AI505

Optimization

Local Descent

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Line Search Methods

2. Convergence Analysis

3. Trust Region Methods

Preface

For multivariate functions, we have argued that:

- derivatives can have exponential growth in the resulting analytical expression
- calculating zeros might be challenging

Hence, minimizing by solving $\nabla f(\boldsymbol{x}) = 0$ may be computationally demanding.

Outline

Descent Direction Iteration

Descent Direction Methods use a local model to incrementally improve design point until some convergence criteria is met.

1. Check termination conditions at $\boldsymbol{x}_k$; if not met, continue.
2. Decide **descent direction** $\boldsymbol{d}_k$ using local information, commonly required that $\boldsymbol{d}_k^T \nabla f(\boldsymbol{x}_k) < 0$.
3. Decide **step size** (ie, magnitude of the overall step that depends on α_k, sometimes but not always $\|\boldsymbol{d}_k\|_2 = 1$)
4. Compute next design point $\boldsymbol{x}_{k+1}$

$$\boxed{\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k}$$

Descent Direction

The search direction often has the form

$$\boldsymbol{d}_k = -B_k^{-1}\nabla f(\boldsymbol{x}_k)$$

where B_k is a symmetric and nonsingular matrix.

- in the steepest descent method, B_k is the identity matrix I
- in Newton's method, B_k is the exact Hessian $\nabla^2 f(\boldsymbol{x}_k)$.
- in quasi-Newton methods, B_k is an approximation to the Hessian that is updated at every iteration by means of a low-rank formula.

When $\boldsymbol{d}_k = -B_k^{-1}\nabla f(\boldsymbol{x}_k)$ and B_k is positive definite, we have

$$\boldsymbol{d}_k^T\nabla f(\boldsymbol{x}_k) = -\nabla f(\boldsymbol{x}_k)^T B_k^{-1}\nabla f(\boldsymbol{x}_k) < 0$$

and therefore $\boldsymbol{d}_k$ is a descent direction. In fact, it is a double implication!

Outline

- We discuss how to choose α_k and $\boldsymbol{d}_k$ to promote convergence from remote starting points.
- We also consider the rate of convergence of steepest descent, quasi-Newton, and Newton methods.

Line Search for Step Size

Assuming we have the search direction:

- Use it to compute α
- Using the techniques discussed in the previous class, find the minimum of a univariate function:

$$\phi(\alpha) = f(\boldsymbol{x} + \alpha \boldsymbol{d}) \qquad \text{minimize}_{\alpha \geq 0}\, \phi(\alpha)$$

We assume that $\boldsymbol{p}_k$ is a descent direction, that is, $\phi'(0) < 0$, so that our search can be confined to positive values of α.

```
def line_search(f, x, d)
  objective = lambda alpha: f(x + alpha * d)
  a, b = bracket_minimum(objective)
  alpha = minimize(objective, a, b)
  return x + alpha * d
```

Often computationally costly, so approximate line search is used instead

Line Search: Alternatives

α is called to the **learning rate** or **step factor**:

Equal to the **step size** only when $\|\boldsymbol{d}_k\|_2 = 1$.

- Fixed learning rate

- **Decaying step factor**

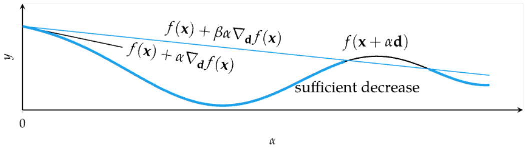
$$\alpha_k = \alpha_1 \gamma^{k-1} \qquad \text{for } \gamma \in [0, 1]$$

Decaying step factor is often required in convergence proofs

Approximate Line Search

- If function calls are expensive, rather than finding the minimum along a search direction, find a point of sufficient decrease

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + \beta\alpha\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k)$$

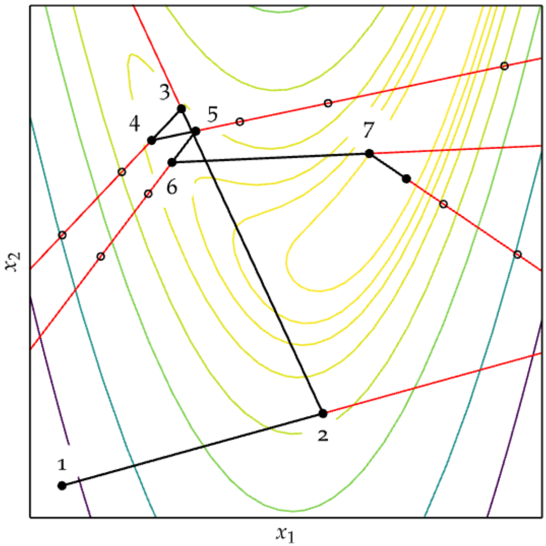


- $\beta \in [0, 1]$, usually $\beta = 1 \times 10^{-4}$
- Alone this condition is insufficient to guarantee convergence to a local minimum. It can converge prematurely.
- Backtracking line search starts with a large step and then backs off

```
def backtracking_line_search(f, grad, x, d, alpha_0=1, p=0.5, beta=1e-4):
    y, g, alpha = f(x), grad(x), alpha_0
    while ( f(x + alpha * d) > y + beta * alpha * np.dot(g, d) ) :
        alpha *= p
    return alpha
```

- Guaranteed to converge to a local minimum, but can be slow.

Approximate Line Search: Example



Approximate Line Search

Building on backtracking line search are the **Wolfe Conditions** together sufficient to guarantee convergence to a local minimum.

1. First Wolfe Condition: Sufficient Decrease (Armijo condition)

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + \beta\alpha\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k)$$

2. Second Wolfe Condition: Curvature Condition

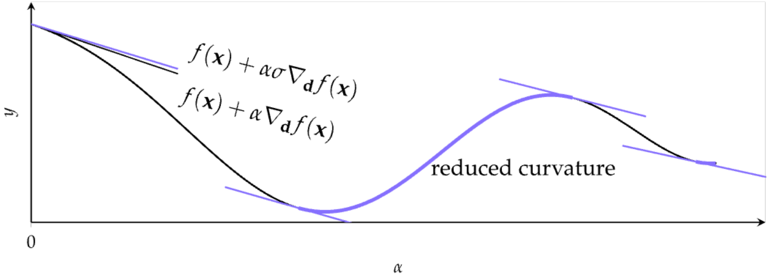
$$\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_{k+1}) \geq \sigma\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k)$$

$\beta < \sigma < 1$ with
- $\sigma = 0.1$ with conjugate gradient method
- $\sigma = 0.9$ with Newton method

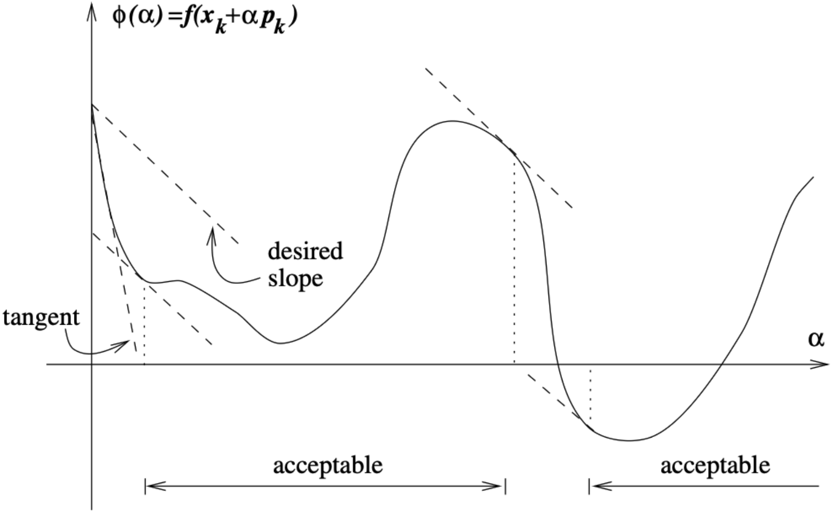
Curvature Condition

Regions satisfying the curvature condition $\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_{k+1}) \geq \sigma \nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k)$

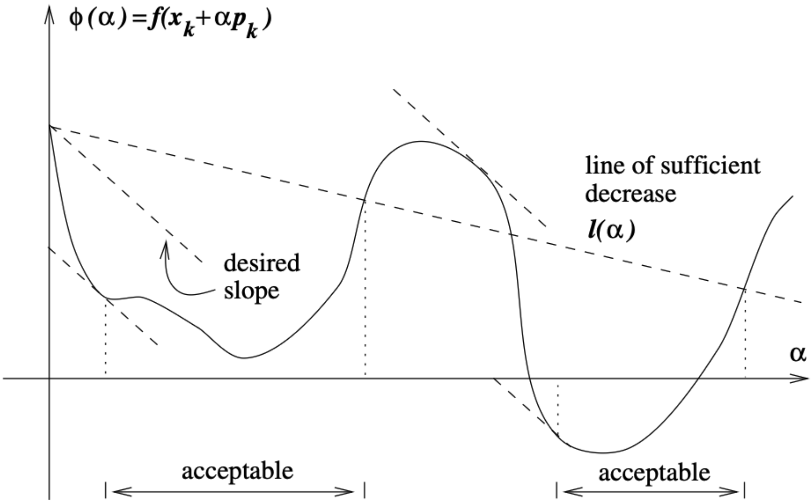
- Consider the univariate function $\phi(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$.
- The left-hand-side is simply the derivative $\phi'(\alpha_k)$, so the curvature condition ensures that the slope of ϕ at α_k is greater than σ times the initial slope $\phi'(0)$.
- If the slope $\phi'(0)$ is negative, then we are looking for a step size α_k such that the slope of ϕ at that point is still negative but not too negative.
- If $\phi'(\alpha_k)$ is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction, so it makes sense to terminate the line search.



Curvature Condition



Wolfe Conditions



Approximate Line Search: Example

Consider approximate line search on $f(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2$
from $\boldsymbol{x} = [1, 2]$ in the direction $\boldsymbol{d} = [-1, -1]$, gradient at $\boldsymbol{x}$ is $\boldsymbol{g} = [4, 5]$
using a maximum step size of 10, a reduction factor of 0.5,
first Wolfe condition parameter $\beta = 1 \times 10^{-4}$, second Wolfe condition parameter $\sigma = 0.9$.

First Wolfe condition ($f(\boldsymbol{x} + \alpha\boldsymbol{d}) \leq f(\boldsymbol{x}) + \beta\alpha(\boldsymbol{g}^T \cdot \boldsymbol{d})$):

$$\begin{aligned}
\alpha = 10 : \quad & f([1,2] + 10 \cdot [-1,-1]) \leq 7 + 1 \times 10^{-4} 10 [4,5]^T [-1,-1] \implies 217 \not\leq 6.991 \\
\alpha = 10 \cdot 0.5 = 5 : \quad & f([1,2] + 5 \cdot [-1,-1]) \leq 7 + 1 \times 10^{-4} 5 [4,5]^T [-1,-1] \implies 37 \not\leq 6.996 \\
\alpha = 2.5 : \quad & f([1,2] + 2.5 \cdot [-1,-1]) \leq 7 + 1 \times 10^{-4} 2.5 [4,5]^T [-1,-1] \implies 3.25 \leq 6.998
\end{aligned}$$

The candidate design point $\boldsymbol{x}' = \boldsymbol{x} + \alpha\boldsymbol{d} = [-1.5, -.0.5]$ is checked against the second Wolfe condition $\nabla_{\boldsymbol{d}} f(\boldsymbol{x}') \geq \sigma \nabla_{\boldsymbol{d}} f(\boldsymbol{x})$:

$$[-3.5, -2.5] \cdot [-1, -1] \geq \sigma [4, 5] \cdot [-1, -1] \implies 6 \geq -8.1$$

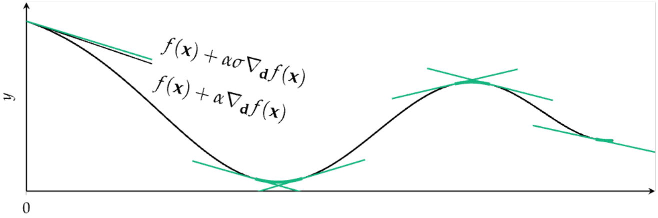
Approximate line search terminates with $\boldsymbol{x} = [-1.5, -0.5]$.

Strong Wolfe Conditions

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$.
We can modify the curvature condition to force $f(\boldsymbol{x}_{k+1})$ to exclude points that are far from stationary points of $f(\boldsymbol{x}_{k+1})$, ie, we no longer allow the gradient $\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_{k+1})$ to be too positive.

Strong Wolf conditions:

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + \beta\alpha\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k) \qquad \text{and} \qquad |\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_{k+1})| \leq \sigma|\nabla_{\boldsymbol{d}_k} f(\boldsymbol{x}_k)|$$



Approximate Line Search Goal

Given:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

with descent direction:

$$\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k < 0$$

Define:

$$\phi(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k), \qquad \phi'(\alpha) = \nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)^T \boldsymbol{d}_k$$

Goal:

Find $\alpha > 0$ satisfying the **Strong Wolfe Conditions**.

Strong Wolfe Conditions

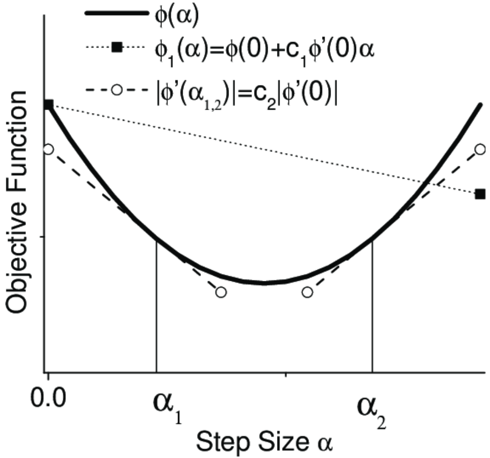
Two requirements:

Sufficient decrease (Armijo)

$$\phi(\alpha) \le \phi(0) + \beta\alpha\phi'(0)$$

Strong curvature condition

$$|\phi'(\alpha)| \le \sigma|\phi'(0)|$$



Ensures:

- sufficient decrease
- step near minimum

Strong Backtracking: Idea

Two phases:

1. Bracketing phase

- Start with $[\alpha_0 = 0, \alpha_1]$
- Increase step
- Find interval containing solution

2. Zoom phase

- Shrink interval
- Use bisection or interpolation
- Find Wolfe step

Idea:

Bracket minimum $\rightarrow$ Zoom to solution

Algorithm

Bracketing

- Start with $[\alpha_0 = 0, \alpha_1]$
- If Armijo fails or slope changes sign

  $\rightarrow$ Zoom

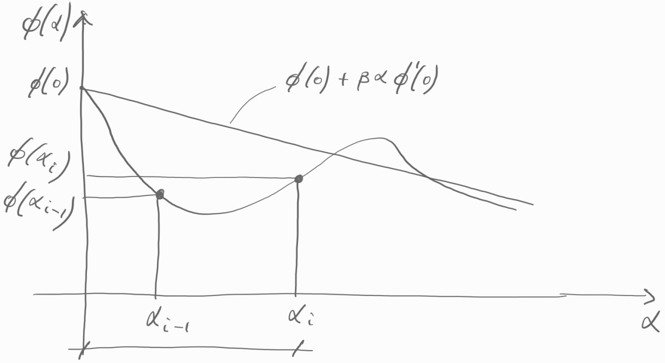
- If strong Wolfe satisfied

  $\rightarrow$ Stop

- Else increase step

Input : α_{max} maximum step allowed
Output: α^* set to a step length that satisfies the strong Wolfe condition

Initialize $k \leftarrow 0$;
Set $\alpha_0 \leftarrow 0$, choose $\alpha_{max} > 0$ and $\alpha_1 \in (0, \alpha_{max})$;
$i \leftarrow 1$;
while True **do**
  Evaluate $\phi(\alpha_i)$;
  if $\phi(\alpha_i) > \phi(0) + \beta\alpha_i\phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1]$ **then** // Armijo or next slide
    $\alpha^* \leftarrow \text{zoom}(\alpha_{i-1}, \alpha_i)$ and break;
  Evaluate $\phi'(\alpha_i)$;
  if $|\phi'(\alpha_i)| \leq -\sigma\phi'(0)$ **then**
    set $\alpha^* \leftarrow \alpha_i$ and break;
  if $\phi'(\alpha_i) \geq 0$ **then** // slope changes
    set $\alpha^* \leftarrow \text{zoom}(\alpha_i, \alpha_{i-1})$ and break;
  Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{max})$; // Eg, $\alpha_{i+1} \leftarrow 2\alpha_i$
  $i \leftarrow i + 1$;


$\phi(\alpha)$
$\phi(0)$
$\phi(0) + \beta\alpha\phi'(0)$
$\phi(\alpha_i)$
$\phi(\alpha_{i-1})$
α_{i-1}
α_i
α

Algorithm

Zoom

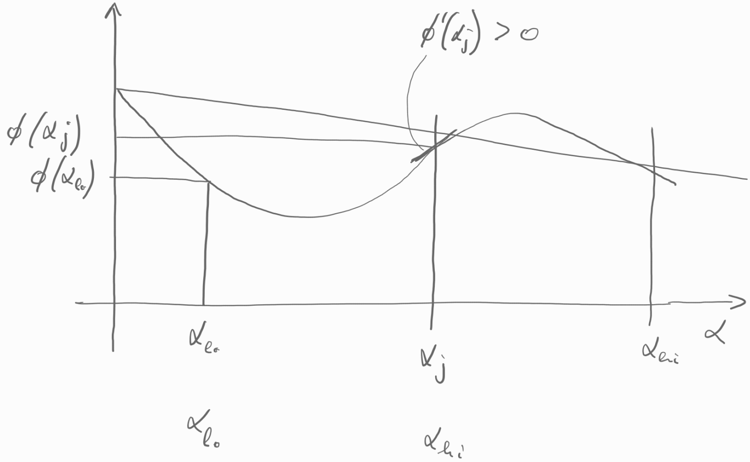
- Interpolate inside bracket
- Shrink interval
- Stop when Wolfe satisfied

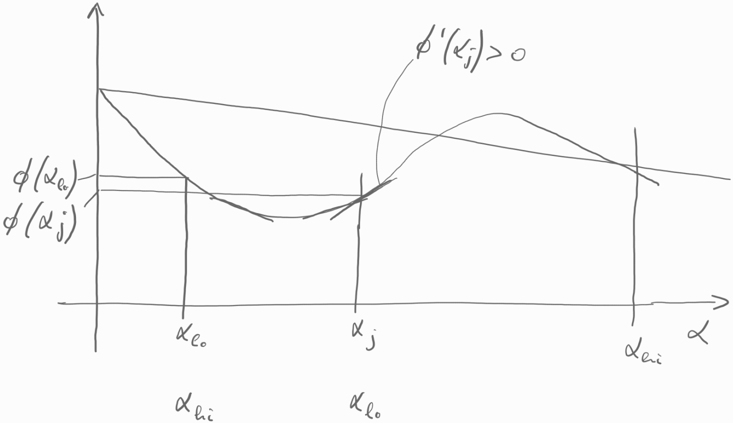
Iterate generating an $\alpha_j \in [\alpha_{lo}, \alpha_{hi}]$, and then replacing one of these endpoints by α_j maintaining the invariants:

1. $[\alpha_{lo}, \alpha_{hi}]$ satisfy the strong Wolfe conditions;
2. α_{lo} smallest function value $\phi(\alpha_{lo})$;
3. α_{hi} chosen so that $\phi'(\alpha_{lo})(\alpha_{hi} - \alpha_{lo}) < 0$.

Input : α_{lo}, α_{hi}
Output: α^* set to a step length that satisfies the strong Wolfe condition

while True **do**
  Interpolate (using quadratic, cubic or bisection) to find a trial step length α_j between α_{lo} and α_{hi} ;
  Evaluate $\phi(\alpha_j)$;
  if $\phi(\alpha_j) > \phi(0) + \beta\alpha_j\phi'(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{lo})$ **then** // next slide
    $\alpha_{hi} \leftarrow \alpha_j$;
  else
    Evaluate $\phi'(\alpha_j)$;
    if $|\phi'(\alpha_j)| \leq -\sigma\phi'(0)$ **then**
      Set $\alpha^* \leftarrow \alpha_j$ and break;
    if $\phi'(\alpha_j)(\alpha_{hi} - \alpha_{lo}) \geq 0$ **then** // violates 3., second next slide
      $\alpha_{hi} \leftarrow \alpha_{lo}$;
    $\alpha_{lo} \leftarrow \alpha_j$;


φ'(α_j) > 0
φ(α_j)
φ(α_lo)
α_lo
α_j
α_hi
α
α_lo
α_hi


$\phi'(\alpha_j) > 0$
$\phi(\alpha_{lo})$
$\phi(\alpha_j)$
α_{lo}
α_j
α_{hi}
α
α_{hi}
α_{lo}

Approximate Line Search: Python Implementation

```
def strong_backtracking(f, nabla, x, d; alpha=1, beta=1e-4, sigma=0.1)
  y0, g0, y_prev, alpha_prev = f(x), nabla(x) @ d, NaN, 0
  alpha_lo, alpha_hi = NaN, NaN
```

```
# bracket phase
while true
  y = f(x + alpha*d)
  if y > y0 + beta*alpha*g0 || (!isnan(y_prev)↪
      ↪ && y >= y_prev)
    alpha_lo, alpha_hi = alpha_prev, alpha
    break
  g = nabla(x + alpha*d) @ d
  if abs(g) <= -sigma * g0
    return alpha
  elif g <= 0
    alpha_lo, alpha_hi = alpha, alpha_prev
    break
  y_prev, alpha_prev, alpha = y, alpha, 2 * ↪
      ↪alpha
```

```
# zoom phase
ylo = f(x + alpha_lo*d)
while true:
  alpha = (alpha_lo + alpha_hi)/2
  y = f(x + alpha*d)
  if y > y0 + beta*alpha*g0 || y >= ↪
       ↪ylo
    alpha_hi = alpha
  else
    g = nabla(x + alpha*d) @ d
    if abs(g) <= -sigma*g0
      return alpha
    elif g*(alpha_hi -alpha_lo) >= 0
      alpha_hi = alpha_lo
    alpha_lo = alpha
```

Why Strong Backtracking?

Used in:

- Gradient descent
- Quasi-Newton: BFGS, L-BFGS
- Nonlinear Conjugate Gradient

Guarantees:

- global convergence
- stable steps
- fast convergence of quasi-Newton

Standard method in modern optimization.

Note: it is an algorithm that make use of derivative information!

Outline

1. Line Search Methods

2. Convergence Analysis

3. Trust Region Methods

Convergence Analysis

Let $\{\boldsymbol{x}_k\}$ be a sequence of points belonging in $\mathbb{R}^n$.
We say that a sequence $\{\boldsymbol{x}_k\}$ converges to some point $\boldsymbol{x}$, written $\lim_{k\text{ß}\infty} \boldsymbol{x}_k = x$, if for any $\epsilon > 0$, there is an index K such that

$$\|\boldsymbol{x}_k - \boldsymbol{x}\| \leq \epsilon \qquad \text{for all } k \geq K$$

For example, the sequence $\boldsymbol{x}_k$ defined by $\boldsymbol{x}_k = (1 - 2^{-k}, 1/k^2)^T$ converges to $(1, 0)^T$.

Convergence of Line Search

Let θ_k be the angle between $\boldsymbol{d}_k$ and the steepest descent direction $-\nabla f_k$, defined by:

$$\cos\theta_k = \frac{-\nabla f_k^T \boldsymbol{d}_k}{\|\nabla f_k\| \, \|\boldsymbol{d}_k\|}$$

Theorem (Zoutendijk condition)

Consider any iteration of the form $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where $\boldsymbol{d}_k$ is a descent direction and α_k satisfies the Wolfe conditions. Suppose that f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set N containing the level set $\mathcal{L} = \{x : f(x) \le f(x_0)\}$, where $\boldsymbol{x}_0$ is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on N, that is, there exists a constant $L > 0$ such that:

$$|f(x) - f(y)| \le L|x - y|, \quad \forall x, y \in N$$

Then:

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

The Zoutendijk condition implies that

$$\cos^2\theta_k \, \|\nabla f_k\|^2 \to 0$$

We can be sure that the gradient norms $\|\nabla f_k\|$ converge to zero, provided that the search directions are never too close to orthogonality with the gradient.

It is a **global convergence** result.
Here the strongest possible result: we cannot guarantee that the method converges to a minimizer, but only that it is attracted by stationary points (only introducing the Hessian we can prove convergence to a local minimum — see next slide)

In

$$\boldsymbol{d}_k = -B_k^{-1}\nabla f(\boldsymbol{x}_k)$$

assume that the matrices B_k are **positive definite** with a uniformly bounded condition number. That is, there is a constant M such that $\|B_k\| \left\|B_k^{-1}\right\| \leq M$ for all k.

Then from the definition of θ_k:

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

By combining this bound with $\cos^2\theta_k \|\nabla f_k\|^2 \to 0$ we find that

$$\lim_{k\to\infty} \|\nabla f_k\| = 0$$

So, globally convergent under the positive definitness assumptions on B_k, (which is needed to ensure that $\boldsymbol{d}_k$ is a descent direction), and if the step lengths satisfy the Wolfe conditions.

Rate of Convergence

Let $\{\boldsymbol{x}_k\}$ be a sequence in $\mathbb{R}^n$ that converges to $\boldsymbol{x}^*$.
The convergence is said to be **Q-linear** (quotient-linear) if there is a constant $r \in (0, 1)$ such that

$$\boxed{\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} \leq r \qquad \text{for all } k \text{ sufficiently large}}$$

ie, the distance to the solution $\boldsymbol{x}^*$ decreases at each iteration by at least a constant factor bounded away from 1 (ie, < 1).

Example:
sequence $\{1 + (0.5)^k\}$ converges Q-linearly to 1, with rate $r = 0.5$.

Rate of Convergence

The convergence is said to be **Q-superlinear** if

$$\boxed{\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 0}$$

Example: the sequence $\{1 + k^{-k}\}$ converges superlinearly to 1.

An even more rapid convergence rate: The convergence is said to be **Q-quadratic** if

$$\boxed{\frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^2} \leq M \qquad \text{for all } k \text{ sufficiently large}}$$

where M is a positive constant, not necessarily less than 1. Example: the sequence $\{1 + (0.5)^{2^k}\}$. The values of r and M depend not only on the algorithm but also on the properties of the particular problem. Regardless of these values a quadratically convergent sequence will always eventually converge faster than a linearly convergent sequence.

Rate of Convergence

Superlinear convergence (quadratic, cubic, quartic, etc) is regarded as fast and desirable, while sublinear convergence is usually impractical.

- Steepest descent algorithms converge only at a Q-linear rate, and when the problem is ill-conditioned the convergence constant r in is close to 1.
- Quasi-Newton methods for unconstrained optimization typically converge Q-superlinearly
- Newton's method converges Q-quadratically under appropriate assumptions.

Rate of Convergence

A slightly weaker form of convergence:
overall rate of decrease in the error, rather than the decrease over each individual step of the algorithm.
We say that convergence is **R-linear** (root-linear) if there is a sequence of nonnegative scalars $\{v_k\}$ such that

$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq \{v_k\}$ for all k, and $\{v_k\}$ converges Q-linearly to zero.

Outline

1. Line Search Methods

2. Convergence Analysis

3. Trust Region Methods

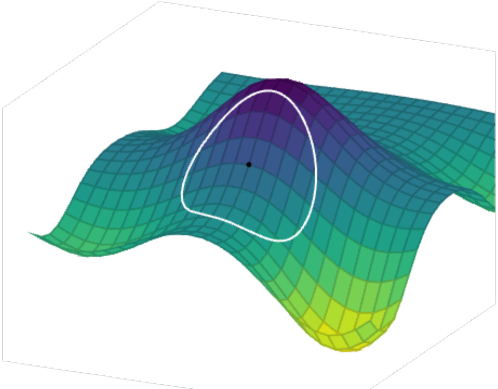
Trust Region Methods

- Descent methods can place too much trust in their first and second order information
- A **trust region** is the local area of the design space where the local model is believed to be reliable.
- Trust region methods, or restricted step methods, limit the step size to ensure local approximation error is minimized
- If the improvement matches the predicted value, the trust region is expanded; otherwise it is contracted

Trust Region Methods

- $\boldsymbol{x}'$ is new design point
- $\hat{f}(\boldsymbol{x}')$ is local function approximation, eg, second-order Taylor approximation
- δ is trust region radius

$$\begin{aligned} \text{minimize}_{\boldsymbol{x}'} \ & \hat{f}(\boldsymbol{x}') \\ \text{subject to} \ & ||\boldsymbol{x} - \boldsymbol{x}'|| \leq \delta \end{aligned}$$



Constrained optimization problem.
It can be solved efficiently if $\hat{f}$ quadratic

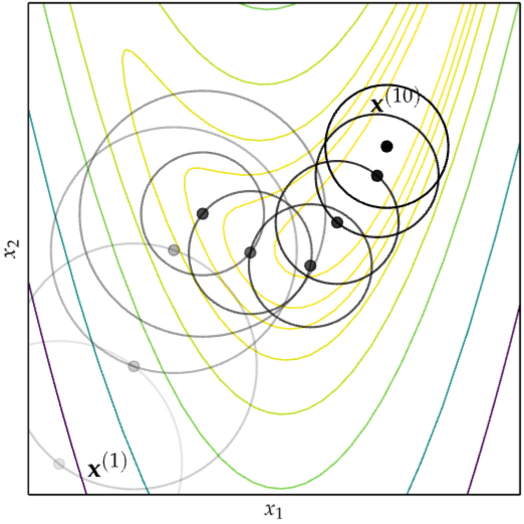
Trust Region Methods

δ can be expanded or contracted based on performance

$$\eta = \frac{\text{actual improvement}}{\text{predicted improvement}} = \frac{f(\boldsymbol{x}) - f(\boldsymbol{x}')}{f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}')}$$

If $\eta < \eta_1$ contract by a factor $\gamma_k < 1$
if $\eta > \eta_2$ expand by a factor $\gamma_k > 1$

Trust Region Methods: Example



Trust regions can be also non circular.

Trust Region Methods

Termination Conditions (commonly used together):

- Maximum Iterations: $k > k_{\text{max}}$
- Aboslute Improvement: $f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) < \epsilon_a$
- Relative Improvement: $f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) < \epsilon_r |f(\boldsymbol{x}_k)|$
- Gradient Magnitude: $||\nabla f(\boldsymbol{x}_{k+1})|| < \epsilon_g$

Then random restart.

Summary

- Descent direction methods incrementally descend toward a local optimum.
- Univariate optimization can be applied during line search.
- Approximate line search can be used to identify appropriate descent step sizes.
- Trust region methods constrain the step to lie within a local region that expands or contracts based on predictive accuracy.
- Termination conditions for descent methods can be based on criteria such as the change in the objective function value or magnitude of the gradient.