Methods that avoid calculating the Hessian

A disadvantage with the Newton method is that the Hessian has to be

- derived and implemented, which may introduce blunders,
- calculated, which requires $\mathcal{O}(n^2)$ operations,
- stored, which require storage for $n^2/2$ elements,
- ”inverted”, which requires $\mathcal{O}(n^3)$ operations.

Methods that avoid calculating the Hessian

Since any positive definite matrix will guarantee global convergence, there are many methods that replaces the Hessian in the Newton equation

$$\nabla^2 f(x_k)p = -\nabla f(x_k)$$

with some other positive definite matrix B_k

$$B_k p = -\nabla f(x_k).$$

The algorithms we have seen so far also fit in this framework:

	B_k
Modified Newton	$\nabla^2 f(x_k) + E$
Trust region	$\nabla^2 f(x_k) + \lambda I$
Gauss-Newton	$J(x_k)^T J(x_k)$
Levenberg-Marquardt	$J(x_k)^T J(x_k) + \lambda I$

We will study two more classes of such methods; *steepest descent* and *quasi-Newton*.

Steepest descent

The *steepest-descent* method uses the simplest Hessian approximation

$$B_k = I \quad \Rightarrow \quad p_k = -\nabla f(x_k).$$

Since all $\lambda_i(B_k) = 1$, the search direction p_k satisfies both the sufficient descent and gradient-related condition.

The steepest-descent method uses the cheapest possible approximiation of the Hessian since the search direction is obtained without any additional calculations. This makes the cost per iteration minimal.

Unfortunately, the downside is a poor convergence rate. It is possible to show that the steepest-descent method converges linearly with a convergence constant C bounded from above by

$$C_{sup} = \left(\frac{\kappa(Q) - 1}{\kappa(Q) + 1} \right)^2,$$

where $Q = \nabla^2 f(x_*)$.

Steepest descent

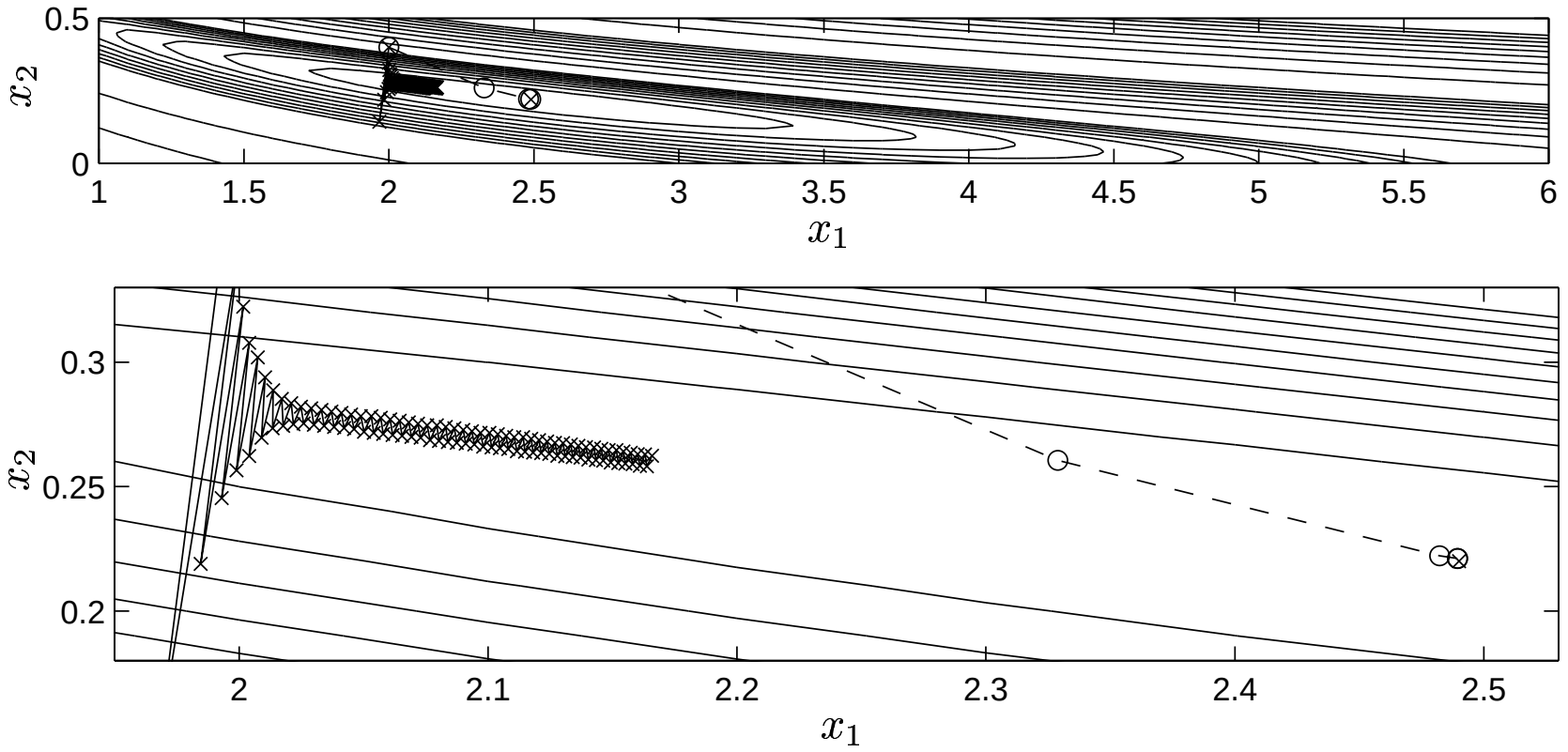
A table of convergence constants shows that even for moderate condition numbers, the convergence rate will be poor.

$\kappa(Q)$	1	10	100	1000	10000
C	0	0.6694	0.9608	0.9960	0.9996

The steepest-descent method should only be used for problems that are known to be well-conditioned.

Steepest-descent on the antelope problem

Comparison of the Gauss-Newton and the steepest-descent methods for the antelope problem ($\kappa(Q) \approx 500$).



Quasi-Newton methods

Quasi-Newton methods may be seen as a generalization of the one-dimensional secant method.

In the secant method the following approximation is used

$$\begin{aligned} f''(x_k) &\approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} \\ f''(x_k)(x_k - x_{k-1}) &\approx f'(x_k) - f'(x_{k-1}), \end{aligned}$$

which in multiple dimensions corresponds to

$$\nabla^2 f(x_k)(x_k - x_{k-1}) \approx \nabla f(x_k) - \nabla f(x_{k-1}).$$

From this we obtain the *secant condition*

$$B_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1}).$$

which will define the Quasi-Newton approximations B_k.

The secant condition

Since B_k has n^2 elements and the secant condition has only n equations, there are many degrees of freedom left to define B_k.

For the quadratic function $f(x) = \frac{1}{2}x^T Q x - c^T x$,

$$Q(x_k - x_{k-1}) = (Qx_k - c) - (Qx_{k-1} - c) = \nabla f(x_k) - \nabla f(x_{k-1}),$$

i.e. the Hessian Q satisfies the secant condition. Generally, we will require the approximation B_k to imitate the effect of the Hessian along a certain direction.

Introduce the definitions $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. With these definitions, the secant condition becomes

$$B_k s_{k-1} = y_{k-1}$$

or

$$B_{k+1} s_k = y_k.$$

Properties of quasi-Newton methods

An example of a quasi-Newton approximation is given by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k} \tag{1}$$

This formula illustrates several key features with quasi-Newton approximations.

- The secant condition will be satisfied indepedently of how B_k is chosen:

$$\begin{aligned} B_{k+1}s_k &= B_k s_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k} s_k \\ &= B_k s_k + \frac{(y_k - B_k s_k)((y_k - B_k s_k)^T s_k)}{(y_k - B_k s_k)^T s_k} \\ &= B_k s_k + (y_k - B_k s_k) = y_k. \end{aligned}$$

Properties of quasi-Newton methods

- The new approximation B_{k+1} is found by *updating* the old approximation B_k, and quasi-Newton approximations are often referred to as *update formulas*. As a starting approximation $B_0 = I$ may be used, but if a better approximation is available at a small cost, it should be used.
- The new approximation B_{k+1} can be obtained using $\mathcal{O}(n^2)$ operations since the update only involves products of vectors. More surprisingly may be that the solution of the equation system

$$B_{k+1}p = -\nabla f(x_{k+1})$$

  may be calculated with $\mathcal{O}(n^2)$ operations (in stead of the usual $\mathcal{O}(n^3)$), since it is possible to derive update formulas for the Cholesky factorization of B_k rather than B_k itself.

Common quasi-Newton methods

The most widely used quasi-Newton formula is Broyden-Fletcher-Goldfarb-Shanno (BFGS):

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

BFGS preserves symmetry and positive definiteness if $y_k^T s_k > 0$, which may be satisfied with a line search using the Wolfe condition.

BFGS belongs to the *Broyden class* of updates

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi(s_k^T B_k s_k) u_k u_k^T,$$

where ϕ is a scalar and

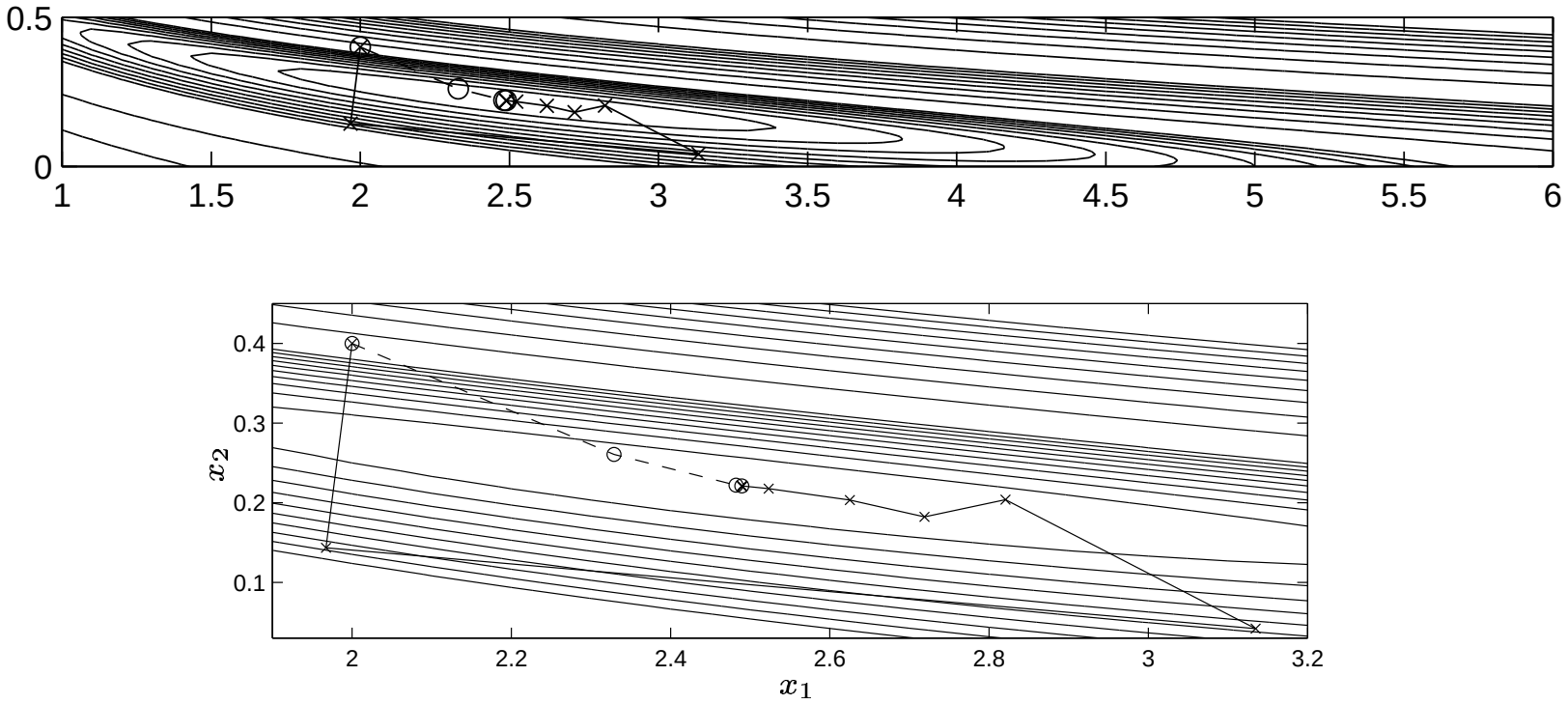
$$u_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}.$$

Selecting $\phi = 0$ gives BFGS. Selecting $\phi = 1$ gives an update known as Davidon-Fletcher-Powell (DFP).

Quasi-Newton methods typically converges super-linearly, since B_k becomes an increasingly good approximation of $\nabla^2 f(x_k)$, and the search direction p_k will thus approach the Newton direction.

BFGS on the antelope problem

Comparison of the Gauss-Newton and the BFGS methods for the antelope problem ($\kappa(Q) \approx 500$).



$$B_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} 15 & 105 \\ 105 & 776 \end{bmatrix}, B_2 = \begin{bmatrix} 6 & 31 \\ 31 & 289 \end{bmatrix}, B_3 = \begin{bmatrix} 8 & 45 \\ 45 & 398 \end{bmatrix}, B_4 = \begin{bmatrix} 9 & 66 \\ 66 & 671 \end{bmatrix}, B_5 = \begin{bmatrix} 10 & 72 \\ 72 & 621 \end{bmatrix},$$

$$B_6 = \begin{bmatrix} 10 & 80 \\ 80 & 700 \end{bmatrix}, B_7 = \begin{bmatrix} 10 & 78 \\ 78 & 691 \end{bmatrix}, B_8 = \begin{bmatrix} 10 & 75 \\ 75 & 664 \end{bmatrix}, B_9 = \begin{bmatrix} 10 & 74 \\ 74 & 652 \end{bmatrix}, \nabla^2 f(x_*) = \begin{bmatrix} 10 & 74 \\ 74 & 652 \end{bmatrix}.$$

Termination rules

In an ideal world, a minimization algorithm would terminate with a solution x_k satisfying $\nabla f(x_k) = 0$ and $\nabla^2 f(x_k)$ positiv semi-definite. However, due to e.g. the use of finite arithmetic, no algorithm is guaranteed to satisfy this condition in finite time.

As an alternative, we may consider using

$$||\nabla f(x_k)|| \leq \epsilon,$$

for some $\epsilon > 0$, as termination criteria. This absolute condition is however *scale dependent*, since a change of unit in f would rescale ∇f and affect the strength of the condition. E.g. a change of unit from km to m would correspond to a scaling of 10^3.

A small modification of the absolute termination criteria to a relative criteria

$$||\nabla f(x_k)|| \leq \epsilon |f(x_k)|$$

removes the scale dependency. This test will however cause problems if $f(x_*) \approx 0$, and may even be impossible to satisfy due to round-off errors.

Combining the two leads however to a useful test

$$||\nabla f(x_k)|| \leq \epsilon \left(1 + |f(x_k)|\right),$$

which will behave like an absolute test if $f(x_k) \approx 0$, and otherwise like a relative test.

Termination rules

As no perfect convergence test exists, it is common to combine multiple tests. In addition to tests on the gradient and hessian, we might require that the sequences $\{f(x_k)\}$ and $\{x_k\}$ should converge. The complete test set would then be as follows:

$$\begin{aligned} \|\nabla f(x_k)\| &\leq \epsilon_1 (1 + |f(x_k)|) \\ f(x_{k-1}) - f(x_k) &\leq \epsilon_2 (1 + |f(x_k)|) \\ \|x_{k-1} - x_k\| &\leq \epsilon_3 (1 + \|x_k\|) \\ \nabla^2 f(x_k) + \epsilon_4 I & \quad \text{positive semi-definite} \end{aligned}$$

If all values are calculated with the same precision ϵ_{mach}, the following values are common: $\epsilon_2 = \epsilon_{mach}, \epsilon_1 = \epsilon_3 = \sqrt{\epsilon_2}, \epsilon_4 = \epsilon_2 \|\nabla^2 f(x_k)\|$.

Termination rules for least squares problems

For the least squares problem

$$\min_x f(x) = \frac{1}{2} F(x)^T F(x),$$

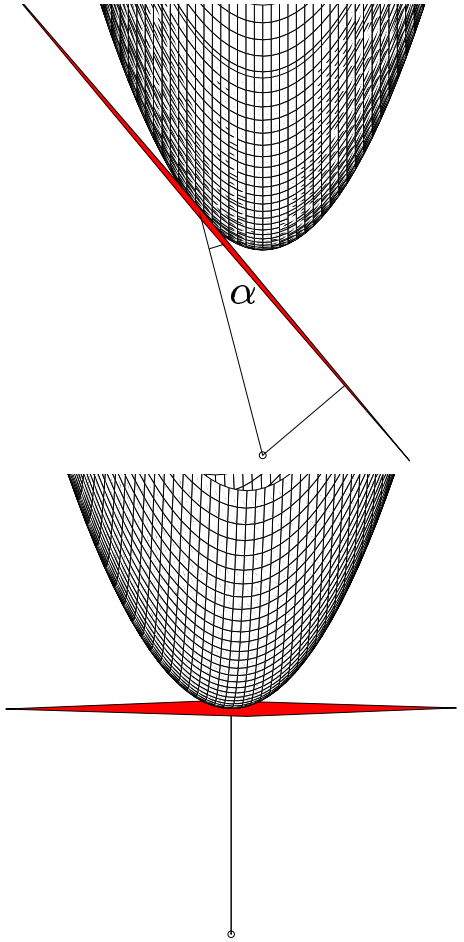
the test

$$||Jp|| = ||J(J^T J)^{-1} J^T F|| \leq \epsilon(1 + ||F||)$$

may be used instead of the gradient test. Since Jp and F belong to the same vector space, this test may be interpreted geometrically. The quotient

$$\frac{||Jp||}{||F||} = \cos\alpha,$$

describes the angle α between the residual F and the tangent plane to F. Close to the solution the residual will approach orthogonality with the tangent plane, i.e. α will approach $\pi/2$ and $||Jp||/||F||$ will approach 0.



Choice of norms, scaling

The tests on $\nabla f(x_k)$ and x_k are based on norms. Theoretically, any norm will do, but if $\nabla f(x_k) = (\gamma, \ldots, \gamma)^T$, then

$$\|\nabla f(x_k)\|_2 = \sqrt{n}|\gamma|$$

and

$$\|\nabla f(x_k)\|_\infty = |\gamma|.$$

Thus, for large problem the infinity norm is a good choice.

Another effect of norms is that different elements in x_k may be determined with different *relative* accuracy if the magnitude of x_k varies a lot.

If e.g. $x_* \approx [1 \ 10^{-4} \ 10^{-6}]^T$, the term $|x_1|$ will dominate $\|x_*\|$, which may cause the method to terminate before the other elements are determined with a good relative accuracy. However, if the approximate magnitude of the parameters are known in advance, a scaling of the variables or the use of a weighted norm

$$\|x\|_W = \|W^{1/2}x\|_p,$$

where $W^{1/2}$ is a diagonal matrix and p is any norm, may be a solution.