Unconstrained Optimization Algorithms

Descent Direction

Second-Order Methods

Newton's Method

Consider an objective $f : \mathbb{R}^n \to \mathbb{R}$.

Assume that f is twice differentiable.

Newton's Method

Consider an objective $f : \mathbb{R}^n \to \mathbb{R}$.

Assume that f is twice differentiable.

Then, by the Taylor's theorem,

$$f(x_k + s) \approx f_k + \nabla f_k^\top s + \frac{1}{2} s^\top H_k s$$

where we denote the Hessian $\nabla^2 f(x_k)$ by H_k.

Newton's Method

Consider an objective $f : \mathbb{R}^n \to \mathbb{R}$.

Assume that f is twice differentiable.

Then, by the Taylor's theorem,

$$f\left(x_k + s\right) \approx f_k + \nabla f_k^\top s + \frac{1}{2} s^\top H_k s$$

where we denote the Hessian $\nabla^2 f(x_k)$ by H_k.

Define

$$q(s) = f_k + \nabla f_k^\top s + \frac{1}{2} s^\top H_k s$$

and minimize q w.r.t. s by setting $\nabla q(s) = 0$.

Newton's Method

Consider an objective $f : \mathbb{R}^n \to \mathbb{R}$.

Assume that f is twice differentiable.

Then, by the Taylor's theorem,

$$f(x_k + s) \approx f_k + \nabla f_k^\top s + \frac{1}{2} s^\top H_k s$$

where we denote the Hessian $\nabla^2 f(x_k)$ by H_k.

Define

$$q(s) = f_k + \nabla f_k^\top s + \frac{1}{2} s^\top H_k s$$

and minimize q w.r.t. s by setting $\nabla q(s) = 0$. We obtain:

$$H_k s = -\nabla f_k$$

Denote by s_k the solution, and set $x_{k+1} = x_k + s_k$.

Newton's Method

Algorithm 1 Newton's Method

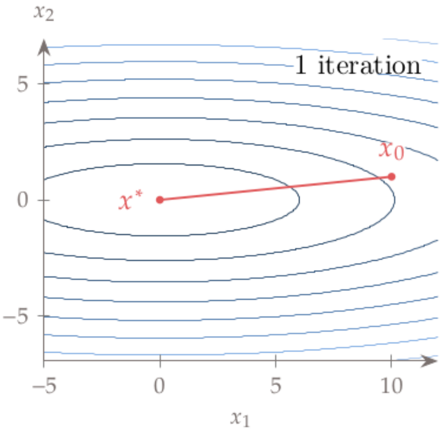
Input: x_0 starting point, $\varepsilon > 0$

Output: x^* approximation to a stationary point

```
1: k ← 0
2: while ‖∇f_k‖_∞ > ε do
3:     p_k ← −H_k^{−1} ∇f(x_k)
4:     x_{k+1} ← x_k + p_k
5:     k ← k + 1
6: end while
```

Newton's Method - Example

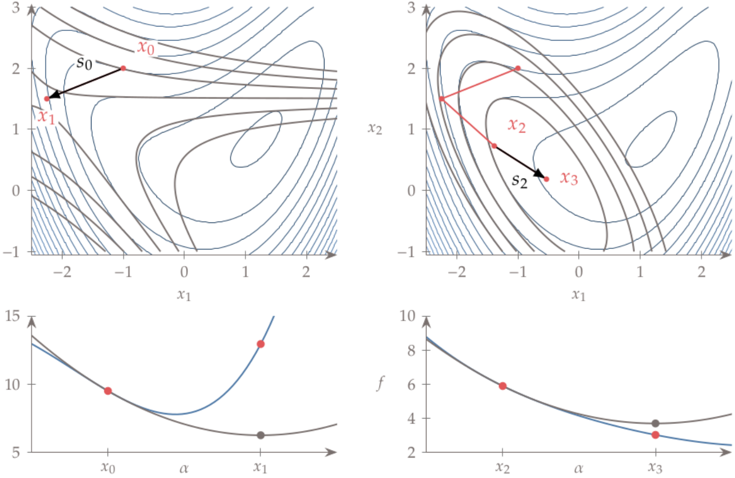
Newton's method finds the minimum of a quadratic function in a single step.



Note that the Newton's method is scale-invariant!

$$f(x_1, x_2) = (1 - x_1)^2 + (1 - x_2)^2 + \frac{1}{2}\left(2x_2 - x_1^2\right)^2$$

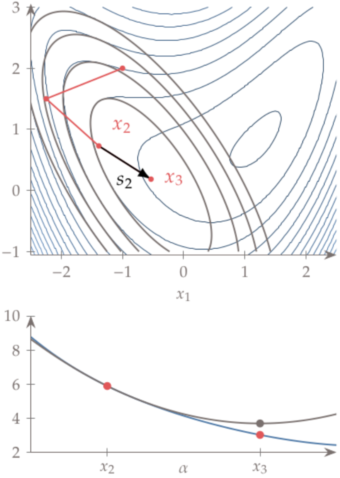
Stopping: $||\nabla f||_\infty \leq 10^{-6}$.



$k = 0$ $k = 2$

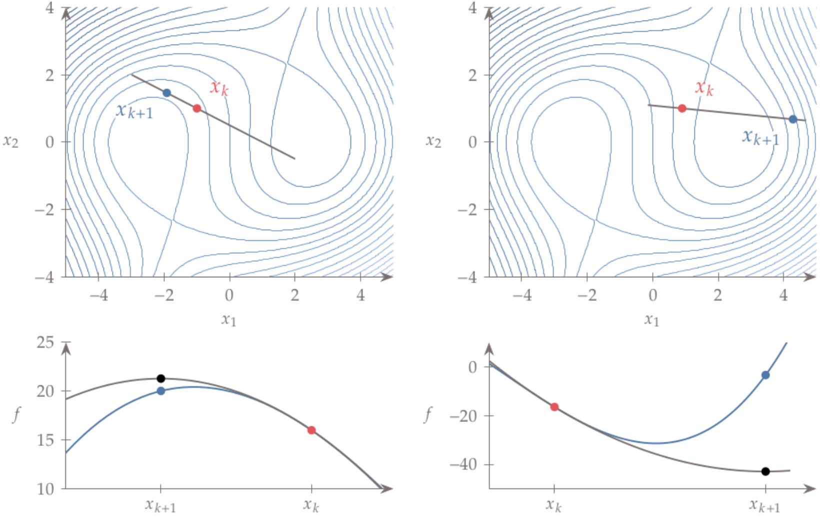
$$f(x_1, x_2) = (1 - x_1)^2 + (1 - x_2)^2 + \frac{1}{2}\left(2x_2 - x_1^2\right)^2$$

Stopping: $||\nabla f||_\infty \leq 10^{-6}$.



$k = 2$ $k = 8$

Convergence Issues



Negative curvature

Overshoot

Also, the computation of the Hessian is costly.

Local Quadratic Convergence of Newton's Method

Theorem 1

Assume f is defined and twice differentiable and assume that ∇f is L-smooth on $\mathcal{N}$.

Let x_* be a minimizer of $f(x)$ in $\mathcal{N}$ and assume that $\nabla^2 f(x_*)$ is positive definite.

If $\|x_0 - x_*\|$ is sufficiently small, then $\{x_k\}$ converges quadratically to x_*.

Local Quadratic Convergence of Newton's Method

Theorem 1

Assume f is defined and twice differentiable and assume that ∇f is L-smooth on $\mathcal{N}$.
Let x_* be a minimizer of $f(x)$ in $\mathcal{N}$ and assume that $\nabla^2 f(x_*)$ is positive definite.
If $\|x_0 - x_*\|$ is sufficiently small, then $\{x_k\}$ converges quadratically to x_*.

Note that the theorem implicitly assumes that $\nabla^2 f(x_k)$ is nonsingular for every k.

Local Quadratic Convergence of Newton's Method

Theorem 1

Assume f is defined and twice differentiable and assume that ∇f is L-smooth on $\mathcal{N}$.
Let x_* be a minimizer of $f(x)$ in $\mathcal{N}$ and assume that $\nabla^2 f(x_*)$ is positive definite.
If $\|x_0 - x_*\|$ is sufficiently small, then $\{x_k\}$ converges quadratically to x_*.

Note that the theorem implicitly assumes that $\nabla^2 f(x_k)$ is nonsingular for every k.

As the theorem is concerned only with x_k approaching x^*, the continuity of $\nabla^2 f(x_k)$ and positive definiteness of $\nabla^2 f(x^*)$ imply that $\nabla^2 f(x_k)$ is positive definite for all sufficiently large k.

Local Quadratic Convergence of Newton's Method

Theorem 1

Assume f is defined and twice differentiable and assume that ∇f is L-smooth on $\mathcal{N}$.
Let x_* be a minimizer of $f(x)$ in $\mathcal{N}$ and assume that $\nabla^2 f\,(x_*)$ is positive definite.
If $\|x_0 - x_*\|$ is sufficiently small, then $\{x_k\}$ converges quadratically to x_*.

Note that the theorem implicitly assumes that $\nabla^2 f(x_k)$ is nonsingular for every k.

As the theorem is concerned only with x_k approaching x^*, the continuity of $\nabla^2 f(x_k)$ and positive definiteness of $\nabla^2 f(x^*)$ imply that $\nabla^2 f(x_k)$ is positive definite for all sufficiently large k.

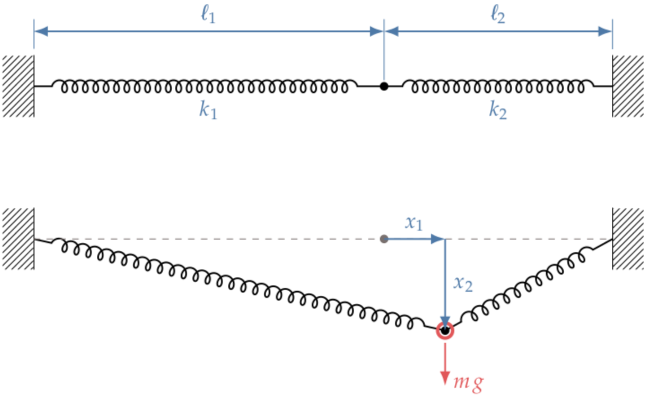
However, what happens if we start far away from a minimizer?

Newton's Method with Line Search

Algorithm 2 Newton's Method with Line Search

Input: x_0 starting point, $\varepsilon > 0$
Output: x^* approximation to a stationary point

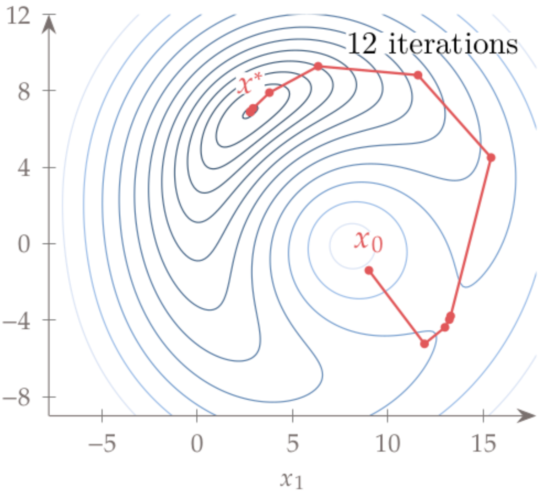
$k \leftarrow 0$
$\alpha_{\text{init}} \leftarrow 1$
while $\|\nabla f_k\|_\infty > \varepsilon$ **do**
    $p_k \leftarrow -H_k^{-1} \nabla f(x_k)$
    $\alpha_k \leftarrow \text{linesearch}(p_k, \alpha_{\text{init}})$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $k \leftarrow k + 1$
end while



$$
\begin{aligned}
f(x_1, x_2) = &\frac{1}{2} k_1 \left(\sqrt{(\ell_1 + x_1)^2 + x_2^2} - \ell_1 \right)^2 \\
&+ \frac{1}{2} k_2 \left(\sqrt{(\ell_2 - x_1)^2 + x_2^2} - \ell_2 \right)^2 - mg x_2
\end{aligned}
$$

Here $\ell_1 = 12, \ell_2 = 8, k_1 = 1, k_2 = 10, mg = 7$

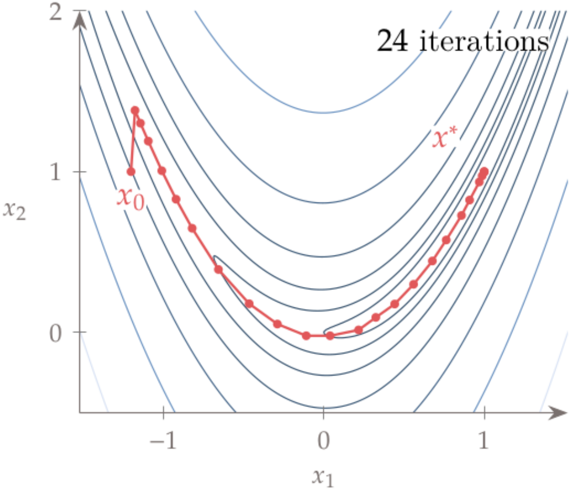
Two Spring Problem - Newton's Method



Gradient descent, line search, stop. cond. $||\nabla f||_\infty \leq 10^{-6}$.
Compare this with 32 iterations of gradient descent.

Rosenbrock Function - Newton's Method

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



Gradient descent, line search, stop. cond. $||\nabla f||_\infty \le 10^{-6}$.
Compare this with 10,662 iterations of gradient descent.

Global Convergence of Line Search

Denote by θ_k the angle between p_k and $-\nabla f_k$, i.e., satisfying

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \, \|p_k\|}$$

Recall that f is L-smooth for some $L > 0$ if

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\|, \qquad \text{for all } x, \tilde{x} \in \mathbb{R}^n$$

Theorem 2 (Zoutendijk)

Consider $x_{k+1} = x_k + \alpha_k p_k$, where p_k is a descent direction *and α_k satisfies the* strong Wolfe conditions*. Suppose that f is bounded below, continuously differentiable, and* L-smooth*. Then*

$$\sum_{k \geq 0} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty.$$

Global Convergence of Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Global Convergence of Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the Hessians H_k are positive definite with a uniformly bounded condition number:

$$||H_k|| \, \left|\left|H_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Global Convergence of Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the Hessians H_k are positive definite with a uniformly bounded condition number:

$$||H_k|| \, \left|\left|H_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Then θ_k between $p_k = -H_k^{-1}\nabla f_k$ and $-\nabla f_k$ satisfies

$$\cos\theta_k \geq 1/M$$

Global Convergence of Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the Hessians H_k are positive definite with a uniformly bounded condition number:

$$||H_k|| \, \left|\left|H_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Then θ_k between $p_k = -H_k^{-1}\nabla f_k$ and $-\nabla f_k$ satisfies

$$\cos\theta_k \geq 1/M$$

Thus, under the assumptions of Zoutendijk's theorem, we obtain

$$\frac{1}{M^2}\sum_{k\geq 0}\|\nabla f_k\|^2 \leq \sum_{k\geq 0}\cos^2\theta_k\,\|\nabla f_k\|^2 < \infty$$

which implies that $\lim_{k\to\infty} ||\nabla f_k|| = 0$.

Global Convergence of Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the Hessians H_k are positive definite with a uniformly bounded condition number:

$$||H_k|| \, \left|\left|H_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Then θ_k between $p_k = -H_k^{-1} \nabla f_k$ and $-\nabla f_k$ satisfies

$$\cos \theta_k \geq 1/M$$

Thus, under the assumptions of Zoutendijk's theorem, we obtain

$$\frac{1}{M^2} \sum_{k \geq 0} \|\nabla f_k\|^2 \leq \sum_{k \geq 0} \cos^2 \theta_k \, \|\nabla f_k\|^2 < \infty$$

which implies that $\lim_{k \to \infty} ||\nabla f_k|| = 0$.

What if H_k is not positive definite or is (nearly) singular?

Eigenvalue Modification

Consider $H_k = \nabla^2 f(x_k)$ and consider its diagonal form:

$$H_k = QDQ^T$$

Where D contains the eigenvalues of H_k on the diagonal, i.e., $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and Q is an orthogonal matrix.

Eigenvalue Modification

Consider $H_k = \nabla^2 f(x_k)$ and consider its diagonal form:

$$H_k = QDQ^T$$

Where D contains the eigenvalues of H_k on the diagonal, i.e., $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and Q is an orthogonal matrix.

Observe that

- H_k is not positive definite iff $\lambda_i \leq 0$ for some i
- $||H_k||$ grows with $\max\{\lambda_1, \ldots, \lambda_n\}$ going to infinity.
- $\left|\left|H_k^{-1}\right|\right|$ grows with $\min\{\lambda_1, \ldots, \lambda_n\}$ going to 0
  (i.e., the matrix becomes close to a singular matrix)

We want to prevent all three cases, i.e., make sure that for some reasonably large $\delta > 0$ we have $\lambda_i \geq \delta$ but not too large.

Eigenvalue Modification

Consider $H_k = \nabla^2 f(x_k)$ and consider its diagonal form:

$$H_k = QDQ^T$$

Where D contains the eigenvalues of H_k on the diagonal, i.e., $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and Q is an orthogonal matrix.

Observe that

- H_k is not positive definite iff $\lambda_i \leq 0$ for some i
- $||H_k||$ grows with $\max\{\lambda_1, \ldots, \lambda_n\}$ going to infinity.
- $\left|\left|H_k^{-1}\right|\right|$ grows with $\min\{\lambda_1, \ldots, \lambda_n\}$ going to 0
  (i.e., the matrix becomes close to a singular matrix)

We want to prevent all three cases, i.e., make sure that for some reasonably large $\delta > 0$ we have $\lambda_i \geq \delta$ but not too large.

Two questions are in order:

- What is a reasonably large δ?
- How to modify H_k so the minimum is large enough?

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Now, the diagonalization is trivial:

$$\nabla^2 f(x_k) = Q\,\mathrm{diag}(10, 3, -1)\; Q^\top \quad Q = I \text{ is the identity matrix}$$

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Now, the diagonalization is trivial:

$$\nabla^2 f(x_k) = Q\,\mathrm{diag}(10, 3, -1)\; Q^\top \quad Q = I \text{ is the identity matrix}$$

What if we consider a minimum modification replacing the negative eigenvalue with a small number, say $\delta = 10^{-8}$?

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Now, the diagonalization is trivial:

$$\nabla^2 f(x_k) = Q\,\mathrm{diag}(10, 3, -1)\;Q^\top \quad Q = I \text{ is the identity matrix}$$

What if we consider a minimum modification replacing the negative eigenvalue with a small number, say $\delta = 10^{-8}$? Obtain

$$B_k = Q\;\mathrm{diag}(10, 3, 10^{-8})\;Q^\top = \mathrm{diag}(10, 3, 10^{-8})$$

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Now, the diagonalization is trivial:

$$\nabla^2 f(x_k) = Q\,\mathrm{diag}(10, 3, -1)\;Q^\top \quad Q = I \text{ is the identity matrix}$$

What if we consider a minimum modification replacing the negative eigenvalue with a small number, say $\delta = 10^{-8}$? Obtain

$$B_k = Q\;\mathrm{diag}(10, 3, 10^{-8})\;Q^\top = \mathrm{diag}(10, 3, 10^{-8})$$

If used in Newton's method, we obtain the following direction:

$$p_k = -B_k^{-1}\nabla f(x_k) = (-1/10, 1, -(2 \cdot 10^8))$$

Thus, a very long vector almost parallel to the third dimension.

Sufficiently Large Eigenvalues

Consider an example:

$$\nabla f(x_k) = (1, -3, 2) \qquad \text{and} \qquad \nabla^2 f(x_k) = \mathrm{diag}(10, 3, -1)$$

Now, the diagonalization is trivial:

$$\nabla^2 f(x_k) = Q\,\mathrm{diag}(10, 3, -1)\; Q^\top \quad Q = I \text{ is the identity matrix}$$

What if we consider a minimum modification replacing the negative eigenvalue with a small number, say $\delta = 10^{-8}$? Obtain

$$B_k = Q\;\mathrm{diag}(10, 3, 10^{-8})\; Q^\top = \mathrm{diag}(10, 3, 10^{-8})$$

If used in Newton's method, we obtain the following direction:

$$p_k = -B_k^{-1}\nabla f(x_k) = (-1/10, 1, -(2 \cdot 10^8))$$

Thus, a very long vector almost parallel to the third dimension.

Even though f decreases along p_k, it is far from the minimum of the quadratic approximation of f.

Note that the original Newton's direction is
$-\mathrm{diag}(1/10, 1/3, -1)(1, -3, 2)^\top = (-1/10, 1, 2)$ which is completely different.

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

The criteria are rather loose. The resulting matrix B_k should be

- positive definite,
- of bounded norm (for all k),
- not too close to being singular.

  (i.e., the eigenvalues should be sufficiently large)

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

The criteria are rather loose. The resulting matrix B_k should be

- positive definite,
- of bounded norm (for all k),
- not too close to being singular.

  (i.e., the eigenvalues should be sufficiently large)

Strategies for eigenvalue modification include flipping negative eigenvalues to positive values, substituting negative eigenvalues with small positive ones, etc.

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

The criteria are rather loose. The resulting matrix B_k should be

- positive definite,
- of bounded norm (for all k),
- not too close to being singular.

  (i.e., the eigenvalues should be sufficiently large)

Strategies for eigenvalue modification include flipping negative eigenvalues to positive values, substituting negative eigenvalues with small positive ones, etc.

There is no consensus on the best method for the modification.

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

The criteria are rather loose. The resulting matrix B_k should be

- positive definite,
- of bounded norm (for all k),
- not too close to being singular.

  (i.e., the eigenvalues should be sufficiently large)

Strategies for eigenvalue modification include flipping negative eigenvalues to positive values, substituting negative eigenvalues with small positive ones, etc.

There is no consensus on the best method for the modification.

The implementation is based on computing $B_k = H_k + \Delta H_k$ for an appropriate modification matrix ΔH_k.

What is ΔH_k in our example?

Modifying the Eigenvalues

Other strategies for eigenvalue modification can be devised.

The criteria are rather loose. The resulting matrix B_k should be

- positive definite,
- of bounded norm (for all k),
- not too close to being singular.

  (i.e., the eigenvalues should be sufficiently large)

Strategies for eigenvalue modification include flipping negative eigenvalues to positive values, substituting negative eigenvalues with small positive ones, etc.

There is no consensus on the best method for the modification.

The implementation is based on computing $B_k = H_k + \Delta H_k$ for an appropriate modification matrix ΔH_k.

What is ΔH_k in our example?

Various methods for computing ΔH_k have been devised in literature. Typically, it is based on some computationally cheaper decomposition than spectral decomposition (e.g., Cholesky).

Modified Newton's Method

Algorithm 3 Newton's Method with Line Search

Input: x_0 starting point, $\varepsilon > 0$

Output: x^* approximation to a stationary point

$k \leftarrow 0$
while $\|\nabla f_k\|_\infty > \varepsilon$ **do**
  $H_k \leftarrow \nabla^2 f(x_k)$
  if H_k is **not** sufficiently positive definite **then**
    $H_k \leftarrow H_k + \Delta H_k$ so that H_k is sufficiently pos. definite
  end if
  Solve $H_k p_k = -\nabla f(x_k)$ for p_k
  Set $x_{k+1} = x_k + \alpha_k p_k$, here α_k sat. the Wolfe cond.
  $k \leftarrow k + 1$
end while

Convergence of Modified Newton's Method

Comments on Newton's Method

- Newton's method is scale invariant.

Comments on Newton's Method

- Newton's method is scale invariant.
- Quadratic convergence in a close vicinity of a strict minimizer.

Comments on Newton's Method

- Newton's method is scale invariant.
- Quadratic convergence in a close vicinity of a strict minimizer.
- Without modification, it may converge to an arbitrary stationary point (maximum, saddle point).

Comments on Newton's Method

- Newton's method is scale invariant.
- Quadratic convergence in a close vicinity of a strict minimizer.
- Without modification, it may converge to an arbitrary stationary point (maximum, saddle point).
- Computationally expensive:
  - $\mathcal{O}(n^2)$ second derivatives in the Hessian, each may be hard to compute.
    Automated derivation methods help but still need store $\mathcal{O}(n^2)$ results.

Comments on Newton's Method

- Newton's method is scale invariant.
- Quadratic convergence in a close vicinity of a strict minimizer.
- Without modification, it may converge to an arbitrary stationary point (maximum, saddle point).
- Computationally expensive:
  - $\mathcal{O}(n^2)$ second derivatives in the Hessian, each may be hard to compute.
    Automated derivation methods help but still need store $\mathcal{O}(n^2)$ results.
  - $\mathcal{O}(n^3)$ arithmetic operations to solve the linear system for the direction p_k.
    May be mitigated by more efficient methods in case of sparse Hessians.

Comments on Newton's Method

- Newton's method is scale invariant.
- Quadratic convergence in a close vicinity of a strict minimizer.
- Without modification, it may converge to an arbitrary stationary point (maximum, saddle point).

- Computationally expensive:
  - $\mathcal{O}(n^2)$ second derivatives in the Hessian, each may be hard to compute.
    Automated derivation methods help but still need store $\mathcal{O}(n^2)$ results.
  - $\mathcal{O}(n^3)$ arithmetic operations to solve the linear system for the direction p_k.
    May be mitigated by more efficient methods in case of sparse Hessians.

In a sense, Newton's method is an impractical "ideal" with which other methods are compared.

The efficiency issues (and the necessity of second-order derivatives) will be mitigated by using quasi-Newton methods.

Quasi-Newton Methods

Quasi-Newton Methods

Recall that Newton's method step p_k in $x_{k+1} = x_k + p_k$ comes from minimization of

$$q(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top H_k p$$

w.r.t. p by setting $\nabla q(p) = 0$ and solving

$$H_k p = -\nabla f_k$$

So Newton's method needs the second derivative (Hessian), which is computationally hard to obtain.

Quasi-Newton Methods

Recall that Newton's method step p_k in $x_{k+1} = x_k + p_k$ comes from minimization of

$$q(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top H_k p$$

w.r.t. p by setting $\nabla q(p) = 0$ and solving

$$H_k p = -\nabla f_k$$

So Newton's method needs the second derivative (Hessian), which is computationally hard to obtain.

Gradient descent needs only the first derivatives but converges slowly.

Quasi-Newton Methods

Recall that Newton's method step p_k in $x_{k+1} = x_k + p_k$ comes from minimization of

$$q(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top H_k p$$

w.r.t. p by setting $\nabla q(p) = 0$ and solving

$$H_k p = -\nabla f_k$$

So Newton's method needs the second derivative (Hessian), which is computationally hard to obtain.

Gradient descent needs only the first derivatives but converges slowly.

Can we find a compromise?

Quasi-Newton Methods

Recall that Newton's method step p_k in $x_{k+1} = x_k + p_k$ comes from minimization of

$$q(p) = f_k + \nabla f_k^\top p + \frac{1}{2} p^\top H_k p$$

w.r.t. p by setting $\nabla q(p) = 0$ and solving

$$H_k p = -\nabla f_k$$

So Newton's method needs the second derivative (Hessian), which is computationally hard to obtain.

Gradient descent needs only the first derivatives but converges slowly.

Can we find a compromise?

Quasi-Newton methods use first derivatives to approximate the Hessian H_k in Newton's method with a matrix $\tilde{H}_k$.

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Consider the Hessian $H_{k+1} = \nabla^2 f(x_{k+1})$ and its approximation denoted by $\tilde{H}_{k+1}$.

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Consider the Hessian $H_{k+1} = \nabla^2 f(x_{k+1})$ and its approximation denoted by $\tilde{H}_{k+1}$.

We aim to use $\tilde{H}_{k+1}$ in the next step, that is, in the equation $\tilde{H}_{k+1} p = -\nabla f_{k+1}$ yielding p_{k+1}.

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Consider the Hessian $H_{k+1} = \nabla^2 f(x_{k+1})$ and its approximation denoted by $\tilde{H}_{k+1}$.

We aim to use $\tilde{H}_{k+1}$ in the next step, that is, in the equation $\tilde{H}_{k+1} p = -\nabla f_{k+1}$ yielding p_{k+1}.

What conditions should $\tilde{H}_{k+1}$ satisfy so that it functions as the "true" Hessian H_{k+1}?

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Consider the Hessian $H_{k+1} = \nabla^2 f(x_{k+1})$ and its approximation denoted by $\tilde{H}_{k+1}$.

We aim to use $\tilde{H}_{k+1}$ in the next step, that is, in the equation $\tilde{H}_{k+1} p = -\nabla f_{k+1}$ yielding p_{k+1}.

What conditions should $\tilde{H}_{k+1}$ satisfy so that it functions as the "true" Hessian H_{k+1}?

First, it should be *symmetric positive definite*.

To always yield decrease direction.

Quasi-Newton Methods

Suppose we have just obtained the new point x_{k+1} after a line search starting from x_k in the direction p_k.

Consider the Hessian $H_{k+1} = \nabla^2 f(x_{k+1})$ and its approximation denoted by $\tilde{H}_{k+1}$.

We aim to use $\tilde{H}_{k+1}$ in the next step, that is, in the equation $\tilde{H}_{k+1} p = -\nabla f_{k+1}$ yielding p_{k+1}.

What conditions should $\tilde{H}_{k+1}$ satisfy so that it functions as the "true" Hessian H_{k+1}?

First, it should be *symmetric positive definite*.

To always yield decrease direction.

Second, extrapolating from the single variable secant method, we demand

$$\tilde{H}_{k+1}(x_{k+1} - x_k) = \nabla f_{k+1} - \nabla f_k$$

This is the *secant condition*.

Secant Condition

Consider the secant condition:

$$\tilde{H}_{k+1}(x_{k+1} - x_k) = \nabla f_{k+1} - \nabla f_k$$

The notation is usually simplified by

$$s_k = x_{k+1} - x_k \qquad y_k = \nabla f_{k+1} - \nabla f_k$$

So that the secant condition becomes

$$\tilde{H}_{k+1} s_k = y_k$$

Secant Condition

Consider the secant condition:

$$\tilde{H}_{k+1}(x_{k+1} - x_k) = \nabla f_{k+1} - \nabla f_k$$

The notation is usually simplified by

$$s_k = x_{k+1} - x_k \qquad y_k = \nabla f_{k+1} - \nabla f_k$$

So that the secant condition becomes

$$\tilde{H}_{k+1} s_k = y_k$$

Does it have a symmetric positive definite solution?

Curvature Condition

Consider the secant condition:

$$\tilde{H}_{k+1} s_k = y_k$$

Curvature Condition

Consider the secant condition:

$$\tilde{H}_{k+1} s_k = y_k$$

The following is true:

- The secant condition has a symmetric positive definite solution iff the following condition is satisfied:

$$s_k^\top y_k > 0$$

Curvature Condition

Consider the secant condition:

$$\tilde{H}_{k+1} s_k = y_k$$

The following is true:

- The secant condition has a symmetric positive definite solution iff the following condition is satisfied:

  $$s_k^\top y_k > 0$$

- The condition $s_k^\top y_k > 0$ is satisfied if the line search satisfies the strong Wolfe conditions.

Curvature Condition

Consider the secant condition:

$$\tilde{H}_{k+1} s_k = y_k$$

The following is true:

- The secant condition has a symmetric positive definite solution iff the following condition is satisfied:

  $$s_k^\top y_k > 0$$

- The condition $s_k^\top y_k > 0$ is satisfied if the line search satisfies the strong Wolfe conditions.

As a corollary, we obtain the following:

Theorem 3

Assume that we use line search satisfying strong Wolfe conditions. Then in every step, the secant condition

$$\tilde{H}_{k+1} s_k = y_k$$

has a symmetric positive definite solution $\tilde{H}_{k+1}$.

Now, we can obtain an approximate Hessian $\tilde{H}_{k+1}$ by solving the secant condition $\tilde{H}_{k+1} s_k = y_k$.

Now, we can obtain an approximate Hessian $\tilde{H}_{k+1}$ by solving the secant condition $\tilde{H}_{k+1}s_k = y_k$.

Note that even if we demand symmetric positive definite solutions to the secant condition, there are infinitely many.

Indeed, there are $n(n+1)/2$ degrees of freedom in a symmetric matrix, and the secant conditions represent only n conditions.

Now, we can obtain an approximate Hessian $\tilde{H}_{k+1}$ by solving the secant condition $\tilde{H}_{k+1} s_k = y_k$.

Note that even if we demand symmetric positive definite solutions to the secant condition, there are infinitely many.

Indeed, there are $n(n+1)/2$ degrees of freedom in a symmetric matrix, and the secant conditions represent only n conditions.

Moreover, we want to obtain $\tilde{H}_{k+1}$ from $\tilde{H}_k$ by

$$\tilde{H}_{k+1} = \tilde{H}_k + \text{something}$$

To have a nice iterative algorithm.

Now, we can obtain an approximate Hessian $\tilde{H}_{k+1}$ by solving the secant condition $\tilde{H}_{k+1}s_k = y_k$.

Note that even if we demand symmetric positive definite solutions to the secant condition, there are infinitely many.

Indeed, there are $n(n+1)/2$ degrees of freedom in a symmetric matrix, and the secant conditions represent only n conditions.

Moreover, we want to obtain $\tilde{H}_{k+1}$ from $\tilde{H}_k$ by

$$\tilde{H}_{k+1} = \tilde{H}_k + \text{something}$$

To have a nice iterative algorithm.

We also want $\tilde{H}_{k+1}$ to be symmetric positive definite.

Now, we can obtain an approximate Hessian $\tilde{H}_{k+1}$ by solving the secant condition $\tilde{H}_{k+1} s_k = y_k$.

Note that even if we demand symmetric positive definite solutions to the secant condition, there are infinitely many.

Indeed, there are $n(n+1)/2$ degrees of freedom in a symmetric matrix, and the secant conditions represent only n conditions.

Moreover, we want to obtain $\tilde{H}_{k+1}$ from $\tilde{H}_k$ by

$$\tilde{H}_{k+1} = \tilde{H}_k + \text{something}$$

To have a nice iterative algorithm.

We also want $\tilde{H}_{k+1}$ to be symmetric positive definite.

We strive to choose $\tilde{H}_{k+1}$ "close" to $\tilde{H}_k$.

Symmetric Rank One Update (SR1)

Note that the information about the solution is present in s_k and y_k, so it is natural to compose the solution using these vectors.

Symmetric Rank One Update (SR1)

Note that the information about the solution is present in s_k and y_k, so it is natural to compose the solution using these vectors.

Consider $u = \left(y_k - \tilde{H}_k s_k\right)$

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{uu^\top}{u^\top s_k}$$

Symmetric Rank One Update (SR1)

Note that the information about the solution is present in s_k and y_k, so it is natural to compose the solution using these vectors.

Consider $u = \left(y_k - \tilde{H}_k s_k \right)$

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{uu^\top}{u^\top s_k}$$

Now, the secant condition is satisfied:

$$\tilde{H}_{k+1} s_k = \tilde{H}_k s_k + \frac{uu^\top s_k}{u^\top s_k} = \tilde{H}_k s_k + u = \tilde{H}_k s_k + \left(y_k - \tilde{H}_k s_k \right) = y_k$$

By the way, the matrix $\frac{uu^\top}{u^\top s_k}$ is of rank one and is a unique symmetric rank one matrix which makes $\tilde{H}_{k+1}$ satisfy the secant condition.

Symmetric Rank One Update (SR1)

Note that the information about the solution is present in s_k and y_k, so it is natural to compose the solution using these vectors.

Consider $u = \left(y_k - \tilde{H}_k s_k\right)$

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{uu^\top}{u^\top s_k}$$

Now, the secant condition is satisfied:

$$\tilde{H}_{k+1} s_k = \tilde{H}_k s_k + \frac{uu^\top s_k}{u^\top s_k} = \tilde{H}_k s_k + u = \tilde{H}_k s_k + \left(y_k - \tilde{H}_k s_k\right) = y_k$$

By the way, the matrix $\frac{uu^\top}{u^\top s_k}$ is of rank one and is a unique symmetric rank one matrix which makes $\tilde{H}_{k+1}$ satisfy the secant condition.

To obtain a quasi-Newton method, it suffices to initialize $\tilde{H}_0$, typically to the identity I, and use $\tilde{H}_k$ instead of the Hessian $H_k = \nabla^2 f_k$ in Newton's method.

Symmetric Rank One Update

Algorithm 4 SR1

$k \leftarrow 0$
$\alpha_{\text{init}} \leftarrow 1$
$\tilde{H}_0 \leftarrow I$
while $\|\nabla f_k\|_\infty > \tau$ **do**
  Solve for p_k in $\tilde{H}_k p_k = -\nabla f_k$
  $\alpha \leftarrow \text{linesearch}(p_k, \alpha_{\text{init}})$
  $x_{k+1} \leftarrow x_k + \alpha p_k$
  $s \leftarrow x_{k+1} - x_k$
  $y \leftarrow \nabla f_{k+1} - \nabla f_k$
  $u \leftarrow y - \tilde{H}_k s$
  $\tilde{H}_{k+1} \leftarrow \tilde{H}_k + \frac{uu^\top}{u^\top s}$
  $k \leftarrow k + 1$
end while

Note that the denominator $u^\top s_k$ can be 0, in which case the update is impossible. The usual strategy is to skip the update and set $\tilde{H}_{k+1} = \tilde{H}_k$.

Example

We will look at a three-dimensional quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

The initial guesses are $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^\top$.

Example

We will look at a three-dimensional quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

The initial guesses are $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^\top$.

At the initial point, $\|\nabla f(x_0)\|_\infty = \|-c\|_\infty = 9$, so this point is not optimal.

Example

We will look at a three-dimensional quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

The initial guesses are $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^\top$.

At the initial point, $\|\nabla f(x_0)\|_\infty = \|-c\|_\infty = 9$, so this point is not optimal.The first search direction is

$$p_0 = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix}.$$

The exact line search gives $\alpha_0 = 0.3333$.

Example

The new estimate of the solution, the update vectors, and the new Hessian approximation are:

$$x_1 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, \nabla f_1 = \begin{pmatrix} 2.66 \\ 0 \\ -2.66 \end{pmatrix}, s_0 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, y_0 = \begin{pmatrix} -5.33 \\ -9.00 \\ -10.66 \end{pmatrix},$$

Example

The new estimate of the solution, the update vectors, and the new Hessian approximation are:

$$x_1 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, \nabla f_1 = \begin{pmatrix} 2.66 \\ 0 \\ -2.66 \end{pmatrix}, s_0 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, y_0 = \begin{pmatrix} -5.33 \\ -9.00 \\ -10.66 \end{pmatrix},$$

and

$$\tilde{H}_1 = I + \frac{(y_0 - Is_0)(y_0 - Is_0)^\top}{(y_0 - Is_0)^\top s_0} = \begin{pmatrix} 1.1531 & 0.3445 & 0.4593 \\ 0.3445 & 1.7751 & 1.0335 \\ 0.4593 & 1.0335 & 2.3780 \end{pmatrix}.$$

Example

The new estimate of the solution, the update vectors, and the new Hessian approximation are:

$$x_1 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, \nabla f_1 = \begin{pmatrix} 2.66 \\ 0 \\ -2.66 \end{pmatrix}, s_0 = \begin{pmatrix} -2.66 \\ -3.00 \\ -2.66 \end{pmatrix}, y_0 = \begin{pmatrix} -5.33 \\ -9.00 \\ -10.66 \end{pmatrix},$$

and

$$\tilde{H}_1 = I + \frac{(y_0 - Is_0)(y_0 - Is_0)^\top}{(y_0 - Is_0)^\top s_0} = \begin{pmatrix} 1.1531 & 0.3445 & 0.4593 \\ 0.3445 & 1.7751 & 1.0335 \\ 0.4593 & 1.0335 & 2.3780 \end{pmatrix}.$$

At this new point $\|\nabla f(x_1)\|_\infty = 2.66$ so we keep going, obtaining the search direction

$$p_1 = \begin{pmatrix} -2.9137 \\ -0.5557 \\ 1.9257 \end{pmatrix},$$

and the step length $\alpha_1 = 0.3942$.

Example

This gives the new estimates:

$$x_2 = \begin{pmatrix} -3.81 \\ -3.21 \\ -1.90 \end{pmatrix}, \quad \nabla f_2 = \begin{pmatrix} 0.36 \\ -0.65 \\ 0.36 \end{pmatrix}, \quad s_1 = \begin{pmatrix} -1.14 \\ -0.21 \\ 0.75 \end{pmatrix}, \quad y_1 = \begin{pmatrix} -2.29 \\ -0.65 \\ 3.03 \end{pmatrix}$$

and

$$\tilde{H}_2 = \begin{pmatrix} 1.6568 & 0.6102 & -0.3432 \\ 0.6102 & 1.9153 & 0.6102 \\ -0.3432 & 0.6102 & 3.6568 \end{pmatrix}.$$

At the point x_2, $\|\nabla f(x_2)\|_\infty = 0.65$ so we keep going, with

$$p_2 = \begin{pmatrix} -0.4851 \\ 0.5749 \\ -0.2426 \end{pmatrix},$$

and $\alpha = 0.3810$.

Example

This gives

$$x_3 = \begin{pmatrix} -4 \\ -3 \\ -2 \end{pmatrix}, \quad \nabla f_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad s_2 = \begin{pmatrix} -0.18 \\ 0.21 \\ -0.09 \end{pmatrix}, \quad y_2 = \begin{pmatrix} -0.36 \\ 0.65 \\ -0.36 \end{pmatrix},$$

and $\tilde{H}_3 = Q$. Now $\|\nabla f(x_3)\|_\infty = 0$, so we stop.

Properties of SR1

Does symmetric rank one update satisfy our demands?

We want every $\tilde{H}_k$ to be a symmetric positive definite solution to the secant condition.

Properties of SR1

Does symmetric rank one update satisfy our demands?

We want every $\tilde{H}_k$ to be a symmetric positive definite solution to the secant condition.

Unfortunately, though $\tilde{H}_k$ is a symmetric positive definite, the updated matrix $\tilde{H}_{k+1}$ does not have to be a positive definite.

Properties of SR1

Does symmetric rank one update satisfy our demands?

We want every $\tilde{H}_k$ to be a symmetric positive definite solution to the secant condition.

Unfortunately, though $\tilde{H}_k$ is a symmetric positive definite, the updated matrix $\tilde{H}_{k+1}$ does not have to be a positive definite.

Still, the symmetric rank one approximation is used in practice, especially in trust region methods.

Properties of SR1

Does symmetric rank one update satisfy our demands?

We want every $\tilde{H}_k$ to be a symmetric positive definite solution to the secant condition.

Unfortunately, though $\tilde{H}_k$ is a symmetric positive definite, the updated matrix $\tilde{H}_{k+1}$ does not have to be a positive definite.

Still, the symmetric rank one approximation is used in practice, especially in trust region methods.

However, for line search, let us try a bit “richer” solution to the secant condition.

Symmetric Rank Two Update

Consider

$$\tilde{H}_{k+1} = \tilde{H}_k - \frac{\left(\tilde{H}_k s_k\right)\left(\tilde{H}_k s_k\right)^\top}{s_k^\top \tilde{H}_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}$$

Once again, verifying $\tilde{H}_{k+1} s_k = y_k$ is not difficult.

Lemma 1

Assume that $\tilde{H}_k$ is symmetric positive definite.
Then $\tilde{H}_{k+1}$ is symmetric positive definite iff $y_k^\top s_k > 0$.

We know that line search satisfying the strong Wolfe conditions preserves $y_k^\top s_k > 0$.

Thus, starting with a symmetric positive definite $\tilde{H}_0$ (e.g., a scalar multiple of I), every $\tilde{H}_k$ is symmetric positive definite and satisfies the secant condition.

BFGS

Algorithm 5 BFGS v1

$k \leftarrow 0$
$\alpha_{\text{init}} \leftarrow 1$
$\tilde{H}_0 \leftarrow I$
while $\|\nabla f_k\|_\infty > \tau$ **do**
  Solve for p_k in $\tilde{H}_k p_k = -\nabla f_k$
  $\alpha \leftarrow \text{linesearch}(p_k, \alpha_{\text{init}})$
  $x_{k+1} \leftarrow x_k + \alpha p_k$
  $s \leftarrow x_{k+1} - x_k$
  $y \leftarrow \nabla f_{k+1} - \nabla f_k$
  $\tilde{H}_{k+1} \leftarrow \tilde{H}_k - \frac{(\tilde{H}_k s)(\tilde{H}_k s)^\top}{s^\top \tilde{H}_k s} + \frac{yy^\top}{y^\top s}$
  $k \leftarrow k + 1$
end while

Note that we still have to solve a linear system for p_k.

Example

Consider the quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{bmatrix} -8 \\ -9 \\ -8 \end{bmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

Example

Consider the quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{bmatrix} -8 \\ -9 \\ -8 \end{bmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

Choose $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^T$.

Example

Consider the quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{bmatrix} -8 \\ -9 \\ -8 \end{bmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

Choose $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^T$.

At iteration 0, $\|\nabla f(x_0)\|_\infty = 9$, so this point is not optimal.

Example

Consider the quadratic problem $f(x) = \frac{1}{2}x^\top Qx - c^\top x$ with

$$Q = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad c = \begin{bmatrix} -8 \\ -9 \\ -8 \end{bmatrix},$$

whose solution is $x_* = (-4, -3, -2)^\top$. Use the exact line search.

Choose $\tilde{H}_0 = I$ and $x_0 = (0, 0, 0)^T$.

At iteration 0, $\|\nabla f(x_0)\|_\infty = 9$, so this point is not optimal.

The search direction is

$$p_0 = \begin{pmatrix} -8 \\ -9 \\ -8 \end{pmatrix}$$

and $\alpha_0 = 0.3333$.

Example

The new estimate of the solution and the new Hessian approximation are

$$x_1 = \begin{pmatrix} -2.6667 \\ -3.0000 \\ -2.6667 \end{pmatrix} \quad \text{and} \quad \tilde{H}_1 = \begin{pmatrix} 1.1021 & 0.3445 & 0.5104 \\ 0.3445 & 1.7751 & 1.0335 \\ 0.5104 & 1.0335 & 2.3270 \end{pmatrix}.$$

Example

The new estimate of the solution and the new Hessian approximation are

$$x_1 = \begin{pmatrix} -2.6667 \\ -3.0000 \\ -2.6667 \end{pmatrix} \quad \text{and} \quad \tilde{H}_1 = \begin{pmatrix} 1.1021 & 0.3445 & 0.5104 \\ 0.3445 & 1.7751 & 1.0335 \\ 0.5104 & 1.0335 & 2.3270 \end{pmatrix}.$$

At iteration 1, $\|\nabla f(x_1)\|_\infty = 2.6667$, so we continue. The next search direction is

$$p_1 = \begin{pmatrix} -3.2111 \\ -0.6124 \\ 2.1223 \end{pmatrix}$$

and $\alpha_1 = 0.3577$.

Example

This gives the estimates.

$$x_2 = \begin{pmatrix} -3.8152 \\ -3.2191 \\ -1.9076 \end{pmatrix} \quad \text{and} \quad \tilde{H}_2 = \begin{pmatrix} 1.6393 & 0.6412 & -0.3607 \\ 0.6412 & 1.8600 & 0.6412 \\ -0.3607 & 0.6412 & 3.6393 \end{pmatrix}.$$

At iteration 2, $\|\nabla f(x_2)\|_\infty = 0.6572$, so we continue, computing

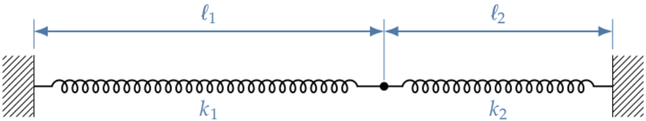
$$p_2 = \begin{pmatrix} -0.5289 \\ 0.6268 \\ -0.2644 \end{pmatrix}$$

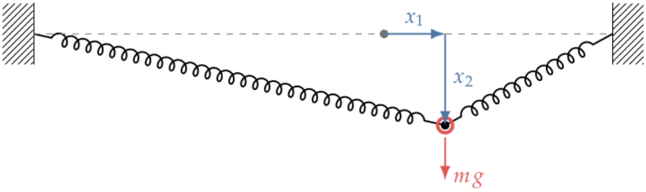
and $\alpha_2 = 0.3495$. This gives

$$x_3 = \begin{pmatrix} -4 \\ -3 \\ -2 \end{pmatrix} \quad \text{and} \quad \tilde{H}_3 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

Now $\|\nabla f(x_3)\|_\infty = 0$, so we stop.

Notice that we got the same x_1, x_2, x_3 as for SR1. This follows from using the exact line search and the quadratic problem. It does not hold in general.

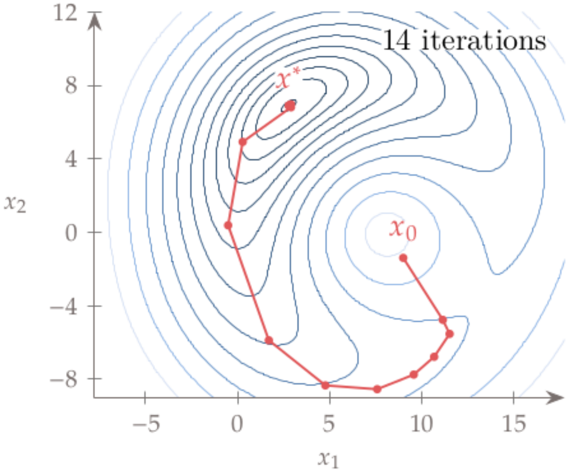




$$f(x_1, x_2) = \frac{1}{2}k_1\left(\sqrt{(\ell_1 + x_1)^2 + x_2^2} - \ell_1\right)^2 + \frac{1}{2}k_2\left(\sqrt{(\ell_2 - x_1)^2 + x_2^2} - \ell_2\right)^2 - mgx_2$$

Here $\ell_1 = 12, \ell_2 = 8, k_1 = 1, k_2 = 10, mg = 7$

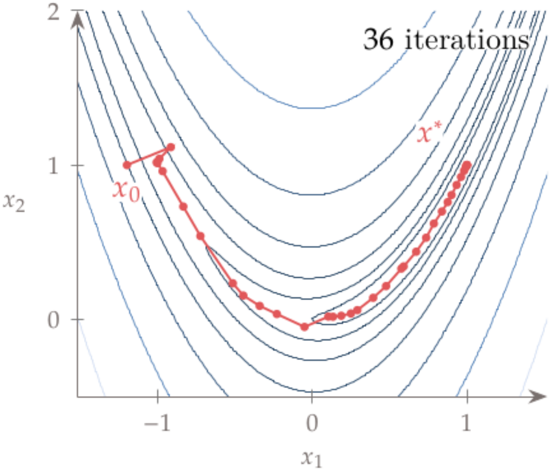
Two Spring Problem - BFGS



Gradient descent, line search, stop. cond. $||\nabla f||_\infty \leq 10^{-6}$.
Compare this with 32 iterations of gradient descent and 12 iterations of Newton's method.

Rosenbrock Function - BFGS

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



Gradient descent, line search, stop. cond. $||\nabla f||_\infty \leq 10^{-6}$.
Compare with 10,662 iterations of gradient descent and 24 iterations of Newton's method.

Sherman–Morrison–Woodbury Formula

Problem: SR1 and BFGS solve $\tilde{H}_k p = -\nabla f_k$ repeatedly. What if we could iteratively update H_k^{-1}?

Sherman–Morrison–Woodbury Formula

Problem: SR1 and BFGS solve $\tilde{H}_k p = -\nabla f_k$ repeatedly. What if we could iteratively update H_k^{-1}?

The equation would be solved by $p_k = -H_k^{-1} \nabla f_k$.

Sherman–Morrison–Woodbury Formula

Problem: SR1 and BFGS solve $\tilde{H}_k p = -\nabla f_k$ repeatedly. What if we could iteratively update H_k^{-1}?
The equation would be solved by $p_k = -H_k^{-1}\nabla f_k$.

Ideally, we would like to compute $\tilde{H}_k^{-1}$ iteratively along the optimization, i.e.,

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \text{something}$$

Sherman–Morrison–Woodbury Formula

Problem: SR1 and BFGS solve $\tilde{H}_k p = -\nabla f_k$ repeatedly. What if we could iteratively update H_k^{-1}?
The equation would be solved by $p_k = -H_k^{-1} \nabla f_k$.

Ideally, we would like to compute $\tilde{H}_k^{-1}$ iteratively along the optimization, i.e.,

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \text{something}$$

To get such a "something" we use the following Sherman–Morrison–Woodbury (SMW) formula:

$$\left(A + UV^T\right)^{-1} = A^{-1} - A^{-1} U \left(I + V^T A^{-1} U\right)^{-1} V^T A^{-1}$$

Here A is a $(n \times n)$-matrix, U, V are $(n \times m)$-matrices with $m \leq n$.

Rank 1 – Iterative Inverse Hessian Approximation

Applying SMW to the rank one update

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{\left(y_k - \tilde{H}_k s_k\right)\left(y_k - \tilde{H}_k s_k\right)^\top}{\left(y_k - \tilde{H}_k s_k\right)^\top s_k}$$

Rank 1 – Iterative Inverse Hessian Approximation

Applying SMW to the rank one update

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{\left(y_k - \tilde{H}_k s_k\right)\left(y_k - \tilde{H}_k s_k\right)^\top}{\left(y_k - \tilde{H}_k s_k\right)^\top s_k}$$

yields

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{\left(s_k - \tilde{H}_k^{-1} y_k\right)\left(s_k - \tilde{H}_k^{-1} y_k\right)^\top}{\left(s_k - \tilde{H}_k^{-1} y_k\right)^\top y_k}$$

Yes, only y and s swapped places.

Rank 1 – Iterative Inverse Hessian Approximation

Applying SMW to the rank one update

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{\left(y_k - \tilde{H}_k s_k\right)\left(y_k - \tilde{H}_k s_k\right)^\top}{\left(y_k - \tilde{H}_k s_k\right)^\top s_k}$$

yields

$$\tilde{H}_{k+1}^{-1} = \tilde{H}_k^{-1} + \frac{\left(s_k - \tilde{H}_k^{-1} y_k\right)\left(s_k - \tilde{H}_k^{-1} y_k\right)^\top}{\left(s_k - \tilde{H}_k^{-1} y_k\right)^\top y_k}$$

Yes, only y and s swapped places.

This allows us to avoid solving $\tilde{H}_k p_k = -\nabla f_k$ for p_k in every iteration.

Rank One Update V2

Algorithm 6 Rank 1 update v1

$k \leftarrow 0$

$\alpha_{\text{init}} \leftarrow 1$

$\tilde{H}_0 \leftarrow I$

while $\|\nabla f_k\|_\infty > \tau$ **do**

    $p_k \leftarrow -\tilde{H}_k^{-1} \nabla f_k$

    $\alpha \leftarrow \text{linesearch}(p_k, \alpha_{\text{init}})$

    $x_{k+1} \leftarrow x_k + \alpha p_k$

    $s \leftarrow x_k - x_{k-1}$

    $y \leftarrow \nabla f_k - \nabla f_{k-1}$

    $\tilde{H}_{k+1}^{-1} \leftarrow \tilde{H}_k^{-1} + \frac{\left(s - \tilde{H}_k^{-1} y\right)\left(s - \tilde{H}_k^{-1} y\right)^\top}{\left(s - \tilde{H}_k^{-1} y\right)^\top y}$

    $k \leftarrow k + 1$

end while

BFGS

Applying SMW to the BFGS Hessian update

$$\tilde{H}_{k+1} = \tilde{H}_k - \frac{\left(\tilde{H}_k s_k\right)\left(\tilde{H}_k s_k\right)^\top}{s_k^\top \tilde{H}_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}$$

BFGS

Applying SMW to the BFGS Hessian update

$$\tilde{H}_{k+1} = \tilde{H}_k - \frac{\left(\tilde{H}_k s_k\right)\left(\tilde{H}_k s_k\right)^\top}{s_k^\top \tilde{H}_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}$$

yields

$$\tilde{H}_{k+1}^{-1} = \left(I - \frac{s_k y_k^\top}{s_k^\top y_k}\right) \tilde{H}_k^{-1} \left(I - \frac{y_k s_k^\top}{s_k^\top y_k}\right) + \frac{s_k s_k^\top}{s_k^\top y_k}$$

We avoid solving the linear system for p_k.

BFGS V2

Algorithm 7 BFGS v2

$k \leftarrow 0$
$\alpha_{\text{init}} \leftarrow 1$
$\tilde{H}_0 \leftarrow I$
while $\|\nabla f_k\|_\infty > \tau$ **do**
    $p_k \leftarrow -\tilde{H}_k^{-1} \nabla f_k$
    $\alpha \leftarrow \text{linesearch}(p_k, \alpha_{\text{init}})$
    $x_{k+1} \leftarrow x_k + \alpha p_k$
    $k \leftarrow k + 1$
    $s \leftarrow x_k - x_{k-1}$
    $y \leftarrow \nabla f_k - \nabla f_{k-1}$
    $\tilde{H}_{k+1}^{-1} \leftarrow \left(I - \frac{sy^\top}{s^\top y}\right) \tilde{H}_k^{-1} \left(I - \frac{ys^\top}{s^\top y}\right) + \frac{ss^\top}{s^\top y}$
end while

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Observe that $\tilde{H}_k$ is determined completely by H_0 and the two sequences $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Observe that $\tilde{H}_k$ is determined completely by H_0 and the two sequences $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

So, the matrix $\tilde{H}_k$ does not have to be stored if the algorithm remembers the values $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

Note that this would be more space efficient for $k < n$.

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Observe that $\tilde{H}_k$ is determined completely by H_0 and the two sequences $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

So, the matrix $\tilde{H}_k$ does not have to be stored if the algorithm remembers the values $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

Note that this would be more space efficient for $k < n$.

However, we may go further and observe that typically only a few, say m, past values of s and y are sufficient for a good approximation of $\tilde{H}_k$ when we set $\tilde{H}_{k-m-1} = I$.

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Observe that $\tilde{H}_k$ is determined completely by H_0 and the two sequences $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

So, the matrix $\tilde{H}_k$ does not have to be stored if the algorithm remembers the values $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

Note that this would be more space efficient for $k < n$.

However, we may go further and observe that typically only a few, say m, past values of s and y are sufficient for a good approximation of $\tilde{H}_k$ when we set $\tilde{H}_{k-m-1} = I$.

This is the basic idea behind limited-memory BFGS which stores only the running window $s_{k-m}, \ldots, s_k$ and $y_{k-m}, \ldots, y_k$ and computes $\tilde{H}_k$ using these values as if initialized by $\tilde{H}_{k-m-1} = I$.

Limited Memory BFGS Idea

Let us denote by $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$ the values of the variables s and y, resp., during the iterations $1, \ldots, k$ of BFGS.

Observe that $\tilde{H}_k$ is determined completely by H_0 and the two sequences $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

So, the matrix $\tilde{H}_k$ does not have to be stored if the algorithm remembers the values $s_0, \ldots, s_k$ and $y_0, \ldots, y_k$.

Note that this would be more space efficient for $k < n$.

However, we may go further and observe that typically only a few, say m, past values of s and y are sufficient for a good approximation of $\tilde{H}_k$ when we set $\tilde{H}_{k-m-1} = I$.

This is the basic idea behind limited-memory BFGS which stores only the running window $s_{k-m}, \ldots, s_k$ and $y_{k-m}, \ldots, y_k$ and computes $\tilde{H}_k$ using these values as if initialized by $\tilde{H}_{k-m-1} = I$.

The space complexity becomes nm, which is beneficial when n is large.

Another View on BFGS (Optional)

We search for $\tilde{H}_{k+1}^{-1}$ where $\tilde{H}_{k+1}$ satisfies $\tilde{H}_{k+1} s_k = y_k$. Search for a solution $\tilde{V}$ for $\tilde{V} y_k = s_k$.

The idea is to use $\tilde{V}$ close to $\tilde{H}_k^{-1}$ (in some sense):

$$\min_{\tilde{H}} \left\| \tilde{V} - \tilde{H}_k^{-1} \right\|$$

$$\text{subject to} \quad \tilde{V} = \tilde{V}^\top, \quad \tilde{V} y_k = s_k$$

Here the norm is *weighted Frobenius norm*:

$$\|A\| \equiv \left\| W^{1/2} A W^{1/2} \right\|_F,$$

where $\|\cdot\|_F$ is defined by $\|C\|_F^2 = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^2$. The weight W can be chosen as any matrix satisfying the relation $W y_k = s_k$.

BFGS is obtained with $W = \bar{G}_k^{-1}$ where $\bar{G}_k$ is the average Hessian defined by $\bar{G}_k = \left[\int_0^1 \nabla^2 f\left(x_k + \tau \alpha_k p_k\right) d\tau \right]$

Solving this gives precisely the BFGS formula for $\tilde{H}_{k+1}^{-1}$.

Global Convergence of Line Search

Denote by θ_k the angle between p_k and $-\nabla f_k$, i.e., satisfying

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \, \|p_k\|}$$

Recall that f is L-smooth for some $L > 0$ if

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\|, \qquad \text{for all } x, \tilde{x} \in \mathbb{R}^n$$

Theorem 4 (Zoutendijk)

Consider $x_{k+1} = x_k + \alpha_k p_k$, *where* p_k *is a* descent direction *and* α_k *satisfies the* strong Wolfe conditions. *Suppose that* f *is bounded below, continuously differentiable, and* L-smooth. *Then*

$$\sum_{k\geq 0} \cos^2\theta_k \, \|\nabla f_k\|^2 < \infty.$$

Global Convergence of Quasi-Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the approximations to the Hessians $\tilde{H}_k$ are positive definite with a uniformly bounded condition number:

$$\left|\left|\tilde{H}_k\right|\right| \left|\left|\tilde{H}_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Global Convergence of Quasi-Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the approximations to the Hessians $\tilde{H}_k$ are positive definite with a uniformly bounded condition number:

$$\left|\left|\tilde{H}_k\right|\right| \left|\left|\tilde{H}_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Then θ_k between $p_k = -\tilde{H}_k^{-1}\nabla f_k$ and $-\nabla f_k$ and satisfies

$$\cos\theta_k \geq 1/M$$

Global Convergence of Quasi-Newton's Method

Assume that all α_k satisfy strong Wolfe conditions.

Assume that the approximations to the Hessians $\tilde{H}_k$ are positive definite with a uniformly bounded condition number:

$$\left|\left|\tilde{H}_k\right|\right| \left|\left|\tilde{H}_k^{-1}\right|\right| \leq M \quad \text{for all } k$$

Then θ_k between $p_k = -\tilde{H}_k^{-1}\nabla f_k$ and $-\nabla f_k$ and satisfies

$$\cos\theta_k \geq 1/M$$

Thus, under the assumptions of Zoutendijk's theorem, we obtain

$$\frac{1}{M^2}\sum_{k\geq 0}\|\nabla f_k\|^2 \leq \sum_{k\geq 0}\cos^2\theta_k\,\|\nabla f_k\|^2 < \infty$$

which implies that $\lim_{k\to\infty}||\nabla f_k|| = 0$.

Behavior of BFGS

- It may happen that $\tilde{H}_k$ becomes a poor approximation of the Hessian H_k. If, e.g., $y_k^\top$ is tiny, then $\tilde{H}_{k+1}$ will be huge.

  However, it has been proven experimentally that if $\tilde{H}_k$ wrongly estimates the curvature of f and this estimate slows down the iteration, then the approximation will tend to correct the bad Hessian approximations.

  The above self-correction works only if an appropriate line search is performed (strong Wolfe conditions).

Behavior of BFGS

- It may happen that $\tilde{H}_k$ becomes a poor approximation of the Hessian H_k. If, e.g., $y_k^\top$ is tiny, then $\tilde{H}_{k+1}$ will be huge.

  However, it has been proven experimentally that if $\tilde{H}_k$ wrongly estimates the curvature of f and this estimate slows down the iteration, then the approximation will tend to correct the bad Hessian approximations.

  The above self-correction works only if an appropriate line search is performed (strong Wolfe conditions).
- There are more sophisticated ways of setting the initial Hessian approximation H_0.

  See Numerical Optimization, Nocedal & Wright, page 201.

Quasi-Newton Methods - Comments

- Each iteration is performed for $\mathcal{O}(n^2)$ operations as opposed to $\mathcal{O}(n^3)$ for methods involving solutions of linear systems.

Quasi-Newton Methods - Comments

- Each iteration is performed for $\mathcal{O}(n^2)$ operations as opposed to $\mathcal{O}(n^3)$ for methods involving solutions of linear systems.
- There is even a memory-limited variant (L-BFGS) that uses only information from past m steps, and its single iteration complexity is $\mathcal{O}(mn)$.

Quasi-Newton Methods - Comments

- Each iteration is performed for $\mathcal{O}(n^2)$ operations as opposed to $\mathcal{O}(n^3)$ for methods involving solutions of linear systems.
- There is even a memory-limited variant (L-BFGS) that uses only information from past m steps, and its single iteration complexity is $\mathcal{O}(mn)$.
- Compared with Newton's method, no second derivatives are computed.

Quasi-Newton Methods - Comments

- Each iteration is performed for $\mathcal{O}(n^2)$ operations as opposed to $\mathcal{O}(n^3)$ for methods involving solutions of linear systems.
- There is even a memory-limited variant (L-BFGS) that uses only information from past m steps, and its single iteration complexity is $\mathcal{O}(mn)$.
- Compared with Newton's method, no second derivatives are computed.
- Local superlinear convergence can be proved under specific conditions.

  Compare with local quadratic convergence of Newton's method and linear convergence of gradient descent.

Limited-Memory BFGS

Limited-Memory BFGS (L-BFGS)

When the number of design variables is extensive, working with the whole Hessian inverse approximation matrix might not be practical.

This motivates limited-memory quasi-Newton methods,

In addition, these methods also improve the computational efficiency of medium-sized problems (hundreds or thousands of design variables) with minimal sacrifice in accuracy.

L-BFGS

Recall that we compute iteratively the approximation to the inverse Hessian by

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

However, eventually, we are interested in

$$p_k = H_k^{-1} \nabla f$$

Note that given the sequences $s_1, \ldots, s_k$ and $y_1, \ldots, y_k$ and H_0^{-1} we can recursively compute H_{k+1}^{-1} for every k.

What if we limit the sequences in memory to just m last elements:

$$s_{k-m+1}, s_{k-m+2}, \ldots, s_k \qquad y_{k-m+1}, y_{k-m+2}, \ldots, y_k$$

In practice, m between 5 and 20 is usually sufficient. We also initialize the recurrence with the last iterate:

L-BFGS

Let us rewrite the BFGS update formula as follows:

$$\tilde{H}_{k+1}^{-1} = V_k^T \tilde{H}_k^{-1} V_k + \rho_k s_k s_k^\top$$

where

$$\rho_k = s_k^\top y_k \qquad \text{and} \qquad V_k = I - \rho_k s_k y_k^\top$$

$$s_k = x_{k+1} - x_k \qquad \text{and} \qquad y_k = \nabla f_{k+1} - \nabla f_k$$

By substitution, we obtain

$$\begin{aligned}
\tilde{H}_k^{-1} = &\left(V_{k-1}^T \cdots V_{k-m}^T\right) \tilde{H}_k^0 \left(V_{k-m} \cdots V_{k-1}\right) \\
&+ \rho_{k-m} \left(V_{k-1}^T \cdots V_{k-m+1}^T\right) s_{k-m} s_{k-m}^T \left(V_{k-m+1} \cdots V_{k-1}\right) \\
&+ \rho_{k-m+1} \left(V_{k-1}^T \cdots V_{k-m+2}^T\right) s_{k-m+1} s_{k-m+1}^T \left(V_{k-m+2} \cdots V_k \right. \\
&+ \cdots \\
&+ \rho_{k-1} s_{k-1} s_{k-1}^T
\end{aligned}$$

L-BFGS Algorithm

Algorithm 8 L-BFGS two-loop recursion

Input: : $s_{k-1}, \dots, s_{k-m}$ and $y_{k-1}, \dots, y_{k-m}$

Output: : p_k the search direction $-\tilde{H}_k^{-1} \nabla f_k$

$q \leftarrow \nabla f_k$

for $i = k-1, k-2, \dots, k-m$ **do**

  $\alpha_i \leftarrow \rho_i s_i^T q$

  $q \leftarrow q - \alpha_i y_i$

end for

$r \leftarrow H_k^0 q$

for $i = k-m, k-m+1, \dots, k-1$ **do**

  $\beta \leftarrow \rho_i y_i^T r$

  $r \leftarrow r + s_i(\alpha_i - \beta)$

end for

stop with result $\tilde{H}_k^{-1} \nabla f_k = r$

L-BFGS Algorithm

Algorithm 9 L-BFGS

Choose starting point x_0, integer $m > 0$

$k \leftarrow 0$

repeat

  Choose H_k^0 e.g. $\frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}$

  Compute $p_k \leftarrow -H_k \nabla f_k$ using the previous algorithm

  Compute $x_{k+1} \leftarrow x_k + \alpha_k p_k$, where α_k is chosen to satisfy the strong Wolfe conditions

  if $k > m$ **then**

    Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage

  end if

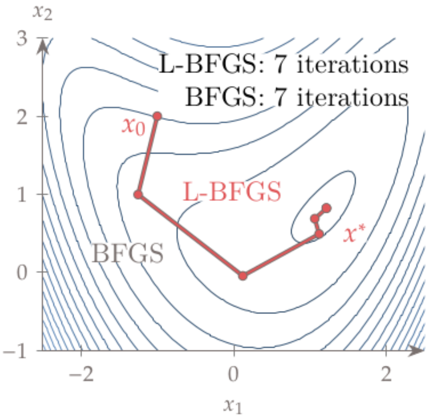
  Compute and save $s_k \leftarrow x_{k+1} - x_k$, $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$

  $k \leftarrow k + 1$

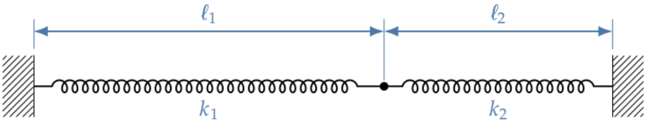
until convergence

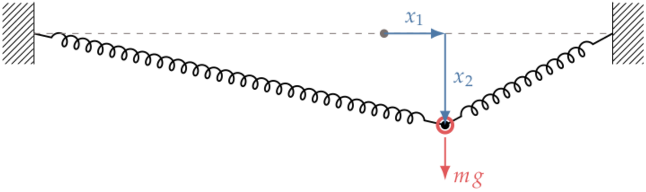
$$f(x_1, x_2) = (1 - x_1)^2 + (1 - x_2)^2 + \frac{1}{2}\left(2x_2 - x_1^2\right)^2$$

Stopping: $||\nabla f||_\infty \leq 10^{-6}$.



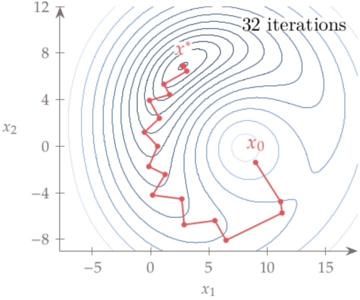
In L-BFGS, the memory length m was 5. The results are similar.



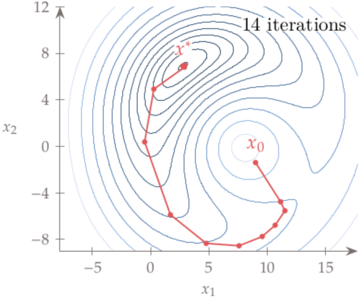


$$
\begin{aligned}
f(x_1, x_2) =& \frac{1}{2} k_1 \left(\sqrt{(\ell_1 + x_1)^2 + x_2^2} - \ell_1 \right)^2 \\
&+ \frac{1}{2} k_2 \left(\sqrt{(\ell_2 - x_1)^2 + x_2^2} - \ell_2 \right)^2 - mg x_2
\end{aligned}
$$

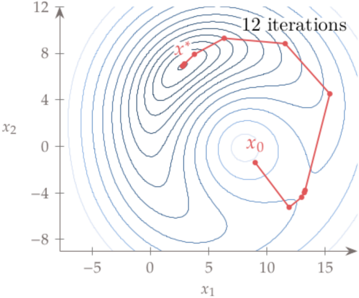
Here $\ell_1 = 12, \ell_2 = 8, k_1 = 1, k_2 = 10, mg = 7$



Steepest descent

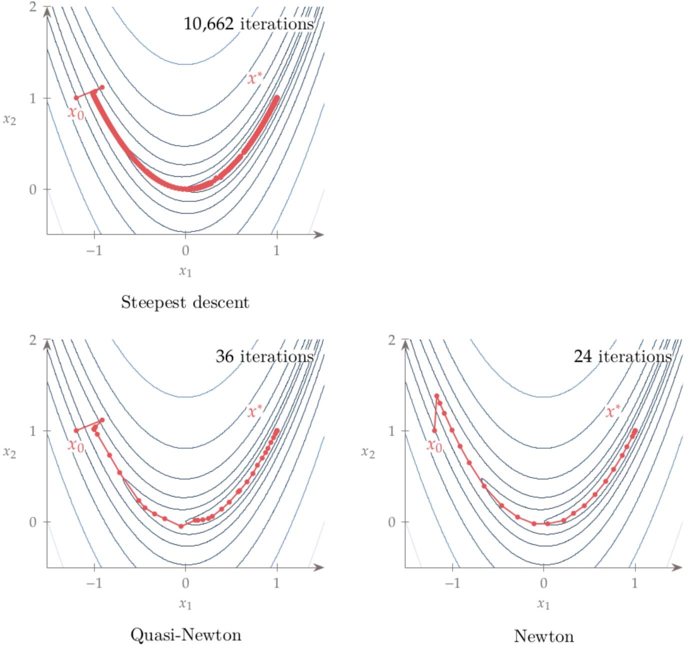


Quasi-Newton



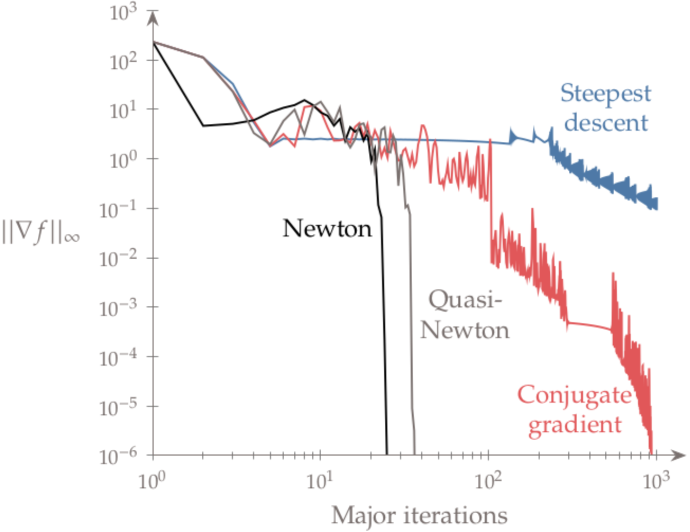
Newton

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



Rosenbrock:

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



Computational Complexity

Algorithm	Computational Complexity
Steepest Descent	$O(n)$ per iteration
Newton's Method	$O(n^3)$ to compute Hessian and solve system
BFGS	$O(n^2)$ to update Hessian approximation

Table: Summary of the computational complexity for each optimization algorithm.

- Steepest Descent: Simple but often slow, requiring many iterations.
- Newton's Method: Fast convergence but expensive per iteration.
- BFGS: Quasi-Newton, no Hessian needed, good speed and iteration count balance.

Constrained Optimization

Constrained Optimization Problem

Recall that the constrained optimization problem is

$$\begin{array}{rll} \text{minimize} & f(x) & \\ \text{by varying} & x & \\ \text{subject to} & g_i(x) \leq 0 & i = 1, \ldots, n_g \\ & h_j(x) = 0 & j = 1, \ldots, n_h \end{array}$$

x^* is now a *constrained minimizer* if

$$f(x^*) \leq f(x) \quad \text{for all} \quad x \in \mathcal{F}$$

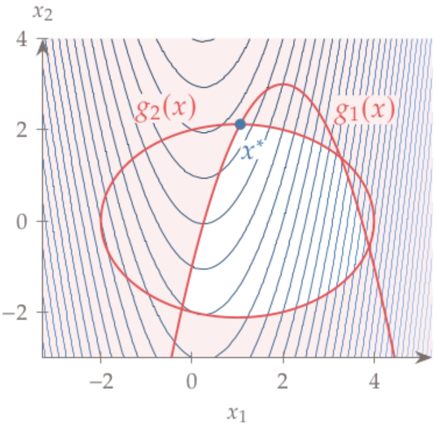
where $\mathcal{F}$ is the feasibility region

$$\mathcal{F} = \{x \mid g_i(x) \leq 0, h_j(x) = 0, i = 1, \ldots, n_g, j = 1, \ldots, n_h\}$$

Thus, to find a constrained minimizer, we have to inspect unconstrained minima of f inside of $\mathcal{F}$ and points along the boundary of $\mathcal{F}$.

COP - Example

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1^2 - \frac{1}{2}x_1 - x_2 - 2 \\ \text{subject to} & g_1(x_1, x_2) = x_1^2 - 4x_1 + x_2 + 1 \leq 0 \\ & g_2(x_1, x_2) = \frac{1}{2}x_1^2 + x_2^2 - x_1 - 4 \leq 0 \end{array}$$



Equality Constraints

Let us restrict our problem only to the equality constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{by varying} \quad & x \\ \text{subject to} \quad & h_j(x) = 0 \quad j = 1, \ldots, n_h \end{aligned}$$

Assume that f and h_j have continuous second derivatives.

Now, we try to imitate the theory from the unconstrained case and characterize minima using gradients.

This time, we must consider the gradient of f and h_j.

Half-Space of Decrease

Consider the first-order Taylor approximation of f at x

$$f(x+p) \approx f(x) + \nabla f(x)^\top p$$

Half-Space of Decrease

Consider the first-order Taylor approximation of f at x

$$f(x+p) \approx f(x) + \nabla f(x)^\top p$$

Note that if x^* is an unconstrained minimum of f, then

$$f(x^* + p) \geq f(x^*)$$

for all p small enough.

Half-Space of Decrease

Consider the first-order Taylor approximation of f at x

$$f(x+p) \approx f(x) + \nabla f(x)^\top p$$

Note that if x^* is an unconstrained minimum of f, then

$$f(x^* + p) \geq f(x^*)$$

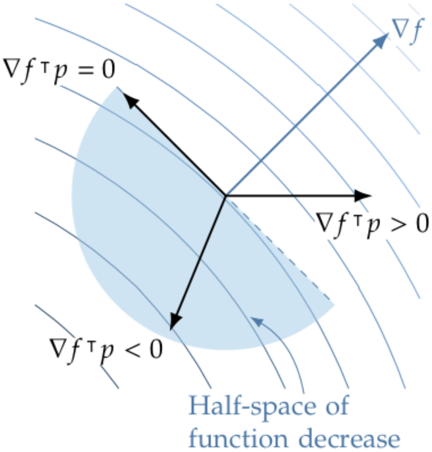
for all p small enough.

Together with the Taylor approximation, we obtain

$$f(x^*) + \nabla f(x^*)^\top p \geq f(x^*)$$

and hence

$$\nabla f(x^*)^\top p \geq 0$$



The hyperplane defined by $\nabla f^\top p = 0$ contains directions p of *zero variation* in f.

In the unconstrained case, x^* is minimizer only if $\nabla f(x^*) = 0$ because otherwise there would be a direction p satisfying $\nabla f(x^*)p < 0$, a *decrease direction*.

Decrease Direction in COP

In COP, p is a decrease direction in $x \in \mathcal{F}$ if $\nabla f(x)^\top p < 0$ and if p is a *feasible direction*!

I.e., point into the feasible region.

Decrease Direction in COP

In COP, p is a decrease direction in $x \in \mathcal{F}$ if $\nabla f(x)^\top p < 0$ and if p is a *feasible direction*!

I.e., point into the feasible region. How do we characterize feasible directions?

Decrease Direction in COP

In COP, p is a decrease direction in $x \in \mathcal{F}$ if $\nabla f(x)^\top p < 0$ and if p is a *feasible direction*!

I.e., point into the feasible region. How do we characterize feasible directions?

Consider Taylor approximation of h_j for all j:

$$h_j(x + p) \approx h_j(x) + \nabla h_j(x)^\top p$$

Decrease Direction in COP

In COP, p is a decrease direction in $x \in \mathcal{F}$ if $\nabla f(x)^\top p < 0$ and if p is a *feasible direction*!

I.e., point into the feasible region. How do we characterize feasible directions?

Consider Taylor approximation of h_j for all j:

$$h_j(x + p) \approx h_j(x) + \nabla h_j(x)^\top p$$

Assuming $x \in \mathcal{F}$, we have $h_j(x) = 0$ for all j and thus

$$h_j(x + p) \approx \nabla h_j(x)^\top p$$

Decrease Direction in COP

In COP, p is a decrease direction in $x \in \mathcal{F}$ if $\nabla f(x)^\top p < 0$ and if p is a *feasible direction*!

I.e., point into the feasible region. How do we characterize feasible directions?

Consider Taylor approximation of h_j for all j:

$$h_j(x + p) \approx h_j(x) + \nabla h_j(x)^\top p$$

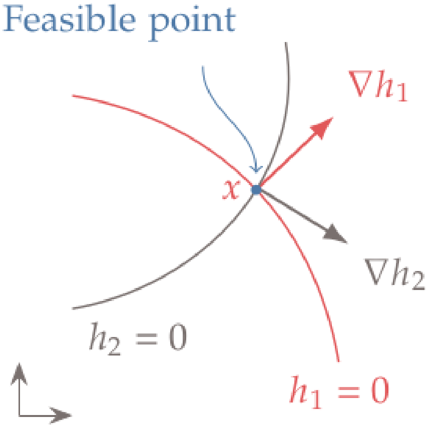
Assuming $x \in \mathcal{F}$, we have $h_j(x) = 0$ for all j and thus

$$h_j(x + p) \approx \nabla h_j(x)^\top p$$

As p is a feasible direction iff $h_j(x + p) = 0$, we obtain that

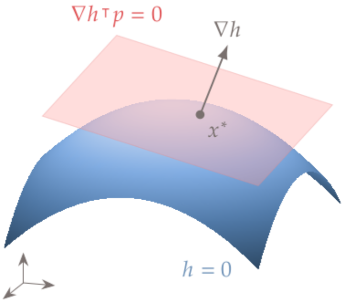
$$p \text{ is a } \textit{feasible direction} \quad \text{iff} \quad \nabla h_j(x)^\top p = 0 \text{ for all } j$$

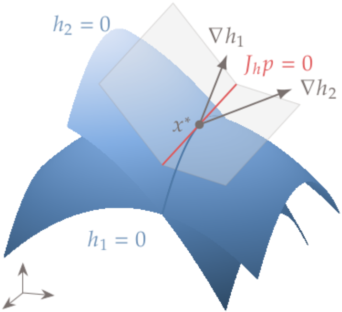
Feasible Points and Directions



Here, the only feasible direction at x is $p = 0$.

Feasible Points and Directions





Here the feasible directions at x^* point along the red line, i.e.,

$$\nabla h_1(x^*)p = 0 \qquad \nabla h_2(x^*)p = 0$$

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x) p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x) p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.
  - $\nabla f(x) p < 0$, then moving a short step in the direction p decreases f and stays in $\mathcal{F}$.

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x) p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.
  - $\nabla f(x) p < 0$, then moving a short step in the direction p decreases f and stays in $\mathcal{F}$.
  - $\nabla f(x) p = 0$, then moving a short step in the direction p does not change f and stays $\mathcal{F}$.

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x)p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.
  - $\nabla f(x)p < 0$, then moving a short step in the direction p decreases f and stays in $\mathcal{F}$.
  - $\nabla f(x)p = 0$, then moving a short step in the direction p does not change f and stays $\mathcal{F}$.

To be a minimizer, x^* must be feasible and every direction satisfying $h_j(x^*)^\top p = 0$ for all j must also satisfy $\nabla f(x^*)^\top p \geq 0$.

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x) p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.
  - $\nabla f(x) p < 0$, then moving a short step in the direction p decreases f and stays in $\mathcal{F}$.
  - $\nabla f(x) p = 0$, then moving a short step in the direction p does not change f and stays $\mathcal{F}$.

To be a minimizer, x^* must be feasible and every direction satisfying $h_j(x^*)^\top p = 0$ for all j must also satisfy $\nabla f(x^*)^\top p \geq 0$.

Note that if p is a feasible direction, then $-p$ is also, and thus $\nabla f(x^*)^\top (-p) \geq 0$. So finally,

Necessary Condition for Constrained Minima

Consider a direction p. Observe that

- If $h_j(x)^\top p \neq 0$, then moving a short step in the direction p violates the constraint $h_j(x) = 0$.
- If $h_j(x)^\top p = 0$ for all j and
  - $\nabla f(x)p > 0$, then moving a short step in the direction p increases f and stays in $\mathcal{F}$.
  - $\nabla f(x)p < 0$, then moving a short step in the direction p decreases f and stays in $\mathcal{F}$.
  - $\nabla f(x)p = 0$, then moving a short step in the direction p does not change f and stays $\mathcal{F}$.

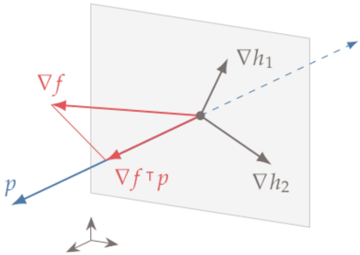
To be a minimizer, x^* must be feasible and every direction satisfying $h_j(x^*)^\top p = 0$ for all j must also satisfy $\nabla f(x^*)^\top p \geq 0$.

Note that if p is a feasible direction, then $-p$ is also, and thus $\nabla f(x^*)^\top(-p) \geq 0$. So finally,

If x^* is a *constrained minimizer*, then

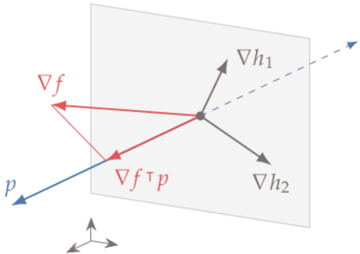
$$\nabla f(x^*)^\top p = 0 \text{ for all } p \text{ satisfying } (\forall j : \nabla h_j(x^*)^\top p = 0)$$

Lagrange Multipliers



Left: f increases along p. **Right:** f does not change along p.

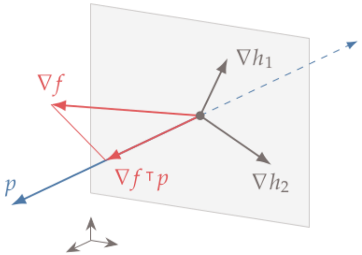
Lagrange Multipliers



Left: f increases along p. **Right:** f does not change along p.

Observe that at an optimum, ∇f lies in the space spanned by the gradients of constraint functions.

Lagrange Multipliers



Left: f increases along p. **Right:** f does not change along p.

Observe that at an optimum, ∇f lies in the space spanned by the gradients of constraint functions.

There are *Lagrange multipliers* λ_1, λ_2 satisfying

$$\nabla f(x^*) = -(\lambda_1 \nabla h_1 + \lambda_2 \nabla h_2)$$

The minus sign is arbitrary for equality constraints but will be significant when dealing with inequality constraints.

Lagrange Multipliers

We know that if x^* is a constrained minimizer, then.

$$\nabla f(x^*)^\top p = 0 \text{ for all } p \text{ satisfying } (\forall j : \nabla h_j(x^*)^\top p = 0)$$

Lagrange Multipliers

We know that if x^* is a constrained minimizer, then.

$$\nabla f(x^*)^\top p = 0 \text{ for all } p \text{ satisfying } (\forall j : \nabla h_j(x^*)^\top p = 0)$$

But then, from the geometry of the problem, we obtain

Theorem 5

Consider the COP with only equality constraints and f and all h_j twice continuously differentiable.
Assume that x^* is a constrained minimizer and that x^* is regular,
which means that $\nabla h_j(x^*)$ are linearly independent.
Then there are $\lambda_1, \ldots, \lambda_{n_h} \in \mathbb{R}$ satisfying

$$\nabla f(x^*) = -\sum_{j=1}^{n_h} \lambda_j \nabla h_j(x^*)$$

The coefficients $\lambda_1, \ldots, \lambda_{n_h}$ are called *Lagrange multipliers*.

Lagrangian Function

Try to transform the constrained problem into an unconstrained one by moving the constraints $h_j(x) = 0$ into the objective.

Lagrangian Function

Try to transform the constrained problem into an unconstrained one by moving the constraints $h_j(x) = 0$ into the objective.

Consider *Lagrangian function* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{n_h} \to \mathbb{R}$ defined by

$$\mathcal{L}(x, \lambda) = f(x) + h(x)^\top \lambda \qquad \text{here} \qquad h(x) = (h_1(x), \ldots, h_{n_h}(x))^\top$$

Lagrangian Function

Try to transform the constrained problem into an unconstrained one by moving the constraints $h_j(x) = 0$ into the objective.

Consider *Lagrangian function* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{n_h} \to \mathbb{R}$ defined by

$$\mathcal{L}(x, \lambda) = f(x) + h(x)^\top \lambda \qquad \text{here} \qquad h(x) = (h_1(x), \ldots, h_{n_h}(x))^\top$$

Note that the stationary point of $\mathcal{L}$ gives us the Lagrange multipliers:

$$\nabla_x \mathcal{L} = \nabla f(x) + \sum_{j=1}^{n_h} \lambda_j \nabla h_j(x)$$

$$\nabla_\lambda \mathcal{L} = h(x)$$

Lagrangian Function

Try to transform the constrained problem into an unconstrained one by moving the constraints $h_j(x) = 0$ into the objective.

Consider *Lagrangian function* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{n_h} \to \mathbb{R}$ defined by

$$\mathcal{L}(x, \lambda) = f(x) + h(x)^\top \lambda \qquad \text{here} \qquad h(x) = (h_1(x), \ldots, h_{n_h}(x))^\top$$

Note that the stationary point of $\mathcal{L}$ gives us the Lagrange multipliers:

$$\nabla_x \mathcal{L} = \nabla f(x) + \sum_{j=1}^{n_h} \lambda_j \nabla h_j(x)$$

$$\nabla_\lambda \mathcal{L} = h(x)$$

Now putting $\nabla \mathcal{L}(x) = 0$, we obtain precisely the above properties of the constrained minimizer:

$$h(x) = 0 \qquad \text{and} \qquad \nabla f(x) = -\sum_{j=1}^{n_h} \lambda_j \nabla h_j(x)$$

However, we cannot use the unconstrained optimization methods here because searching for a minimizer in x asks for a maximizer in λ.

$$
\begin{array}{ll}
\underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\
\text{subject to} & h(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0
\end{array}
$$

The Lagrangian function

$$
\mathcal{L}(x_1, x_2, \lambda) = x_1 + 2x_2 + \lambda \left(\frac{1}{4}x_1^2 + x_2^2 - 1 \right)
$$

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\ \text{subject to} & h(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0 \end{array}$$

The Lagrangian function

$$\mathcal{L}(x_1, x_2, \lambda) = x_1 + 2x_2 + \lambda\left(\frac{1}{4}x_1^2 + x_2^2 - 1\right)$$

Differentiating this to get the first-order optimality conditions,

$$\frac{\partial \mathcal{L}}{\partial x_1} = 1 + \frac{1}{2}\lambda x_1 = 0 \qquad \frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2\lambda x_2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0.$$

$$\begin{aligned} \underset{x_1, x_2}{\text{minimize}} \quad & f(x_1, x_2) = x_1 + 2x_2 \\ \text{subject to} \quad & h(x_1, x_2) = \tfrac{1}{4}x_1^2 + x_2^2 - 1 = 0 \end{aligned}$$

The Lagrangian function

$$\mathcal{L}(x_1, x_2, \lambda) = x_1 + 2x_2 + \lambda \left(\frac{1}{4}x_1^2 + x_2^2 - 1 \right)$$

Differentiating this to get the first-order optimality conditions,

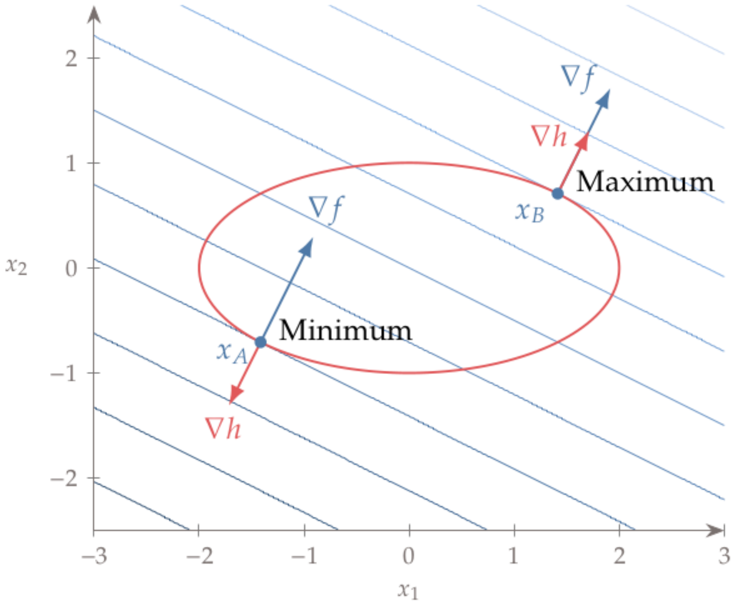
$$\frac{\partial \mathcal{L}}{\partial x_1} = 1 + \frac{1}{2}\lambda x_1 = 0 \qquad \frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2\lambda x_2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0.$$

Solving these three equations for the three unknowns (x_1, x_2, λ), we obtain two possible solutions:

$$x_A = (x_1, x_2) = (-\sqrt{2}, -\sqrt{2}/2), \quad \lambda_A = \sqrt{2}$$

$$x_B = (x_1, x_2) = (\sqrt{2}, \sqrt{2}/2), \quad \lambda_A = -\sqrt{2}$$



∇f
∇h
Maximum
x_B
∇f
Minimum
x_A
∇h
2
1
0
−1
−2
x_2
−3
−2
−1
0
1
2
3
x_1

Second-Order Sufficient Conditions

As in the unconstrained case, the first-order conditions characterize any "stable" point (minimum, maximum, saddle).

Consider *Lagrangian Hessian*:

$$H_{\mathcal{L}}(x, \lambda) = H_f(x) + \sum_{j=1}^{n_h} \lambda_j H_{h_j}(x)$$

Here H_f is the Hessian of f, and each H_{h_j} is the Hessian of h_j.

The second-order sufficient conditions are as follows: Assume x^* is regular and feasible. Also, assume that there is λ s.t.

$$\nabla f(x^*) = \sum_{j=1}^{n_h} -\lambda_j \nabla h_j(x^*)$$

and that

$$p^\top H_{\mathcal{L}}(x^*, \lambda) p > 0 \text{ for all } p \text{ satisfying } (\forall j : \nabla h_j(x^*)^\top p = 0)$$

Then, x^* is a constrained minimizer of f.

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\ \text{subject to} & h(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0 \end{array}$$

The Lagrangian function

$$\mathcal{L}(x_1, x_2, \lambda) = x_1 + 2x_2 + \lambda\left(\frac{1}{4}x_1^2 + x_2^2 - 1\right)$$

Differentiating this to get the first-order optimality conditions,

$$\frac{\partial \mathcal{L}}{\partial x_1} = 1 + \frac{1}{2}\lambda x_1 = 0 \qquad \frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2\lambda x_2 = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0.$$

Solving these three equations for the three unknowns (x_1, x_2, λ), we obtain two possible solutions:

$$x_A = (x_1, x_2) = (-\sqrt{2}, -\sqrt{2}/2), \quad \lambda_A = \sqrt{2}$$
$$x_B = (x_1, x_2) = (\sqrt{2}, \sqrt{2}/2), \quad \lambda_A = -\sqrt{2}$$

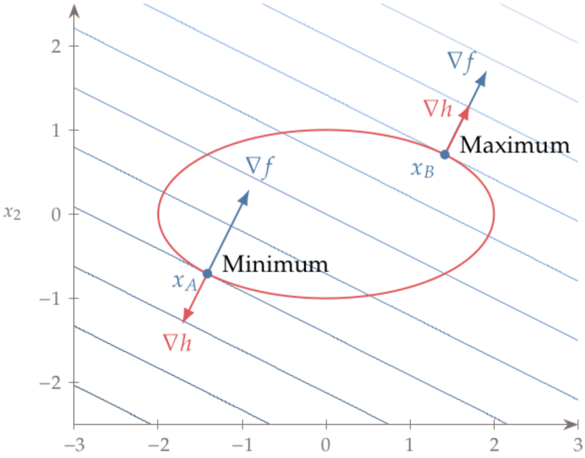
Which one is a minimum?

Second Order Conditions - Example

Compute the Hessian:

$$H_{\mathcal{L}} = \begin{pmatrix} \frac{1}{2}\lambda & 0 \\ 0 & 2\lambda \end{pmatrix}$$

The Hessian is positive definite only for the case $\lambda_A = \sqrt{2}$.



$$
\begin{aligned}
\underset{x_1, x_2}{\text{minimize}} \quad & f(x_1, x_2) = x_1^2 + 3(x_2 - 2)^2 \\
\text{subject to} \quad & h(x_1, x_2) = \beta x_1^2 - x_2 = 0,
\end{aligned}
$$

where β is a parameter. The Lagrangian for this problem is

$$
\mathcal{L}(x_1, x_2, \lambda) = x_1^2 + 3(x_2 - 2)^2 + \lambda\left(\beta x_1^2 - x_2\right).
$$

Differentiating for the first-order optimality conditions, we get

$$
\begin{aligned}
\nabla_x \mathcal{L} &= \begin{bmatrix} 2x_1(1 + \lambda\beta) \\ 6(x_2 - 2) - \lambda \end{bmatrix} = 0 \\
\nabla_\lambda \mathcal{L} &= \beta x_1^2 - x_2 = 0.
\end{aligned}
$$

Solving these three equations for the three unknowns (x_1, x_2, λ), the solution is $x_A = (0, 0)$, $\lambda_A = -12$, independent of β.

The Hessian of the Lagrangian,

$$
H_{\mathcal{L}} = \begin{bmatrix} 2(1 - 12\beta) & 0 \\ 0 & 6 \end{bmatrix}
$$

We need this to be positive definite in feasible directions.

$$\begin{array}{ll}\underset{x_1,x_2}{\text{minimize}} & f\left(x_1,x_2\right)=x_1^2+3\left(x_2-2\right)^2 \\ \text{subject to} & h\left(x_1,x_2\right)=\beta x_1^2-x_2=0,\end{array}$$

The Hessian of the Lagrangian,

$$H_{\mathcal{L}}=\left[\begin{array}{cc}2(1-12\beta) & 0 \\ 0 & 6\end{array}\right]$$

What are the feasible directions?

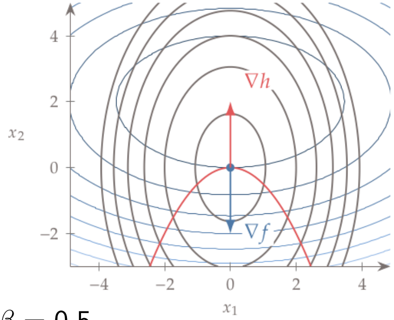
$\nabla h=(2\beta x_1,-1)$ and thus $\nabla h(x^*)=(0,-1)$.
Thus all p satisfying $\nabla h^\top p=0$ are $(\alpha,0)$ for $\alpha\in\mathbb{R}$.

Thus, for positive curvature in the feasible direction, we need

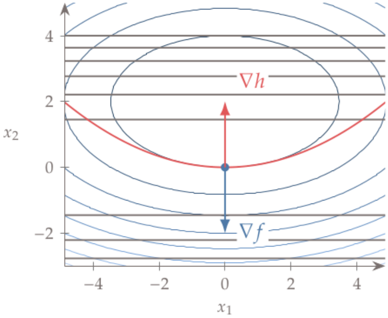
$$p^\top H_{\mathcal{L}}p=2\alpha^2(1-12\beta)>0$$

which is equivalent to $\beta<1/12$.

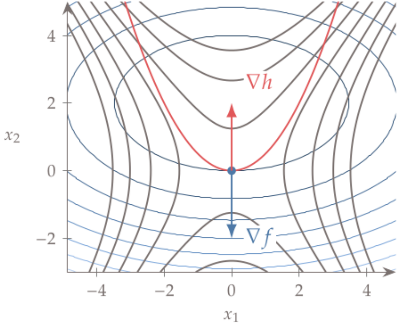
$\beta = -0.5$



$\beta = 1/12$



$\beta = 0.5$



Inequality Constraints

Recall that the constrained optimization problem is

$$
\begin{array}{rll}
\text{minimize} & f(x) & \\
\text{by varying} & x & \\
\text{subject to} & g_i(x) \leq 0 & i = 1, \dots, n_g \\
& h_j(x) = 0 & j = 1, \dots, n_h
\end{array}
$$

We say that a constraint $g_i(x) \leq 0$ is *active* for x if $g_i(x) = 0$, otherwise it is *inactive* for x.

As before, if x^* is a minimizer, any small step in a feasible direction p must not decrease f, i.e.,

$$\nabla f(x^*)^\top p \geq 0$$

How do we identify feasible directions for inequality constraints?

Feasible Directions

For inactive constraints, arbitrary direction p is feasible.

Feasible Directions

For inactive constraints, arbitrary direction p is feasible.

For active constraints $g_i(x) = 0$ we have p feasible at x if

$$g_i(x + p) \approx g_i(x) + \nabla g_i(x)^\top p \leq 0, \quad i = 1, \dots, n_g$$

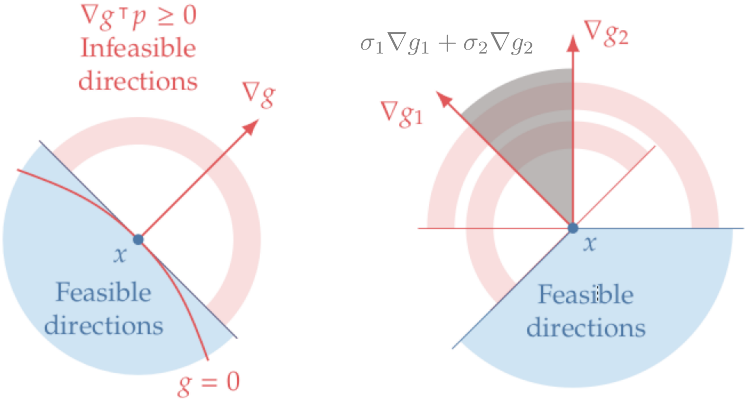
Feasible Directions

For inactive constraints, arbitrary direction p is feasible.

For active constraints $g_i(x) = 0$ we have p feasible at x if

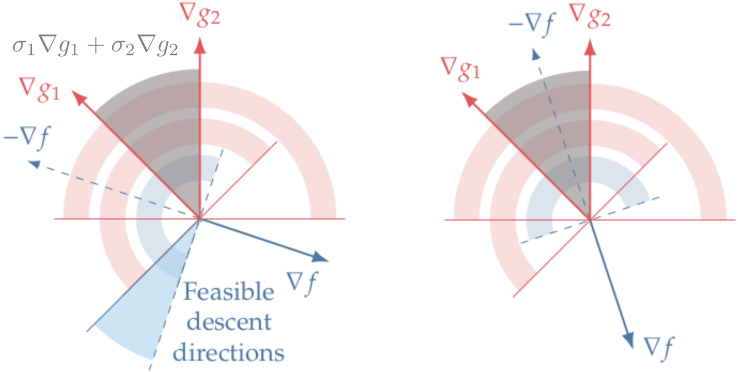
$$g_i(x+p) \approx g_i(x) + \nabla g_i(x)^\top p \le 0, \quad i = 1, \dots, n_g$$

thus p is *feasible* iff $\nabla g_i(x)^\top p \le 0$ for all active constr. $g_i(x) = 0$.



Lagrange Multipliers

When could f be decreased in a feasible direction?



Left: f decreases in the blue cone. **Right:** f does not decrease in any feasible direction.

At an optimum there are *Lagrange multipliers* $\sigma_1, \sigma_2 \geq 0$:

$$-\nabla f = \sigma_1 \nabla g_1 + \sigma_2 \nabla g_2$$

Lagrange Multipliers

We know that if x^* is a constrained minimizer, then

$$\nabla f(x)^\top p = 0 \quad \text{for all } p \text{ feasible at } x$$

Lagrange Multipliers

We know that if x^* is a constrained minimizer, then

$$\nabla f(x)^\top p = 0 \quad \text{for all } p \text{ feasible at } x$$

One can prove the following

Theorem 6

Consider the COP with f and all g_i, h_j twice continuously differentiable.
Assume that x^* is a constrained minimizer and that x^* is regular *which means that $\nabla g_i(x^*), \nabla h_j(x^*)$ are linearly independent.*
Then there are Lagrange multipliers $\lambda_1, \ldots, \lambda_{n_h} \in \mathbb{R}$ and $\sigma_1, \ldots, \sigma_{n_g} \in \mathbb{R}$ satisfying

$$-\nabla f(x^*) = \sum_{j=1}^{n_h} \lambda_j \nabla h_j(x^*) + \sum_{i=1}^{n_h} \sigma_i \nabla g_i(x^*) \qquad \textit{where } \sigma_i \geq 0$$

Lagrangian Function

Note that inequality $g_i(x) \leq 0$ can be equivalently expressed using a *slack variable* s_i by

$$g(x) + s_i^2 = 0$$

The Lagrangian function then generalizes from equality to inequality COP as follows.

$$\mathcal{L}(x, \lambda, \sigma, s) = f(x) + h(x)^\top \lambda + (g(x) + s \odot s)^\top \sigma$$

Here, $h(x) = (h_1(x), \ldots, h_{n_h}(x))^\top$, $g(x) = (g_1(x), \ldots, g_{n_g}(x))^\top$, $s = (s_1, \ldots, s_{n_g})$, and $\odot$ is the component-wise multiplication.

Now compute the stable point of $\mathcal{L}$ by considering

$$\nabla_x \mathcal{L} = 0$$
$$\nabla_\lambda \mathcal{L} = 0$$
$$\nabla_\sigma \mathcal{L} = 0$$
$$\nabla_s \mathcal{L} = 0$$

(see the whiteboard)

KKT

If x^* is a constrained minimizer and x^* is regular. Then there are λ, σ, s satisfying

$$\begin{aligned}
\frac{\partial f}{\partial x_\ell}(x^*) + \sum_{j=1}^{n_h} \lambda_j \frac{\partial h_j}{\partial x_\ell}(x^*) + \sum_{j=1}^{n_g} \sigma_j \frac{\partial g_j}{\partial x_\ell}(x^*) &= 0 \qquad \ell = 1, \dots, n \\
h_j(x^*) &= 0 \quad j = 1, \dots, n_h \\
g_i(x^*) + s_i^2 &= 0 \quad i = 1, \dots, n_g \\
2\sigma_i s_i &= 0 \quad i = 1, \dots, n_g \\
\sigma_i &\geq 0
\end{aligned}$$

So, solving the above system allows us to identify potential constrained minimizers.

KKT

If x^* is a constrained minimizer and x^* is regular. Then there are λ, σ, s satisfying

$$\begin{aligned}
\frac{\partial f}{\partial x_\ell}(x^*) + \sum_{j=1}^{n_h} \lambda_j \frac{\partial h_j}{\partial x_\ell}(x^*) + \sum_{j=1}^{n_g} \sigma_j \frac{\partial g_j}{\partial x_\ell}(x^*) &= 0 \qquad \ell = 1, \ldots, n \\
h_j(x^*) &= 0 \quad j = 1, \ldots, n_h \\
g_i(x^*) + s_i^2 &= 0 \quad i = 1, \ldots, n_g \\
2\sigma_i s_i &= 0 \quad i = 1, \ldots, n_g \\
\sigma_i &\geq 0
\end{aligned}$$

So, solving the above system allows us to identify potential constrained minimizers.

To decide whether x^* solving KKT is a minimizer, check whether

$$p^\top H_{\mathcal{L}}(x^*, \lambda) p > 0$$

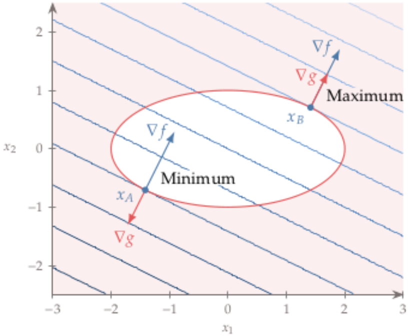
For all feasible directions p (similarly to the equality case).

Example

$$\underset{x_1,x_2}{\text{minimize}} \quad f(x_1,x_2) = x_1 + 2x_2$$
$$\text{subject to} \quad g(x_1,x_2) = \tfrac{1}{4}x_1^2 + x_2^2 - 1 \le 0.$$

The Lagrangian function for this problem is

$$\mathcal{L}(x_1,x_2,\sigma,s) = x_1 + 2x_2 + \sigma\left(\frac{1}{4}x_1^2 + x_2^2 - 1 + s^2\right)$$



Example

$$\frac{\partial \mathcal{L}}{\partial x_1} = 1 + \frac{1}{2}\sigma x_1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2\sigma x_2 = 0$$
$$\frac{\partial \mathcal{L}}{\partial \sigma} = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial s} = 2\sigma s = 0.$$

Example

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x_1} &= 1 + \frac{1}{2}\sigma x_1 = 0 \\
\frac{\partial \mathcal{L}}{\partial x_2} &= 2 + 2\sigma x_2 = 0 \\
\frac{\partial \mathcal{L}}{\partial \sigma} &= \frac{1}{4}x_1^2 + x_2^2 - 1 = 0 \\
\frac{\partial \mathcal{L}}{\partial s} &= 2\sigma s = 0.
\end{aligned}$$

Setting $\sigma = 0$ does not yield any solution. Setting $s = 0$ and $\sigma \neq 0$ we obtain

$$x_A = \begin{bmatrix} x_1 \\ x_2 \\ \sigma \end{bmatrix} = \begin{bmatrix} -\sqrt{2} \\ -\sqrt{2}/2 \\ \sqrt{2} \end{bmatrix}, \quad x_B = \begin{bmatrix} x_1 \\ x_2 \\ \sigma \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ \sqrt{2}/2 \\ -\sqrt{2} \end{bmatrix}$$

Example

$$\frac{\partial \mathcal{L}}{\partial x_1} = 1 + \frac{1}{2}\sigma x_1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial x_2} = 2 + 2\sigma x_2 = 0$$
$$\frac{\partial \mathcal{L}}{\partial \sigma} = \frac{1}{4}x_1^2 + x_2^2 - 1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial s} = 2\sigma s = 0.$$

Setting $\sigma = 0$ does not yield any solution. Setting $s = 0$ and $\sigma \neq 0$ we obtain

$$x_A = \begin{bmatrix} x_1 \\ x_2 \\ \sigma \end{bmatrix} = \begin{bmatrix} -\sqrt{2} \\ -\sqrt{2}/2 \\ \sqrt{2} \end{bmatrix}, \quad x_B = \begin{bmatrix} x_1 \\ x_2 \\ \sigma \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ \sqrt{2}/2 \\ -\sqrt{2} \end{bmatrix}$$

Now, σ must be non-negative, so only x_A is the solution. There is no feasible descent direction at x_A. We already know that the Hessian Lagrangian is positive definite, so this is a minimizer.

$$
\begin{array}{ll}
\underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\
\text{subject to} & g_1(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 \leq 0 \\
& g_2(x_2) = -x_2 \leq 0.
\end{array}
$$

The feasible region is the top half of the ellipse defined by g_1.

$$
\begin{array}{ll}
\underset{x_1,x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\
\text{subject to} & g_1(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 \leq 0 \\
& g_2(x_2) = -x_2 \leq 0.
\end{array}
$$

The feasible region is the top half of the ellipse defined by g_1.
The Lagrangian for this problem is

$$
\mathcal{L}(x, \sigma, s) = x_1 + 2x_2 + \sigma_1 \left(\frac{1}{4}x_1^2 + x_2^2 - 1 + s_1^2 \right) + \sigma_2 \left(-x_2 + s_2^2 \right).
$$

$$
\begin{array}{ll}
\underset{x_1, x_2}{\text{minimize}} & f(x_1, x_2) = x_1 + 2x_2 \\
\text{subject to} & g_1(x_1, x_2) = \frac{1}{4}x_1^2 + x_2^2 - 1 \leq 0 \\
& g_2(x_2) = -x_2 \leq 0.
\end{array}
$$

The feasible region is the top half of the ellipse defined by g_1.
The Lagrangian for this problem is

$$
\mathcal{L}(x, \sigma, s) = x_1 + 2x_2 + \sigma_1 \left(\frac{1}{4}x_1^2 + x_2^2 - 1 + s_1^2 \right) + \sigma_2 \left(-x_2 + s_2^2 \right).
$$

Differentiating the Lagrangian with respect to all the variables, we get the first-order optimality conditions,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x_1} &= 1 + \frac{1}{2}\sigma_1 x_1 = 0 & \qquad \frac{\partial \mathcal{L}}{\partial \sigma_2} &= -x_2 + s_2^2 = 0 \\
\frac{\partial \mathcal{L}}{\partial x_2} &= 2 + 2\sigma_1 x_2 - \sigma_2 = 0 & \qquad \frac{\partial \mathcal{L}}{\partial s_1} &= 2\sigma_1 s_1 = 0 \\
\frac{\partial \mathcal{L}}{\partial \sigma_1} &= \frac{1}{4}x_1^2 + x_2^2 - 1 + s_1^2 = 0 & \qquad \frac{\partial \mathcal{L}}{\partial s_2} &= 2\sigma_2 s_2 = 0.
\end{aligned}
$$

Assumption	Meaning	x_1	x_2	σ_1	σ_2	s_1	s_2	Point
$s_1 = 0$	g_1 is active	-2	0	1	2	0	0	x^*
$s_2 = 0$	g_2 is active	2	0	-1	2	0	0	x_C
$\sigma_1 = 0$	g_1 is inactive							
$\sigma_2 = 0$	g_2 is inactive		-	-	-	-	-	
$s_1 = 0$	g_1 is active	$\sqrt{2}$	$\frac{\sqrt{2}}{2}$	$-\sqrt{2}$	0	0	$2^{-\frac{1}{4}}$	x_B
$\sigma_2 = 0$	g_2 is inactive							
$\sigma_1 = 0$	g_1 is inactive							
$s_2 = 0$	g_2 is active			-	-