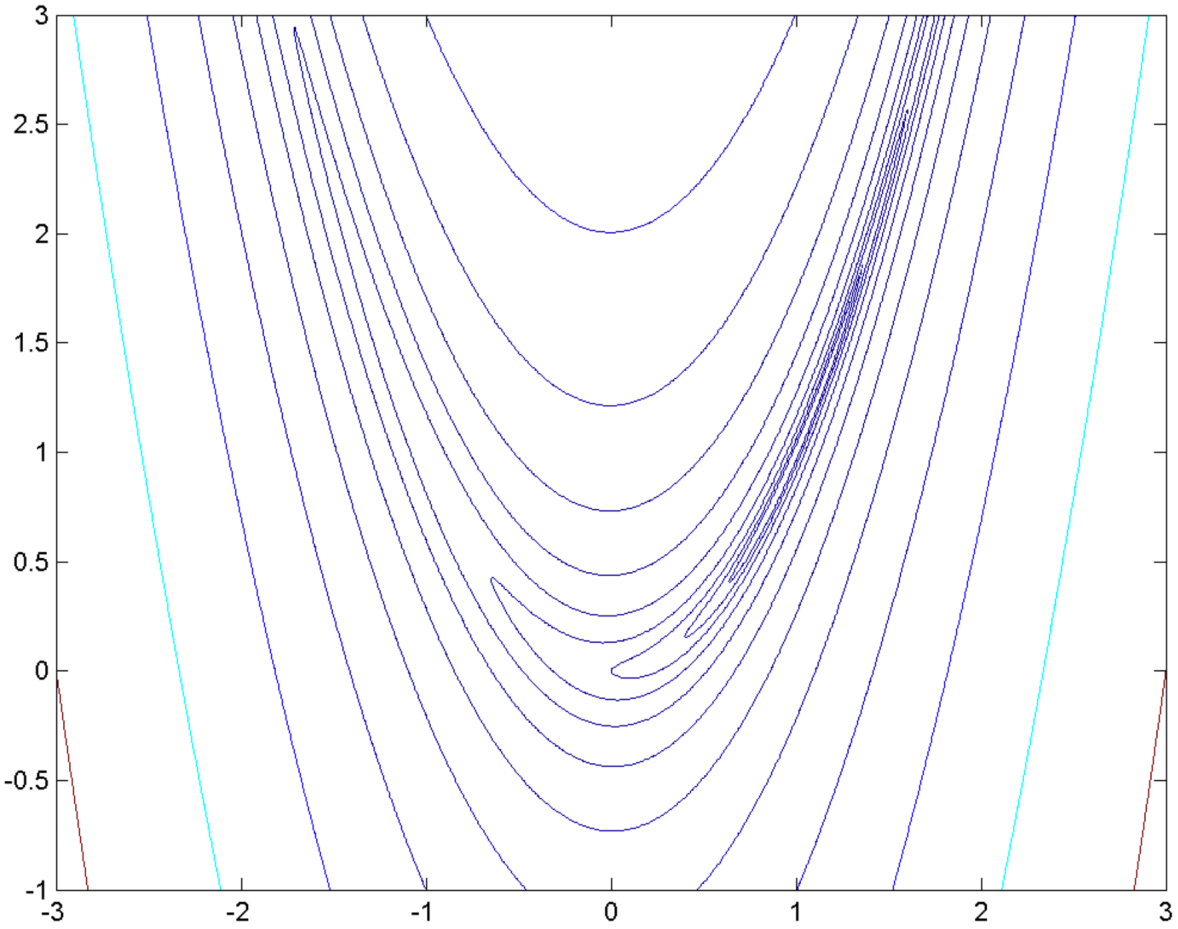
$$\nabla f(x_{k+1})^T d_k \geq c_2 \nabla f(x_k)^T d_k.$$

Thus, (assuming $s_k \neq 0$)

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

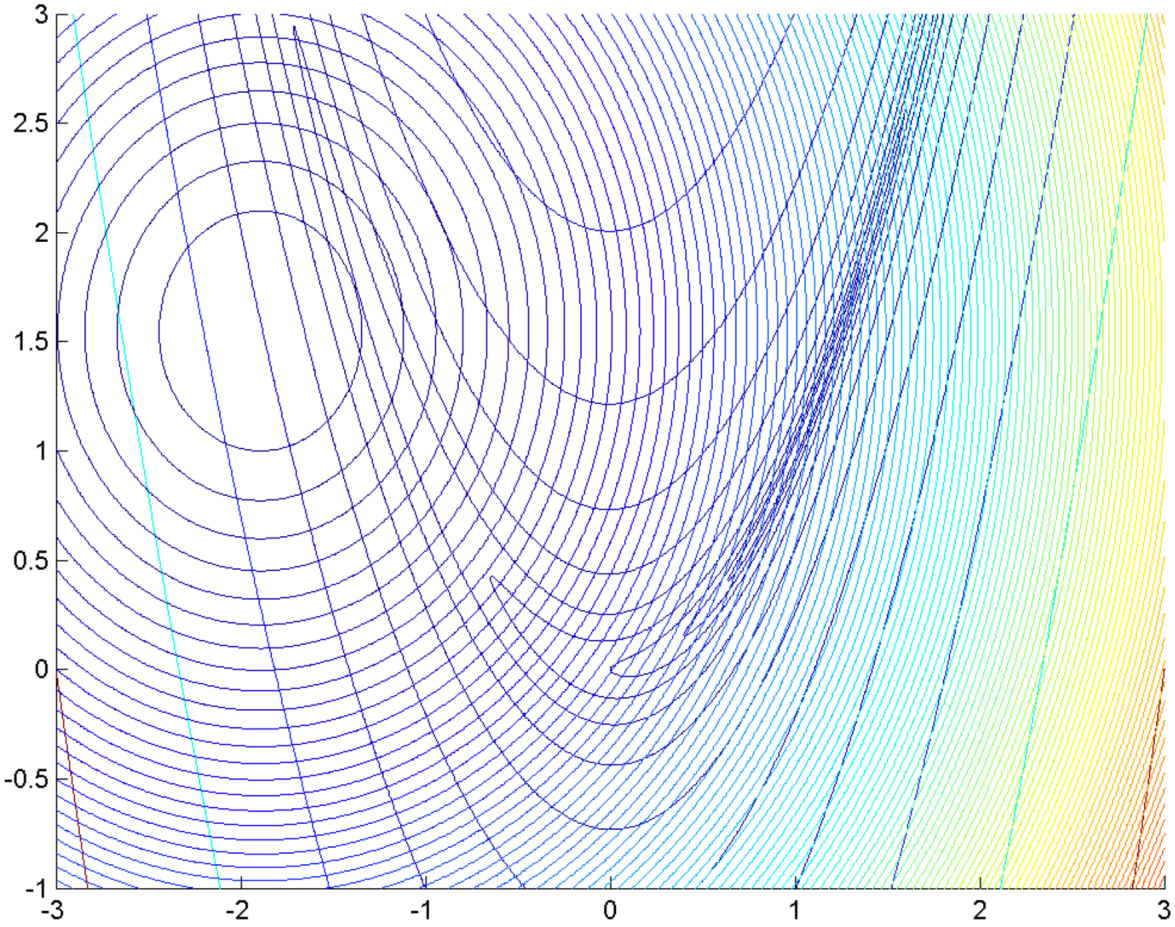
# Example

Recall the Rosenbrock function:



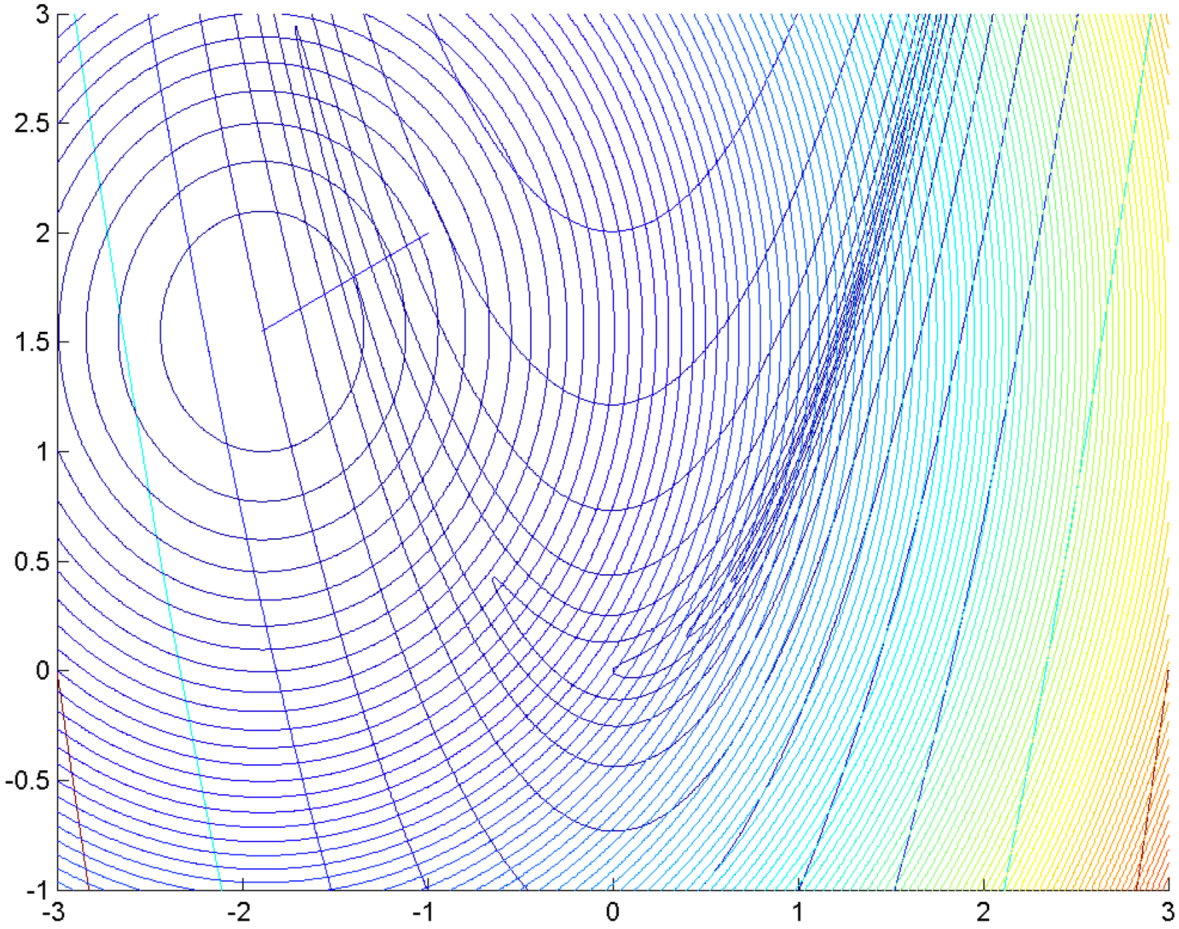
# Example: Initial quadratic model

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



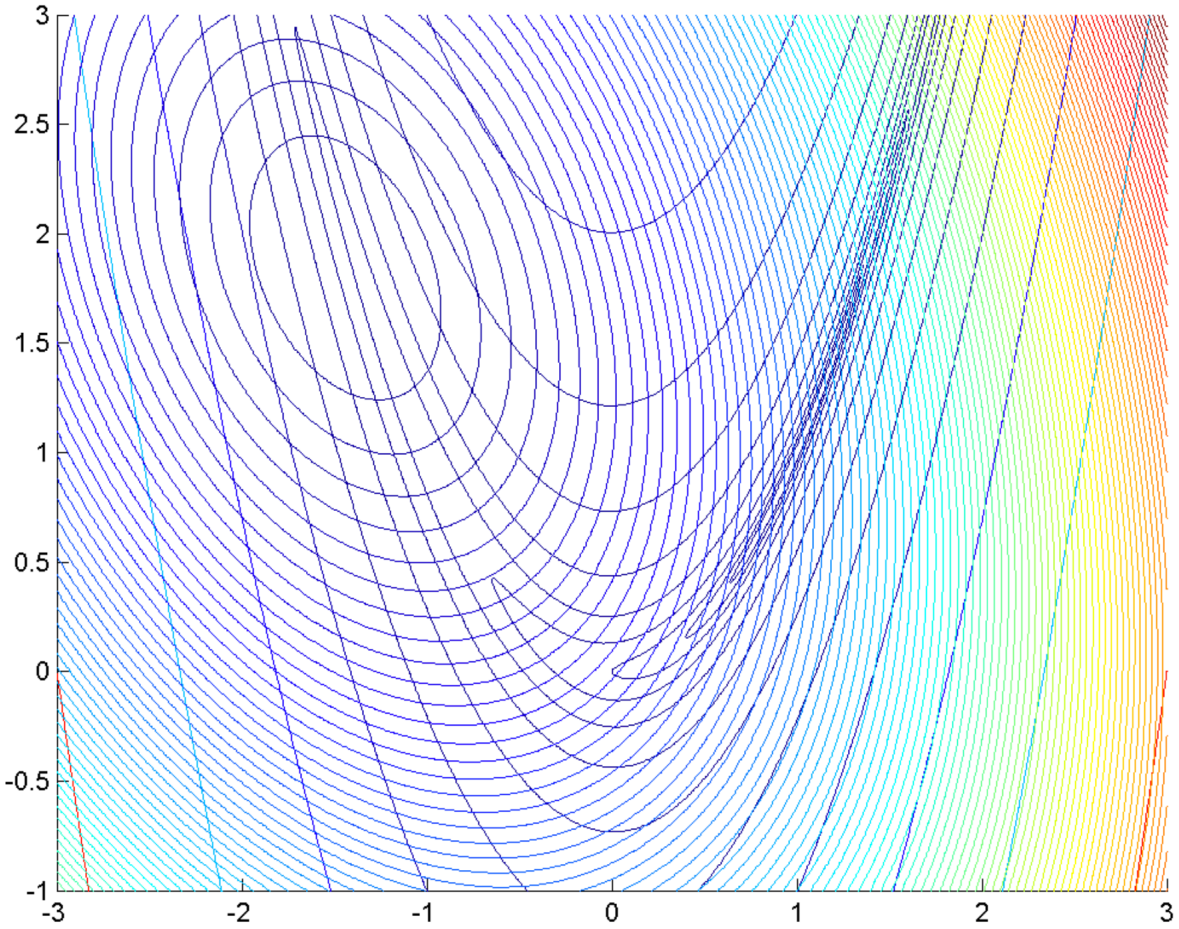
# Example

The step is obtained by minimizing the quadratic model:



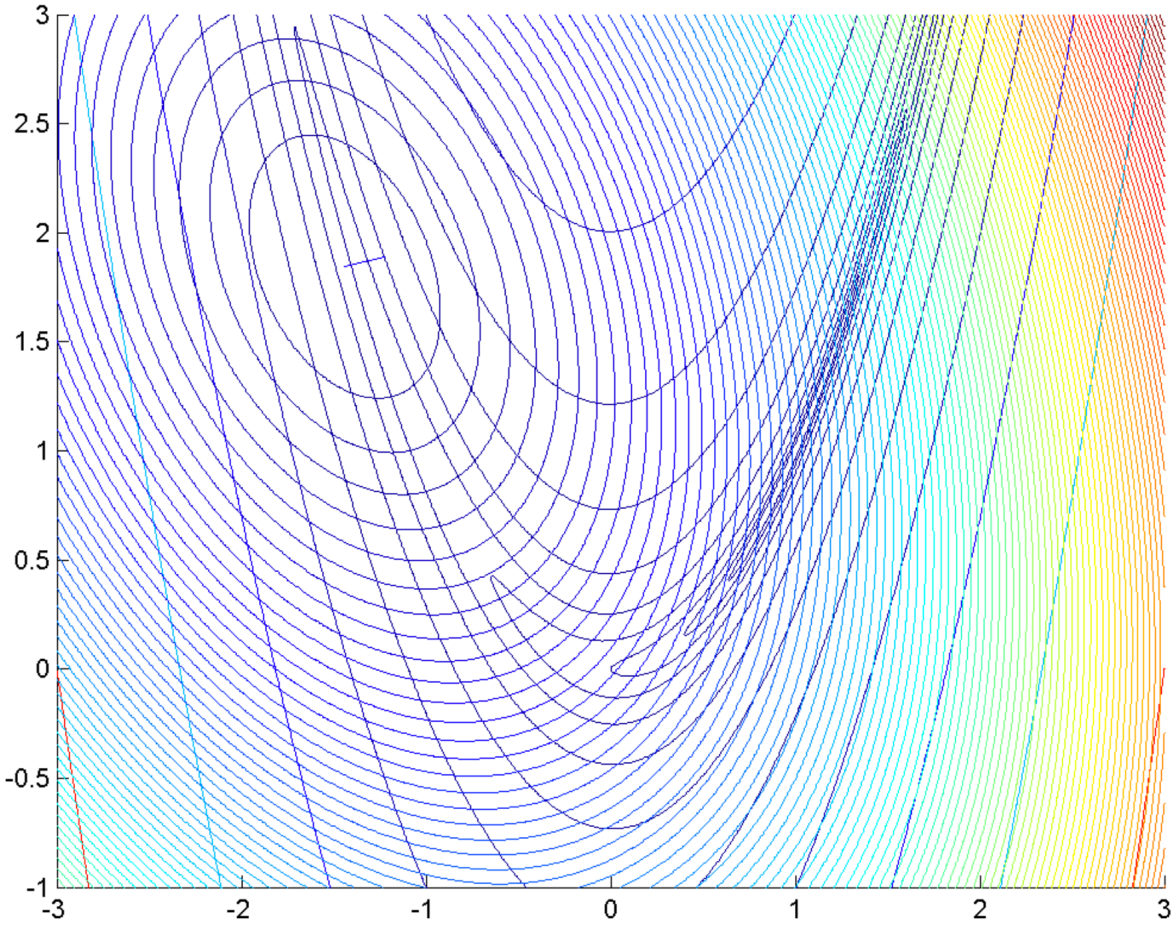
# Example

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



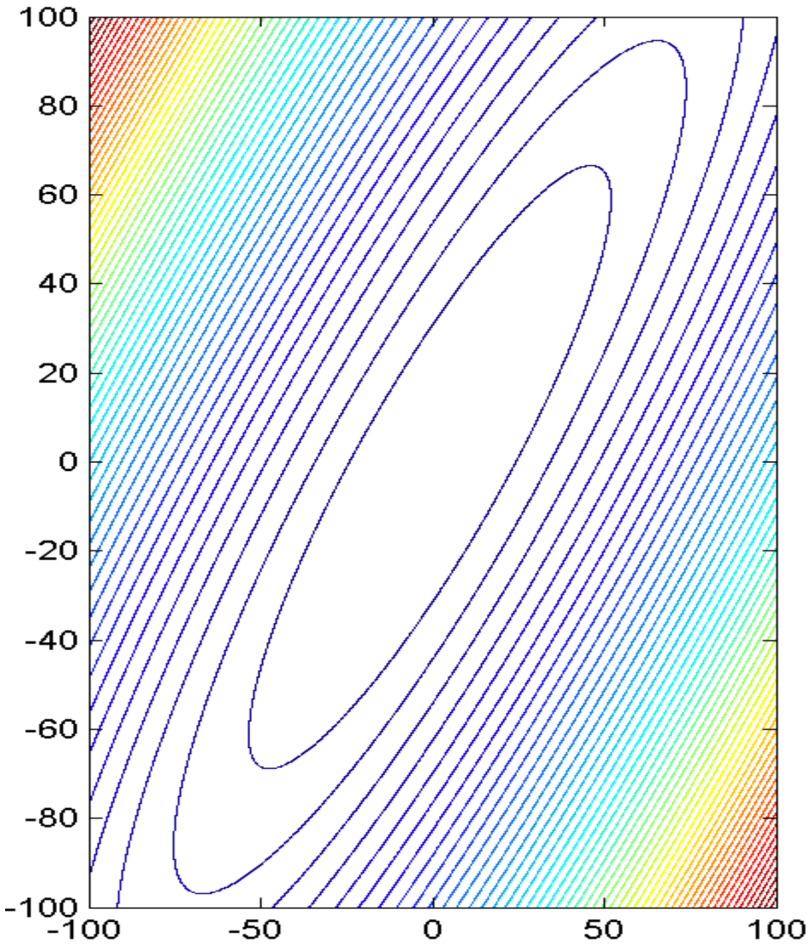
# Example

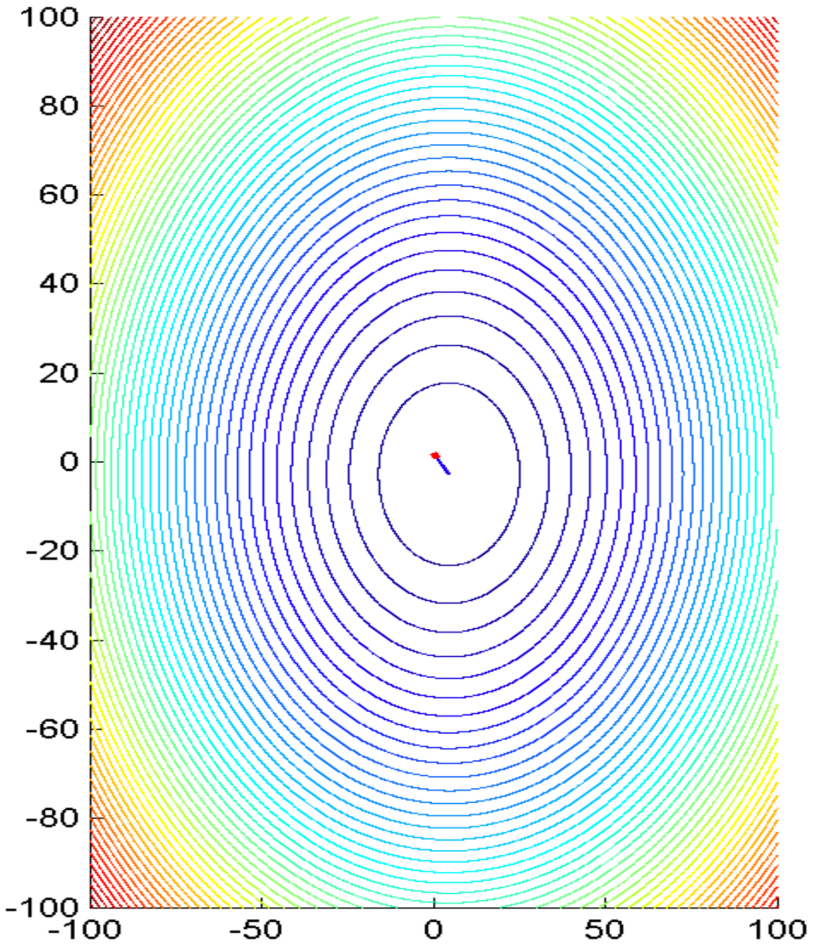
The step is again obtained by minimizing the quadratic model:



# 2D example: Initialization

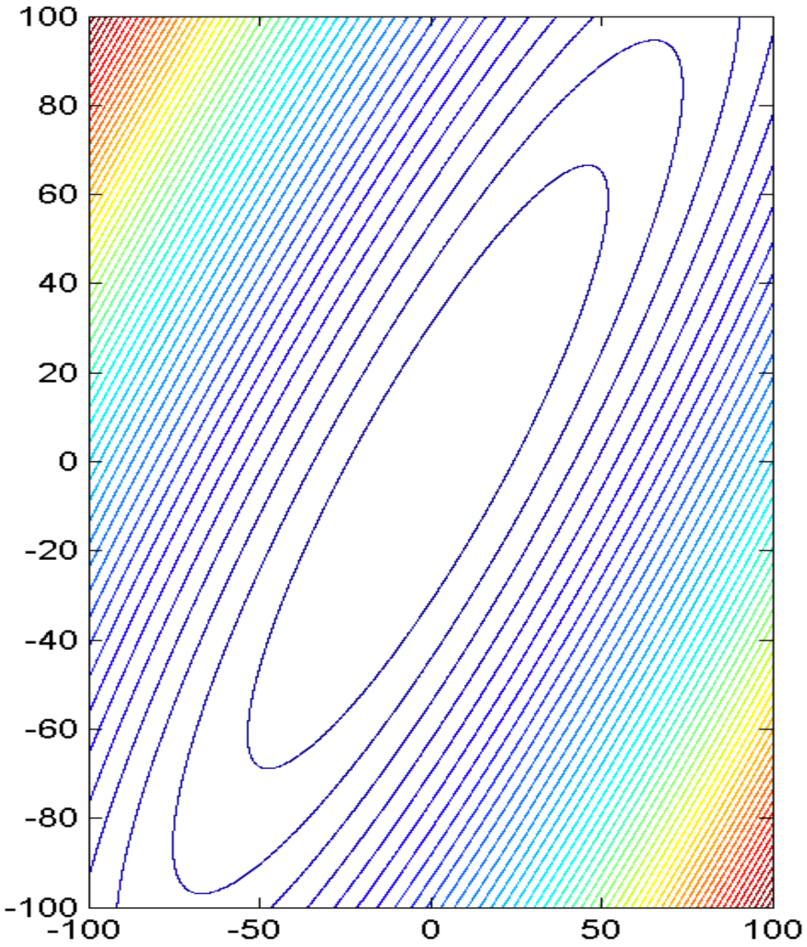
True function plotted on left; model function and step plotted on right:

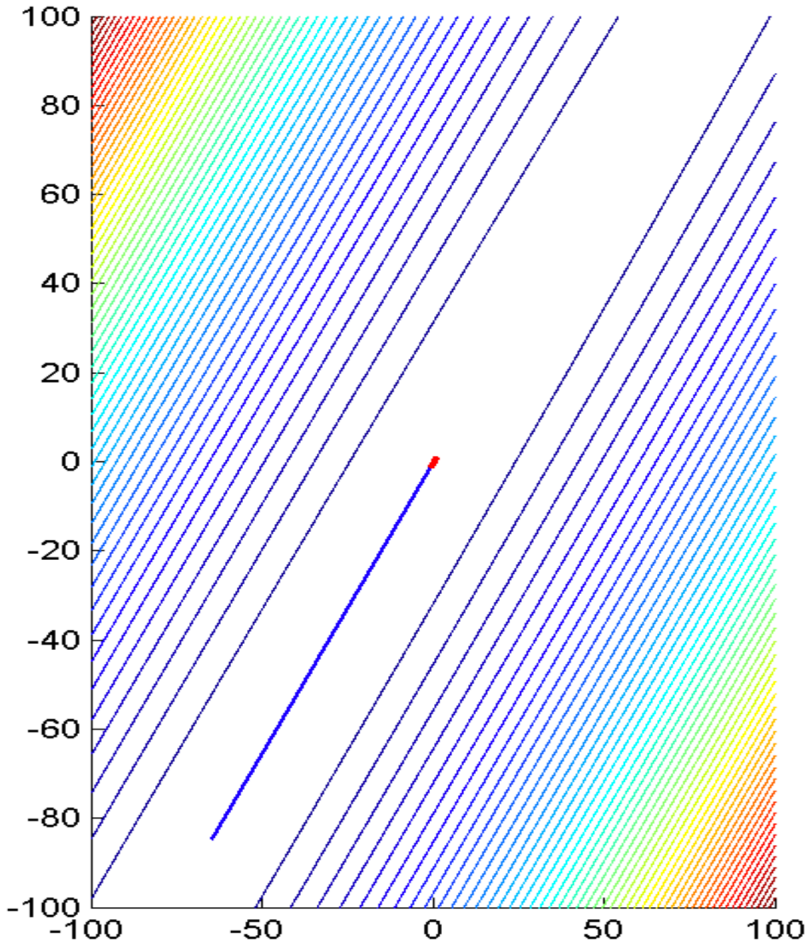


# 2D example: After one iteration

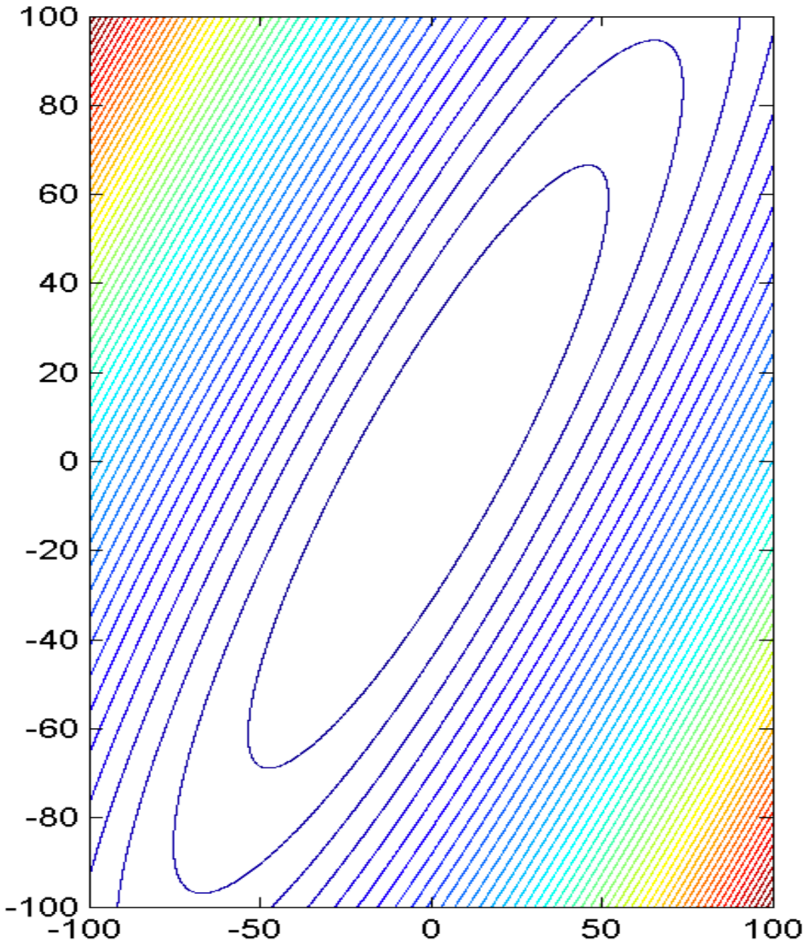
True function plotted on left; model function and step plotted on right:

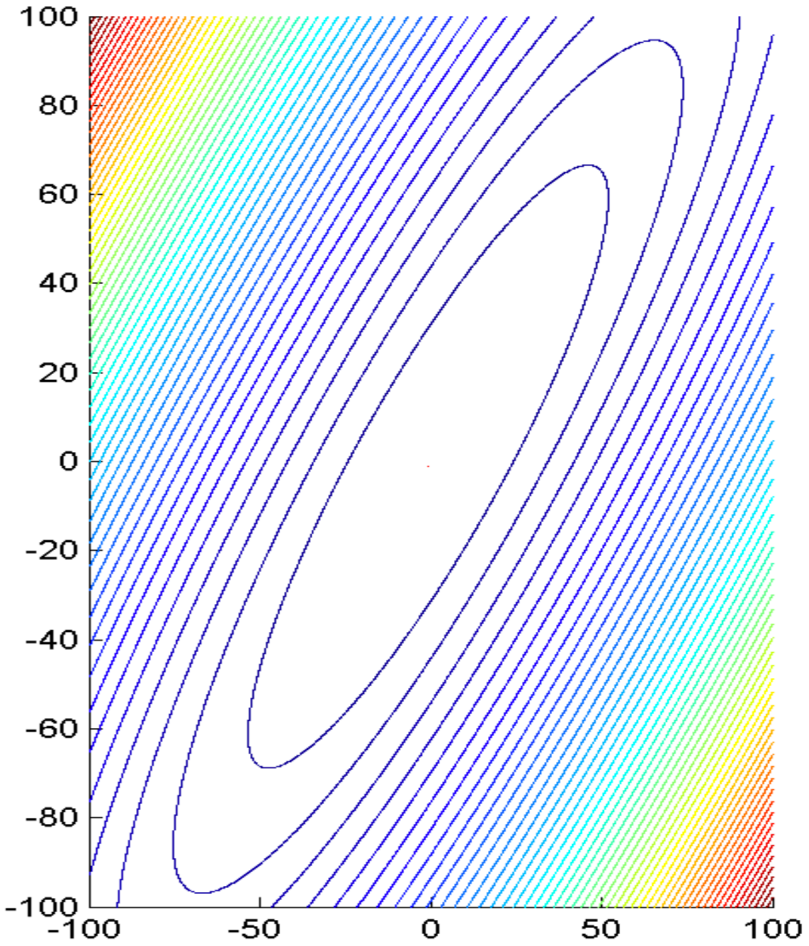


# 2D example: After two iterations

True function plotted on left; model function and step plotted on right:



# Global convergence of BFGS method w/ Wolfe line search

## Theorem

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

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

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

*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 $f$.*

## Sketch of proof.

Investigate angle between $d_k$ and $-\nabla f(x_k)$ and use Zoutendijk.
Indeed, we find that this angle satisfies

$$\cos\theta_k = \frac{s_k^T H_k s_k}{\|s_k\|\|H_k s_k\|}.$$

# Superlinear convergence of BFGS method

## Theorem

*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.*

## Sketch of proof.

The central part of the result is to show the limit

$$\lim_{k \to \infty} \frac{(H_k - \nabla^2 f(x_*))s_k}{\|s_k\|} = 0.$$

In the limit, $\{H_k\}$ sufficiently approximate $\nabla^2 f(x_*)$ in the directions $\{s_k\}$.

# Dense approximations

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:

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

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

# Limited-memory approximations

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).

# Limited-memory approximations

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).