Recall: Problems with basic Newton

(i) Needs derivatives: may be impossible or expensive to compute.
(ii) Needs solution of a linear system each iteration. Demanding for large problems.
(iii) Converges only when starting close enough.
(iv) The Hessian may be singular or indefinite. Can cause problems to solve the linear system, or can cause Newton direction to be non descent.

Problem (iv) “cured” by Modified Cholesky, guaranteeing descent.
Globalization—line search (problem (iii))

- A general descent method: $x_{k+1} = x_k + \alpha_k p_k$, where $p_k$ is a descent direction.
- $p_k$ descent direction: $f(x_k + \alpha p_k) < f(x_k)$ for $\alpha > 0$ small enough.
- Too large $\alpha$ may cause $f(x_k + \alpha p_k) > f(x_k)$!
- The **line search** subproblem: $\min_{\alpha > 0} f(x_k + \alpha p_k)$.
- Not enough to demand $f(x_k + \alpha p_k) < f(x_k)$ (to guarantee convergence). Convergence theory needs a stronger condition, such as the **Armijo condition**:

  \[
  f(x_k + \alpha p_k) \leq f(x_k) + \alpha \mu p_k^T \nabla f(x_k)
  \]

  Very small $\mu = 10^{-4}$ OK.
  And additional condition like backtracking or Wolfe condition.
Damped Newton with modified Cholesky

Algorithm

(i) Choose initial guess $x_0$, tolerance $\epsilon$, and parameter $\mu$.

(ii) For $k = 0, 1, \ldots$

   (a) If $\|\nabla f(x_k)\| \leq \epsilon$, stop.

   (b) Compute modified factorization $LL^T$ of $\nabla^2 f(x_k)$ ($LL^T = E + \nabla^2 f(x_k)$ for some diagonal $E \geq 0$).

   Solve $LL^T p_k = -\nabla f(x_k)^T$.

   (c) $\alpha_k \leftarrow 1$

   While $f(x_k + \alpha_k p_k) > f(x_k) + \mu \alpha_k \nabla f(x_k) p_k$,

   $\alpha \leftarrow \alpha/2$

   Set $x_{k+1} \leftarrow x_k + \alpha p_k$. 
The steepest-descent method

Computing the Newton step with the crude approximation $\nabla^2 f(x_k) \approx I$ yields $p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)^T \approx -\nabla f(x_k)^T$

The negative gradient:
- Is perpendicular to the level curves
- Points in the steepest downhill direction

The steepest-descent direction is the best direction \textit{locally}.
The steepest-descent algorithm:

- Easy to implement.
- Does not need second derivatives: usable for large problems.
- Only linear convergence rate, even with exact line search.
- Terribly inefficient for *ill-conditioned* problem: large condition numbers of the Hessian.
- Sensitive to *scaling* of the variables (Newton’s method is *not* sensitive to scaling).
- Usually much better to use the conjugate gradient or a quasi-Newton algorithm.