Nonsmooth Optimization via Quasi-Newton Methods

Bent Müller

2025-01-10

Outline

Basics of Optimization

The general unconstrained optimization problem is given by:

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice-differentiable real-valued function.

Basics of Optimization

The first-order Taylor approximation of f around x_k yields:

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k).$$

Basics of Optimization

The first-order Taylor approximation of f around x_k yields:

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k).$$

Which motivates the Gradient Descent method as follows:

$$x_{k+1} = x_k - t_k \nabla f(x_k),$$

where $t_k > 0$ is some step size.

Newton's Method

The second-order Taylor approximation of f around x_k yields:

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k).$$

A necessary condition for a local minimizer x_k is that $\nabla f(x_k) = \mathbf{0}$.

Newton's Method

The second-order Taylor approximation of f around x_k yields:

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k).$$

A necessary condition for a local minimizer x_k is that $\nabla f(x_k) = \mathbf{0}$.

'Jumping' to the minimizer of this quadratic approximation yields Newton's Method:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k),$$

where $\nabla^2 f(x_k)$ is the Hessian matrix.

Quasi-Newton Methods

The main disadvantage of Newton's method is the need to compute the Hessian matrix $\nabla^2 f(x_k) \in \mathbb{R}^{n \times n}$ and solve a linear system of equations in every iteration.

Also, the true Hessian might not always be positive definite.

Quasi-Newton Methods

Quasi-Newton methods approximate the **inverse** Hessian

$$H_k \approx (\nabla^2 f(x_k))^{-1}$$

by iteratively updating H_k.

Quasi-Newton Methods

Algorithm 1 (quasi-Newton method)

Choose x_0 with f differentiable at x_0, set H_0 to a positive definite matrix and $k \leftarrow 0$

`repeat`

compute search direction $p_k \leftarrow -H_k \nabla f_k$

set $x_{k+1} \leftarrow x_k + t_k p_k$, where $t_k > 0$

is chosen by a line search

`if` f is not differentiable at x_{k+1}, or $\nabla f_{k+1} = 0$,

stop.

set $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$

choose H_{k+1} to be a positive definite matrix satisfying

the secant condition $H_{k+1} y_k = t_k p_k$

$k \leftarrow k + 1$

`end (repeat)`

Quasi-Newton Methods

The true Hessian has the property

$$\nabla^2 f(x_k) \underbrace{(x_{k+1} - x_k)}_{tp_k} = \underbrace{\nabla f(x_{k+1}) - \nabla f(x_k)}_{y_k},$$

by the Taylor expansion of ∇f at x_k, which results in the secant equation since $H_{k+1} \approx (\nabla^2 f(x_k))^{-1}$.

Quasi-Newton Methods

The BFGS algorithm updates the inverse Hessian using:

$$H_{k+1} = V_k H_k V_k^T + t_k \frac{p_k p_k^T}{p_k^T p_k}$$

$$V_k = I - \frac{p_k y_k^T}{p_k^T y_k}.$$

Non-smooth Optimization

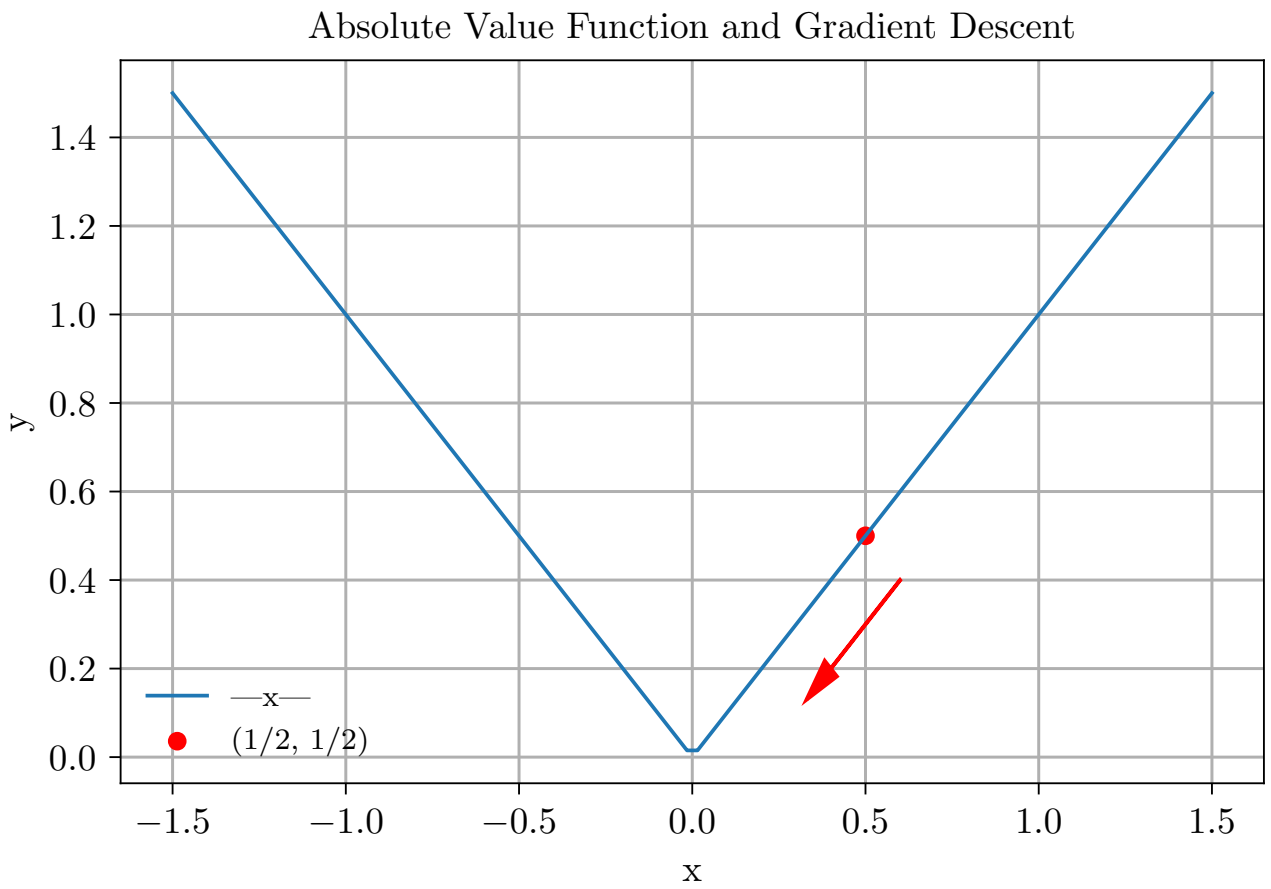

Figure: The absolute value function and Gradient Descent with $x_0 = \frac{1}{2}$ and $t_k = 1$.

Non-smooth Optimization

The problem is that the gradient does not smoothly change near the minimum, but jumps from -1 to 1. The gradient is not continuous at the minimum.

Non-smooth Optimization

In fact, with **any constant step size** t_k, Gradient Descent will oscillate forever on this problem.

Optimization Example on Euclidean Norm Function

Euclidean Norm: $f(x, y) = \sqrt{x^2 + 3y^2}$

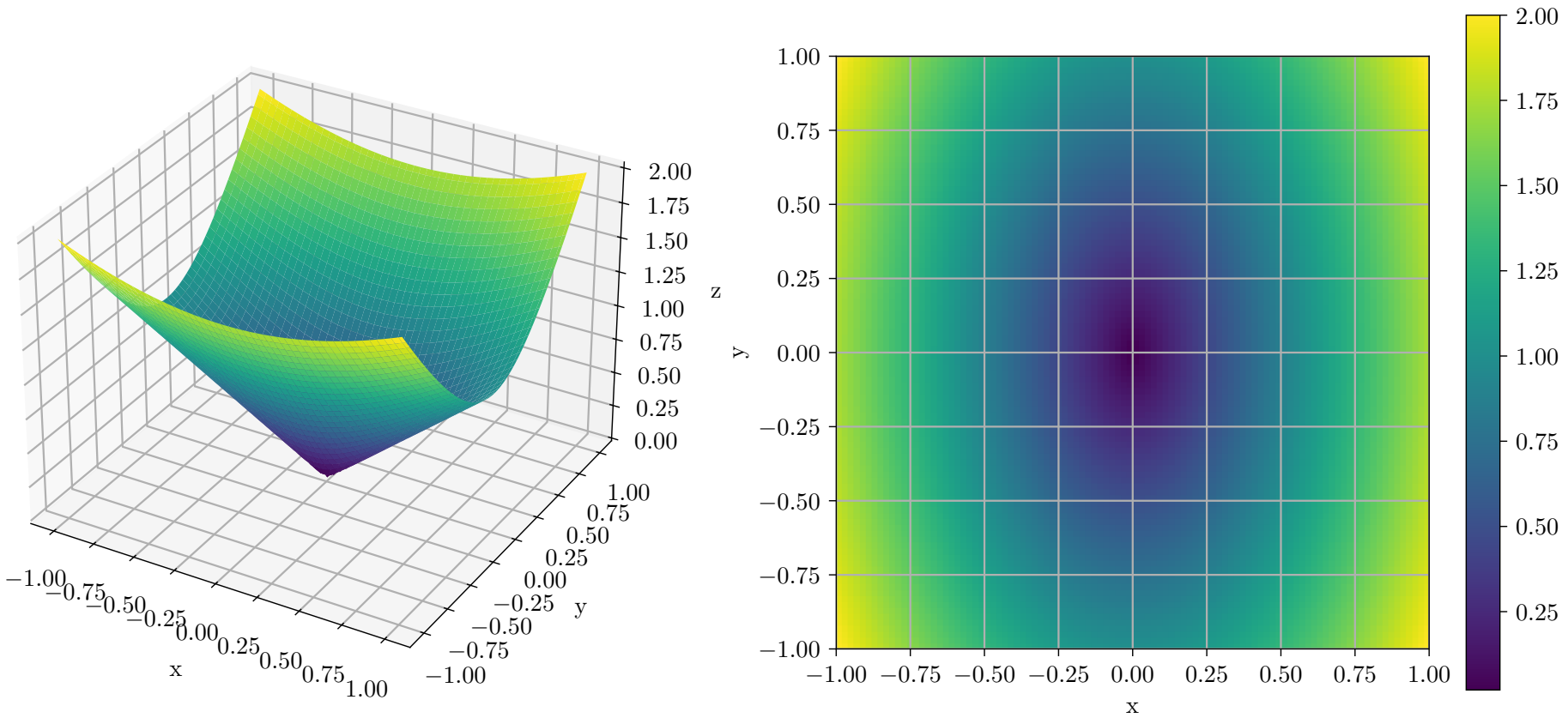

Figure: The (skewed) Euclidean norm function

Optimization Example on Euclidean Norm Function

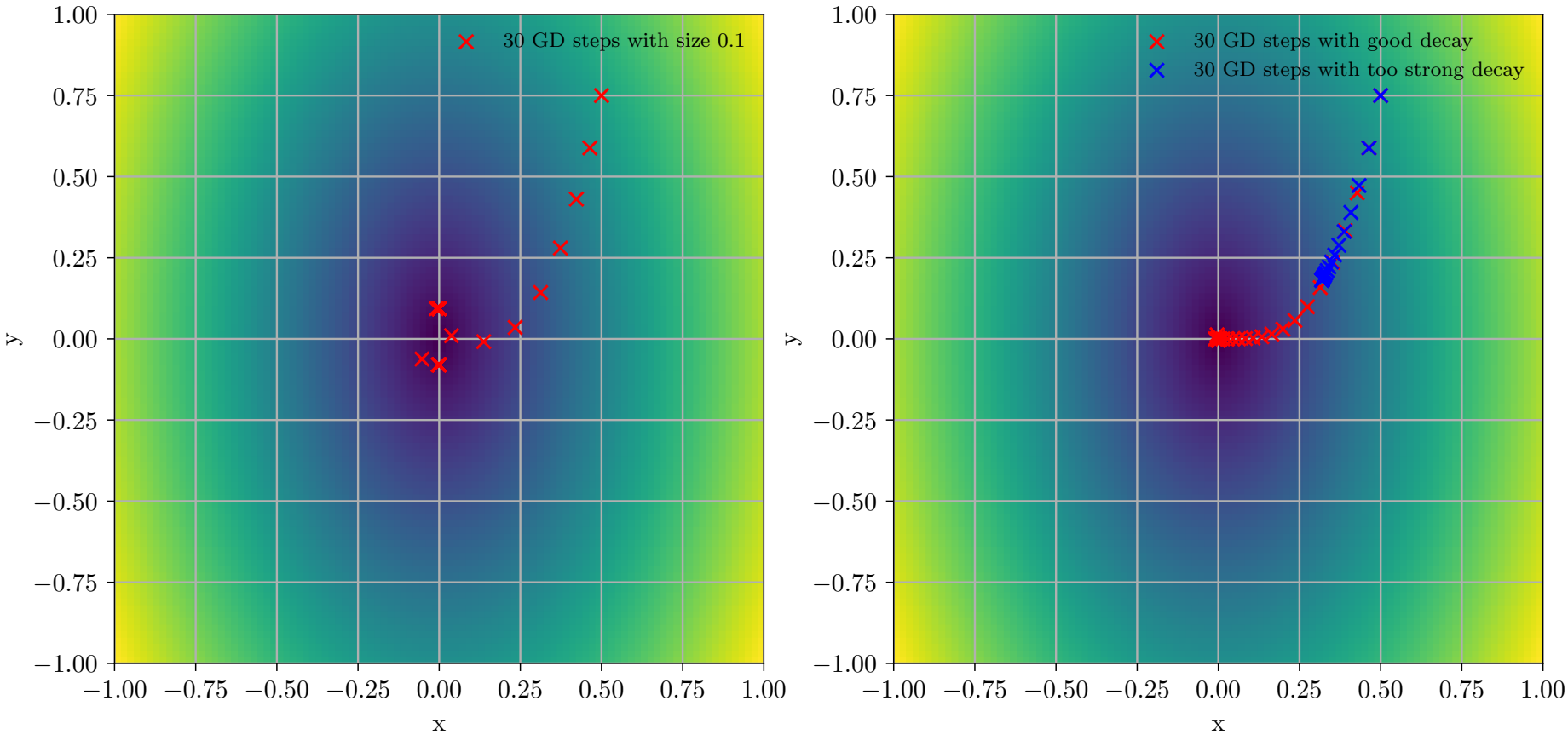

Figure: Gradient Descent with and without step size decay

Optimization Example on Euclidean Norm Function

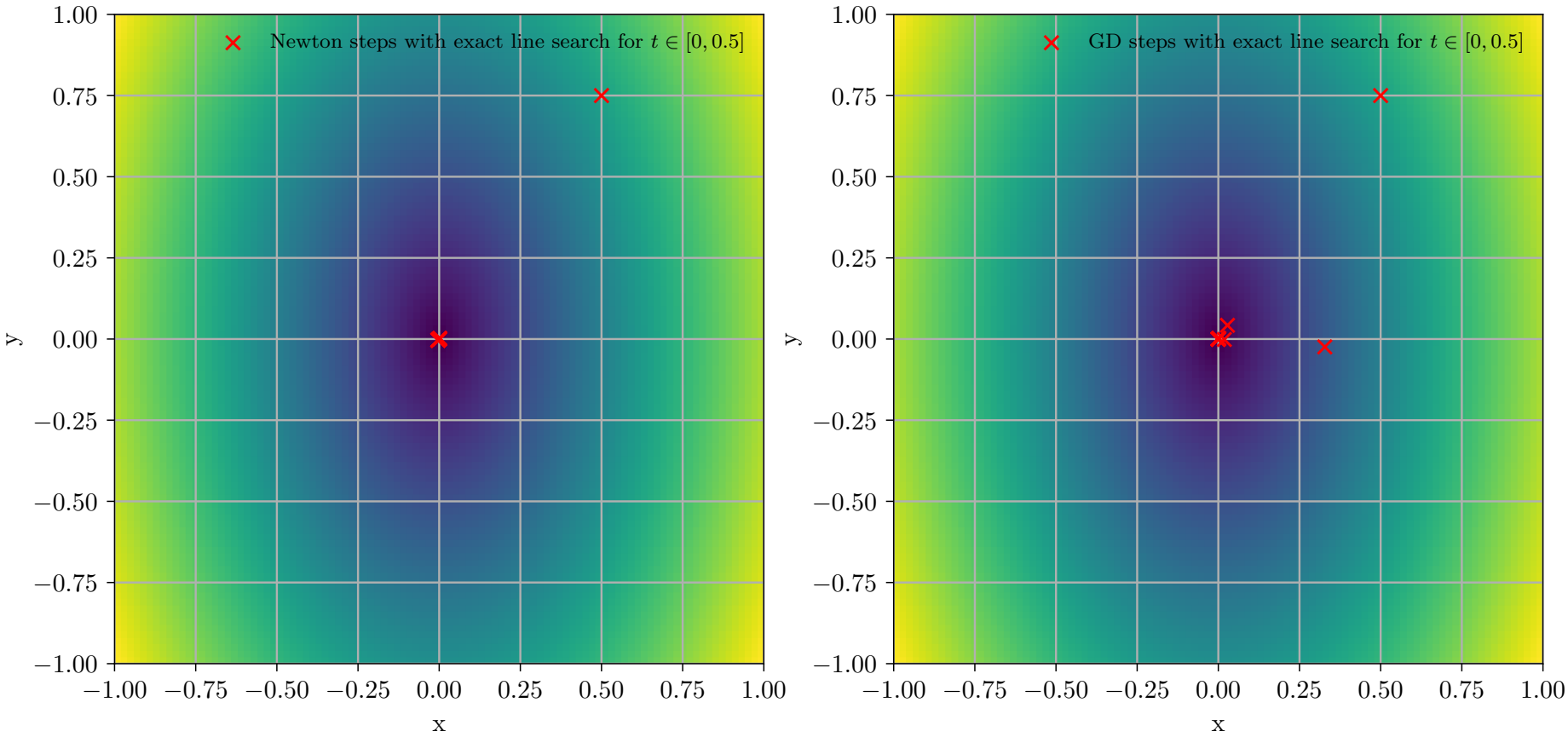

Figure: Newton and Gradient Descent with exact line search

Optimization Example on Euclidean Norm Function

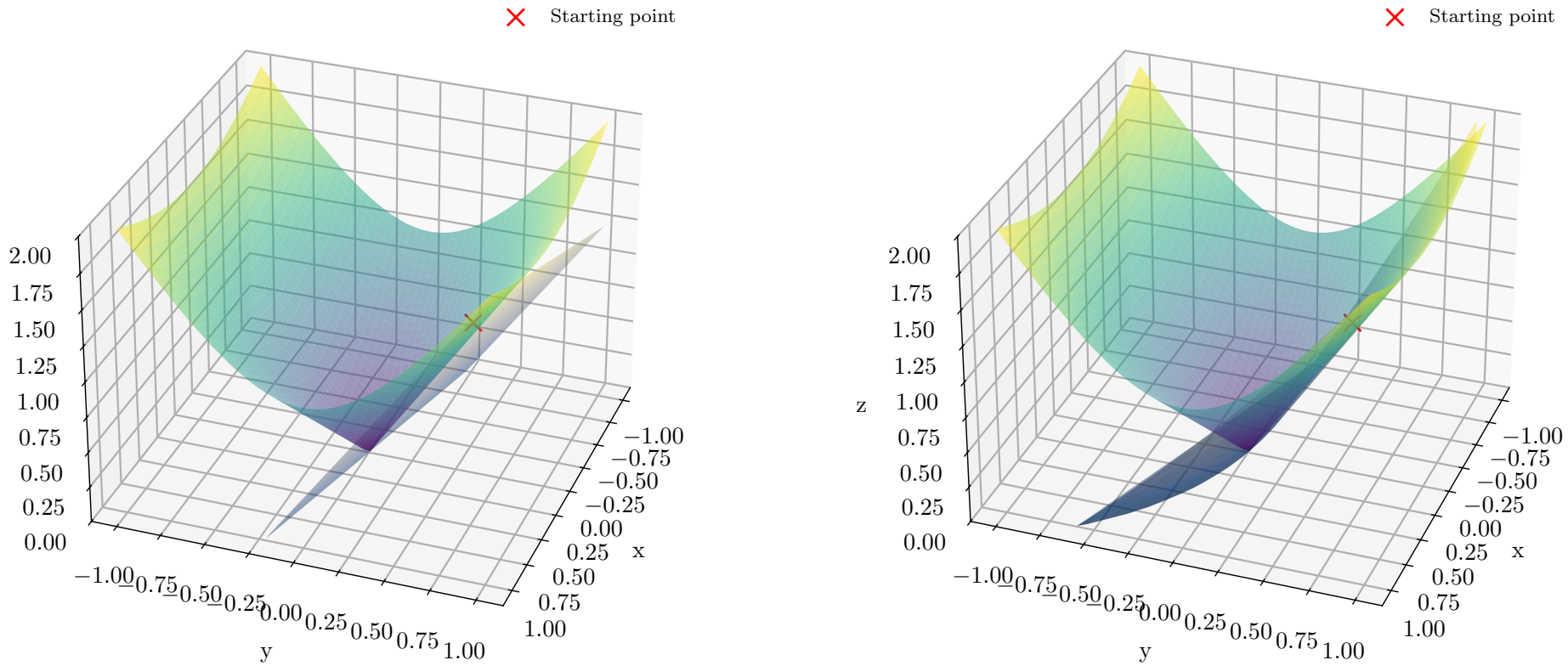

Figure: First and second order Taylor approximations of f

Optimization Example on Euclidean Norm Function

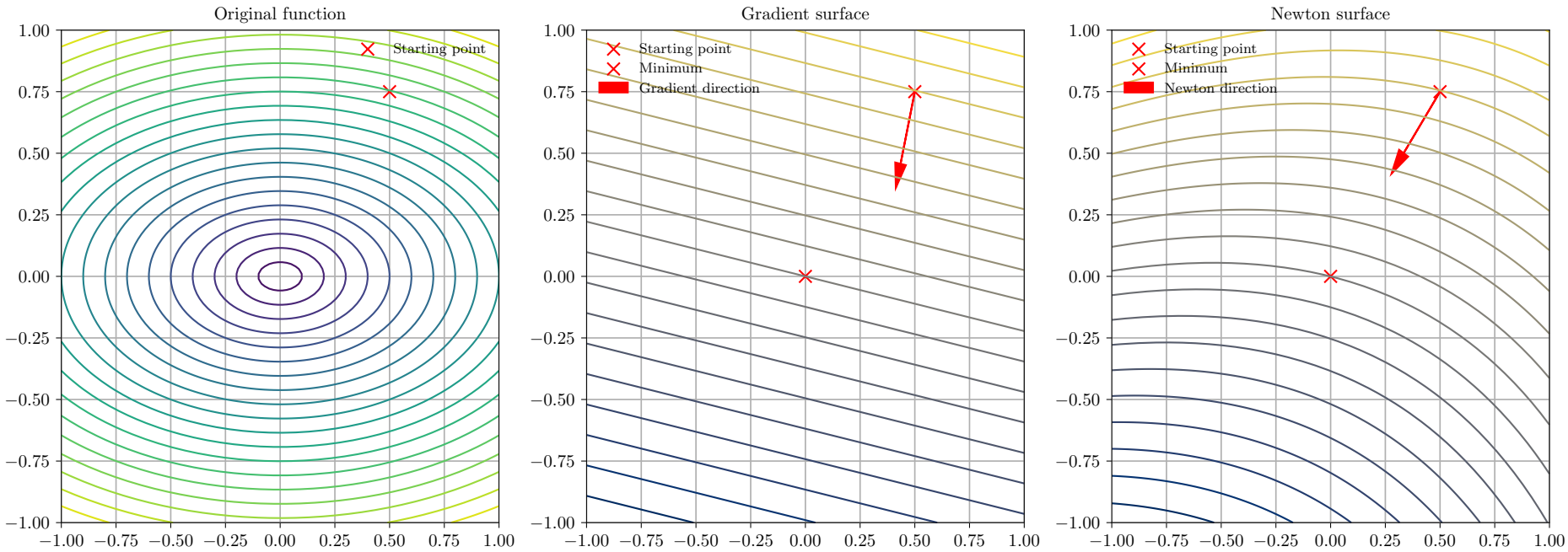

Figure: Contour plot of the first and second order Taylor approximations of f

Exact Line Search

Given a search direction p_k and a starting point x_k, **exact line search** solves:

$$t_k = \arg\min_{t>0} f(x_k + tp_k).$$

Exact Line Search

Given a search direction p_k and a starting point x_k, **exact line search** solves:

$$t_k = \arg\min_{t>0} f(x_k + tp_k).$$

In (quasi-)Newton methods, we search $t \in [0, 1]$ since we don't want to take steps larger than true newton would.

Exact Line Search

Given a search direction p_k and a starting point x_k, **exact line search** solves:

$$t_k = \arg\min_{t>0} f(x_k + tp_k).$$

In non-smooth problems, exact line search could find an x_{k+1} where f is not differentiable that **is not a minimizer**.

E.g. for $f(x, y) = |x| + |y|$ and $x_k = (1, 0)$,

Armijo-Wolfe Conditions

The **Armijo condition** ensures that the step size t_k decreases the objective function sufficiently:

$$f(x_k + t_k p_k) \leq f(x_k) + c_1 t_k \nabla f(x_k)^T p_k,$$

where $0 < c_1 < 1$ is a constant.

Armijo-Wolfe Conditions

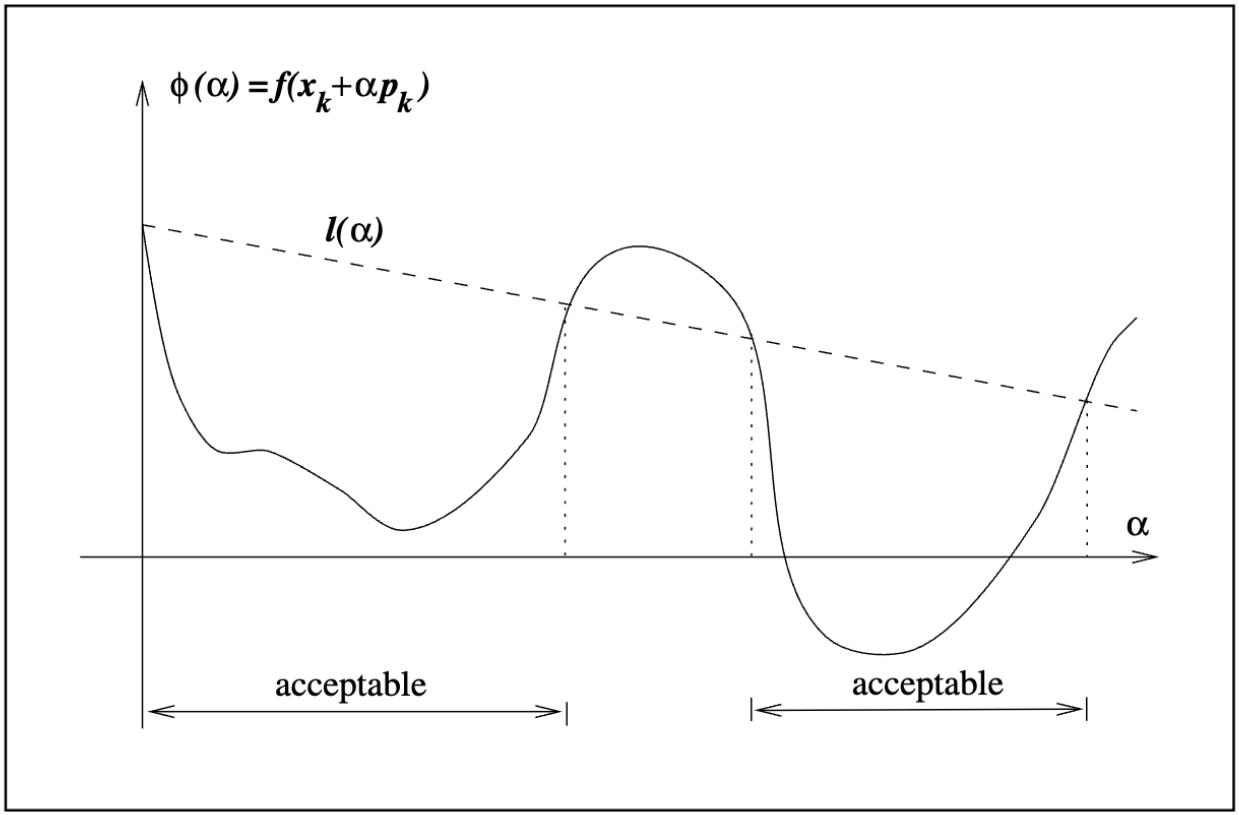

Figure: Armijo condition ensuring sufficient decrease; Source: Numerical Optimization by Nocedal and Wright, Figure 3.3

Armijo-Wolfe Conditions

However, the Armijo condition works for any sufficiently small step size t_k.

Armijo-Wolfe Conditions

The **Wolfe condition** ensures that the step size t_k is not too small:

$$\underbrace{\frac{\partial}{\partial t_k}(f(x_k + t_k p_k))}_{\text{Slope of the LS objective}} = \nabla f(x_k + t_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k,$$

where $c_1 < c_2 < 1$ is a constant.

Armijo-Wolfe Conditions

The intuition here is that: if line search would still improve the objective, then we should keep going

Armijo-Wolfe Conditions

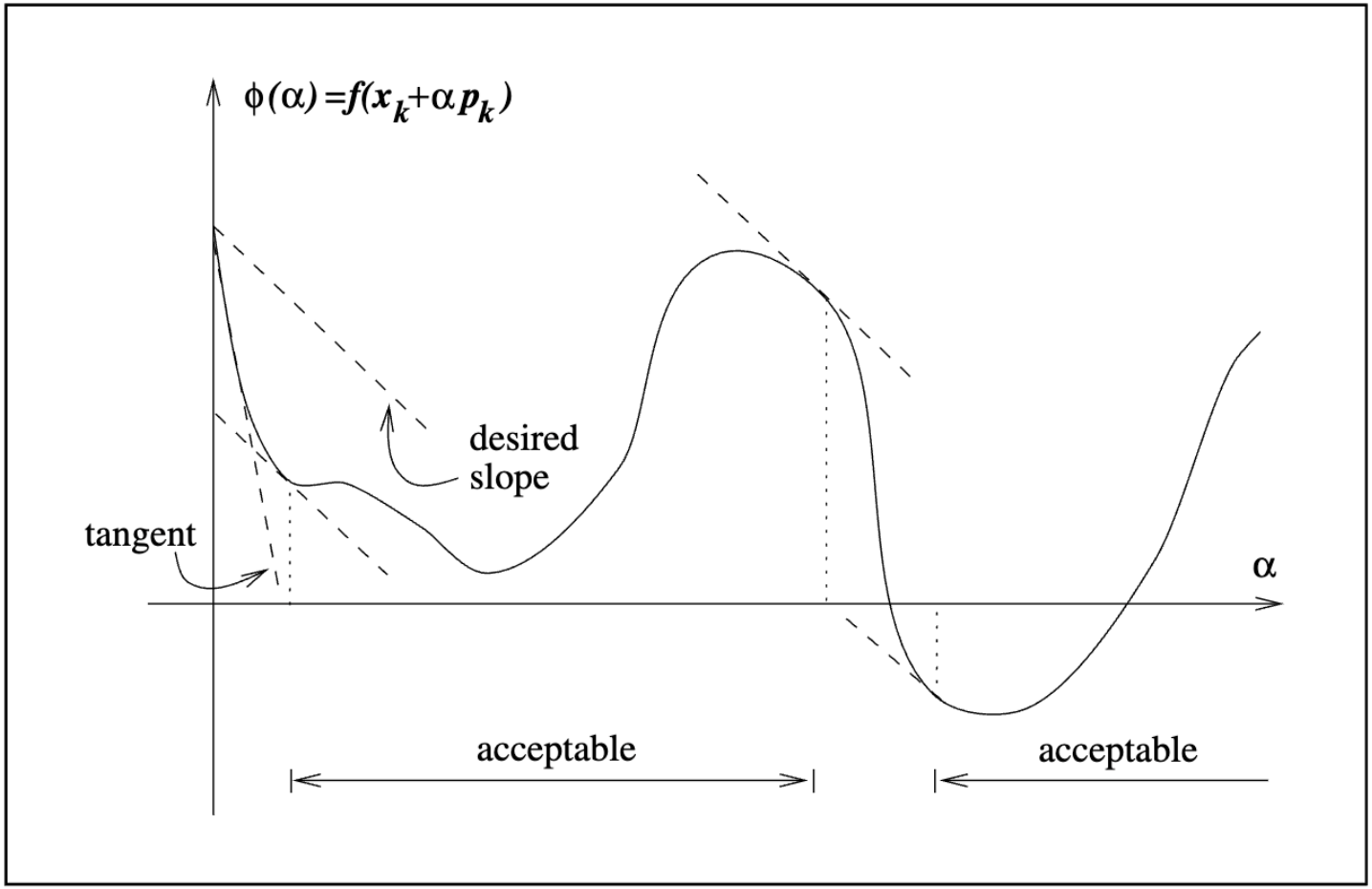

Figure: The Wolfe/Curvature condition; Source: Numerical Optimization by Nocedal and Wright, Figure 3.4

Inexact Line Search

Using these conditions, we can define an **inexact line search**:

Algorithm 2 (inexact line search)

$\alpha \leftarrow 0 \qquad \beta \leftarrow +\infty \qquad t \leftarrow 1$

repeat

  if $A(t)$ **fails** $\beta \leftarrow t$

  elseif $W(t)$ **fails** $\alpha \leftarrow t$

  else stop

  if $\beta < +\infty \quad t \leftarrow (\alpha + \beta)/2$

  else $t \leftarrow 2\alpha$

end(repeat)

Inexact Line Search

Using these conditions, we can define an **inexact line search**:

Algorithm 2 (inexact line search)

$\alpha \leftarrow 0 \qquad \beta \leftarrow +\infty \qquad t \leftarrow 1$

repeat

  if $A(t)$ **fails** $\beta \leftarrow t$

  elseif $W(t)$ **fails** $\alpha \leftarrow t$

  else stop

  if $\beta < +\infty \quad t \leftarrow (\alpha + \beta)/2$

  else $t \leftarrow 2\alpha$

end(repeat)

Here, the Wolfe condition $W(t)$ implicitly checks differentiability of f at $x_k + tp_k$.

Inexact Line Search

Inexact line search is great because we can apply quasi-Newton methods to non-smooth problems.

Conclusion

In Summary:

- (quasi-)Newton gives us the best search direction and hence faster convergence, but comes at additional n^2 cost for the Hessian.
- With inexact line search, this works well for non-smooth problems where exact line search can fail
- If function evaluation and gradient computation are cheap, Gradient Descent is likely the best choice, but line search is always crucial!

Conclusion

Thank you for your attention!