

Lecture 9 - Algorithms for unconstrained optimization - Newton Methods

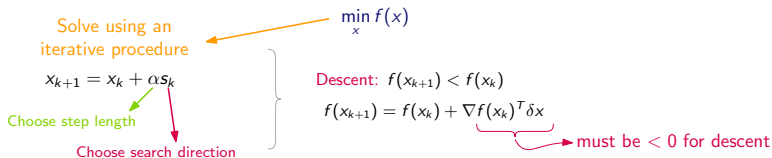
Dinesh Krishnamoorthy

Department of Engineering Cybernetics, NTNU

Spring 2026

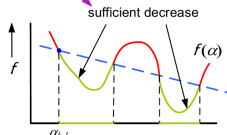
Recap

Unconstrained Optimization



Gradient descent algorithm

1. Choose $s_k = -\nabla f(x_k)$
 2. Choose step length α
 3. Update $x_{k+1} = x_k + \alpha s_k$
- Exact line search
- $$\alpha = \arg \min_{\alpha \geq 0} f(x_k + \alpha s_k)$$
- Backtracking line search



Example 3

$$f = x_1 + x_2 + x_1 e^{-x_2} + x_2^2 e^{-x_1}$$

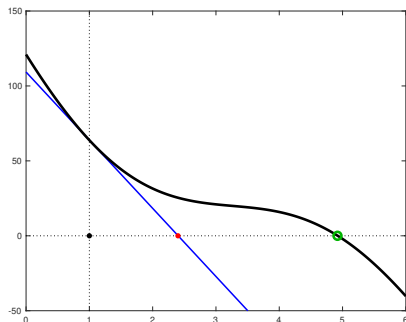
Stationary conditions

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= 1 + e^{-x_2} - x_2^2 e^{-x_1} = 0 \\ \frac{\partial f}{\partial x_2} &= 1 - x_1 e^{-x_2} + 2x_2 e^{-x_1} = 0\end{aligned}$$

This is a system of nonlinear equations!

Solving a system on nonlinear equations (Root finding)

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- Compute $F(w_k)$ and $\nabla F(w_k)$
- Compute Newton direction

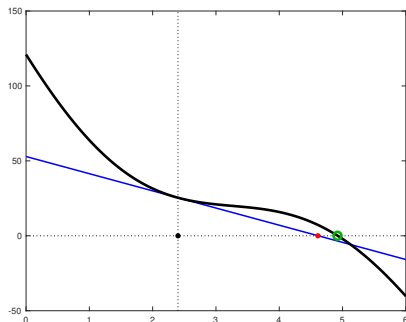
$$\nabla F(w_k)^T \Delta w = -F(w_k)$$

- Full Newton step

$$w_{k+1} \leftarrow w_k + \Delta w$$

Solving a system on nonlinear equations (Root finding)

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w_k)$ and $\nabla F(w_k)$
- ▶ Compute Newton direction

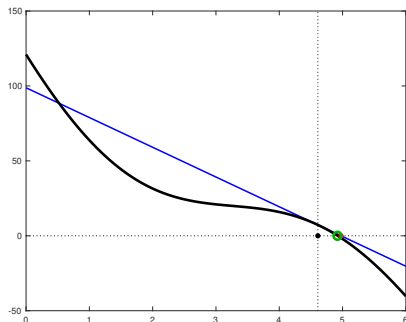
$$\nabla F(w_k)^T \Delta w = -F(w_k)$$

- ▶ Full Newton step

$$w_{k+1} \leftarrow w_k + \Delta w$$

Solving a system on nonlinear equations (Root finding)

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w_k)$ and $\nabla F(w_k)$
- ▶ Compute Newton direction

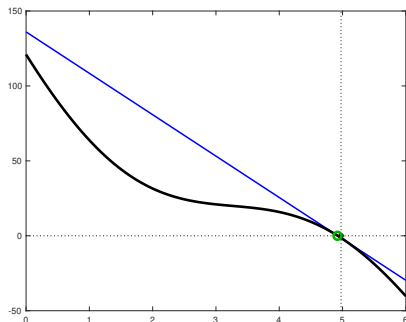
$$\nabla F(w_k)^T \Delta w = -F(w_k)$$

- ▶ Full Newton step

$$w_{k+1} \leftarrow w_k + \Delta w$$

Solving a system on nonlinear equations (Root finding)

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w_k)$ and $\nabla F(w_k)$
- ▶ Compute Newton direction

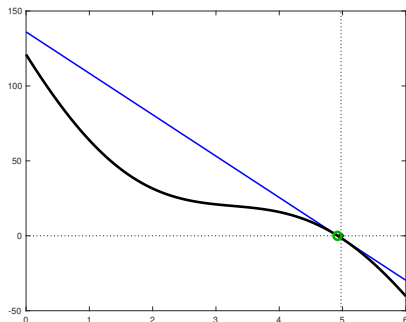
$$\nabla F(w_k)^T \Delta w = -F(w_k)$$

- ▶ Full Newton step

$$w_{k+1} \leftarrow w_k + \Delta w$$

Solving a system on nonlinear equations (Root finding)

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w_k)$ and $\nabla F(w_k)$
- ▶ Compute Newton direction

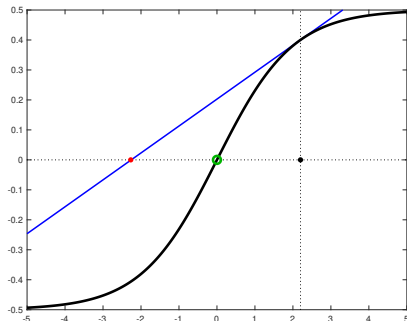
$$\nabla F(w_k)^T \Delta w = -F(w_k)$$

- ▶ Full Newton step

$$w_{k+1} \leftarrow w_k + \Delta w$$

Newton method can be unstable!

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w)$ and $\nabla F(w)$
- ▶ Compute Newton direction

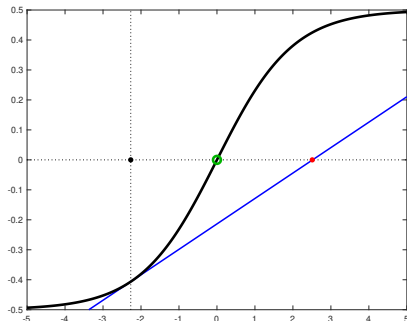
$$\nabla F(w)^T \Delta w = -F(w)$$

- ▶ Full Newton step

$$w \leftarrow w + \Delta w$$

Newton method can be unstable!

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w)$ and $\nabla F(w)$
- ▶ Compute Newton direction

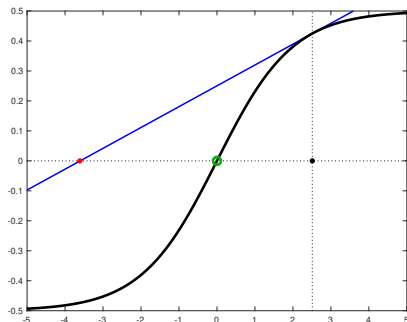
$$\nabla F(w)^T \Delta w = -F(w)$$

- ▶ Full Newton step

$$w \leftarrow w + \Delta w$$

Newton method can be unstable!

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w)$ and $\nabla F(w)$
- ▶ Compute Newton direction

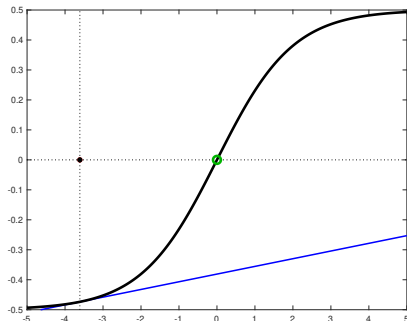
$$\nabla F(w)^T \Delta w = -F(w)$$

- ▶ Full Newton step

$$w \leftarrow w + \Delta w$$

Newton method can be unstable!

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w)$ and $\nabla F(w)$
- ▶ Compute Newton direction

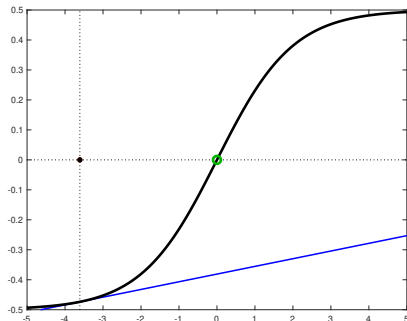
$$\nabla F(w)^T \Delta w = -F(w)$$

- ▶ Full Newton step

$$w \leftarrow w + \Delta w$$

Newton method can be unstable!

$$F(w) = 0$$



Newton Root finder

for $k = 0, 1, 2, \dots$

- ▶ Compute $F(w)$ and $\nabla F(w)$
- ▶ Compute Newton direction

$$\nabla F(w)^T \Delta w = -F(w)$$

- ▶ Newton step with **backtracking**

$$w \leftarrow w + \alpha \Delta w$$

Recap: Newton's method for unconstrained minimization

- Unconstrained problem

$$\min_x f(x)$$

- Necessary optimality condition $\nabla f(x) = 0$
- Solve using Newton method

$$F(w) = \nabla f(x), \quad \nabla F(w) = \nabla^2 f(x)$$

Compute Newton direction

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

- Modified Newton method with line search

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

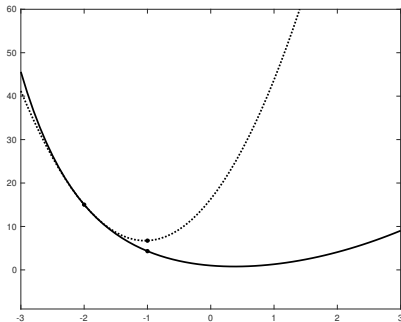
Newton algorithm with line search

1. For $k = 0$: select $\mathbf{x}_0$, $\varepsilon > 0$
2. For $k = 0, 1, 2, \dots$
 - ▶ set $\mathbf{s}_k = -H_k^{-1}\nabla f(\mathbf{x}_k)$, where $H_k := \nabla^2 f(\mathbf{x}_k)$
 - ▶ Choose step size $\alpha_k > 0$ (line search)
 - ▶ set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$
3. Stop if $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon$, else set $k = k + 1$ and repeat step 2

Newton direction: Another intuition

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(x_k)^T \delta x + \frac{1}{2} \delta x^T H_k \delta x + O(\|x - x_k\|^2)$$

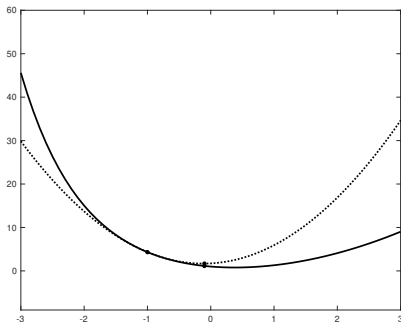


$$\min_x x^2 + e^{-1.2x}$$

Newton direction: Another intuition

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(x_k)^T \delta x + \frac{1}{2} \delta x^T H_k \delta x + O(\|x - x_k\|^2)$$

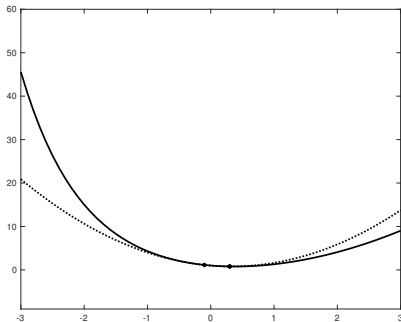


$$\min_x x^2 + e^{-1.2x}$$

Newton direction: Another intuition

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(x_k)^T \delta x + \frac{1}{2} \delta x^T H_k \delta x + O(\|x - x_k\|^2)$$

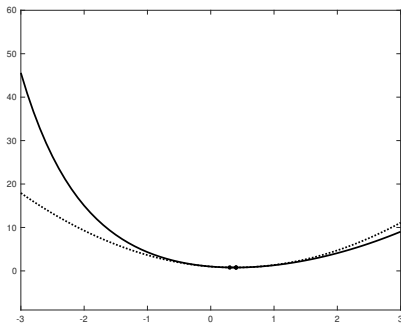


$$\min_x x^2 + e^{-1.2x}$$

Newton direction: Another intuition

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(x_k)^T \delta x + \frac{1}{2} \delta x^T H_k \delta x + O(\|x - x_k\|^2)$$



$$\min_x x^2 + e^{-1.2x}$$

Newton direction

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(\mathbf{x}_k)^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T H_k \delta \mathbf{x} + O(\|\mathbf{x} - \mathbf{x}_k\|^2)$$

Intuition:

If the function f is nearly quadratic, intuition suggests that $\mathbf{x} + \mathbf{s}_k$ with $\mathbf{s}_k := -H_k^{-1} \nabla f(\mathbf{x}_k)^T$ should be a good estimate of the minimizer of f .

Newton direction

Quadratic approximation

$$f_{k+1} = f_k + \nabla f(\mathbf{x}_k)^T \delta \mathbf{x} + \frac{1}{2} \delta \mathbf{x}^T H_k \delta \mathbf{x} + O(\|\mathbf{x} - \mathbf{x}_k\|^2)$$

Gradient of quadratic approximation

$$\nabla f_{k+1} = \nabla f(\mathbf{x}_k) + H_k \delta \mathbf{x}$$

This motivates to compute the search direction from

$$\nabla f(\mathbf{x}_k) + H_k \delta \mathbf{x} = 0$$

which requires the solution of the set of linear equations

$$H_k \delta \mathbf{x} = -\nabla f(\mathbf{x}_k) \quad (\text{Newton equations})$$

Assuming H_k is invertible, the Newton search direction is

$$\mathbf{s}_k = -H_k^{-1} \nabla f(\mathbf{x}_k)$$

Newton direction

If the Hessian H_k is positive definite, a step s_k in Newton direction satisfies

$$\delta f = -\nabla f(x_k)^T H_k^{-1} \nabla f(x_k) < 0$$

(Modified) Newton method

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

Pure Newton method: $\alpha = 1$

Newton algorithm

1. For $k = 0$: select $x_0, \varepsilon > 0$
2. For $k > 0$:
 - ▶ set $s_k = -H^{-1}(x_k)\nabla f(x_k)$
 - ▶ Choose step size $\alpha_k > 0$ (line search)
 - ▶ set $x_{k+1} = x_k + \alpha_k s_k$
3. Stop if $\|\nabla f(x_{k+1})\| < \varepsilon$, else set $k = k + 1$ and repeat step 2

Newton algorithm

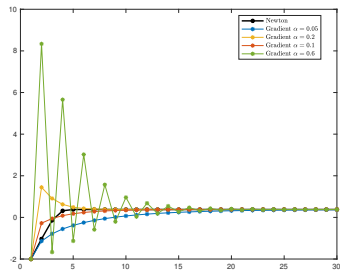
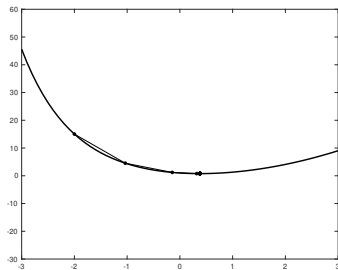
Based on the quadratic approximation of f at x , $\frac{1}{2}\nabla f(x_k)^T H_k^{-1} \nabla f(x_k)$ is an estimate of $f(x) - f(x^*)$.

1. For $k = 0$: select $x_0, \varepsilon > 0$
2. For $k > 0$:
 - ▶ set $s_k = -H^{-1}(x_k)\nabla f(x_k)$
 - ▶ **Stop** if $\frac{1}{2}\nabla f(x_k)^T H_k^{-1} \nabla f(x_k) \leq \varepsilon$
 - ▶ Choose step size $\alpha_k > 0$ (line search)
 - ▶ set $x_{k+1} = x_k + \alpha_k s_k$

Minor difference: Stopping criterion checked after computing search direction, rather than after the update.

Example

$$\min_x x^2 + e^{-1.2x}$$



Newton direction - some potential issues!

- ▶ Newton's method will usually fail (because unstable) if the nearest stationary point is not a minimum.
- ▶ Newton search direction is more efficient than steepest descent step provided that Hessian H_k is positive definite.
- ▶ Away from the solution, the Hessian may not be positive definite, and hence the Newton direction may not be a descent direction.

Newton's method with Hessian modification

Compute the Newton search direction s_k using a modified Hessian B_k instead of H_k

$$s_k = -B_k^{-1} \nabla f_k$$

where $B_k = H_k + E_k$.

- ▶ if $H_k \succ 0$, $E_k = 0$, otherwise choose E_k such that B_k is sufficiently positive definite.
- ▶ Simplest approach: Add a multiple of the identity: $E_k = \mu_k I$ with μ_k such that modified Hessian is positive definite

$$s_k = -(H_k + \mu_k I)^{-1} \nabla f_k$$

Newton method

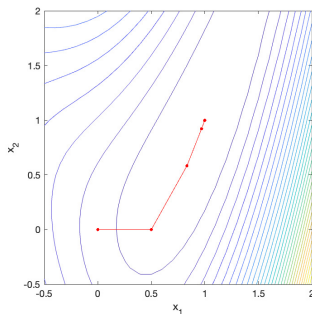
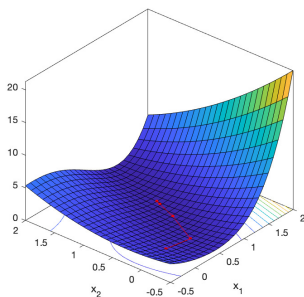
- ▶ exhibits rapid convergence, quadratic near x_*
- ▶ convergence insensitive to condition number
- ▶ scales well with problem size; only modest increase of required number of iterations for increasing number of design variables
- ▶ is insensitive to the choice of algorithm parameters
- ▶ main disadvantage is the cost of forming and storing the Hessian, and of cost computing the Newton step which requires solving a set of linear equations

Boyd and Vandenberghe (2004)

Example

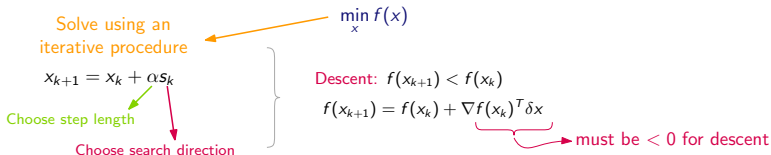
$$\min f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$

Newton direction with backtracking:



See code in blackboard: `Newton_linesearch.m`

Unconstrained Optimization



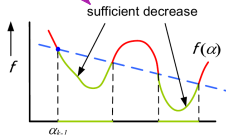
Gradient descent algorithm

1. Choose $s_k = -\nabla f(x_k)$
 2. Choose step length α
 3. Update $x_{k+1} = x_k + \alpha s_k$
- Exact line search
- $$\alpha = \arg \min_{\alpha \geq 0} f(x_k + \alpha s_k)$$
- Backtracking line search

Convergence can be slow!

Newton Search direction:

$$s_k = -H_k^{-1} \nabla f(x_k)$$



Quiz

When using exact Newton Method on the unconstrained optimization problem

$$\min_x 12x_1^2 + 4x_2^2 - 12x_1x_2 + 2x_1$$

which of the following statements is true?

- (A) Starting from $x = [0, 0]^T$ will result in a different x_1 , compared to starting from $x = [-1, -1]^T$, and $x_1 \neq x^*$.
- (B) Starting from $x = [0, 0]^T$ will result in the same x_1 , compared to starting from $x = [-1, -1]^T$, and $x_1 \neq x^*$.
- (C) Starting from $x = [0, 0]^T$ will result the same x_1 , compared to starting from $x = [-1, -1]^T$, which is also the minimizer $x_1 = x^*$.
- (D) Exact Newton method cannot be used for this problem, since Hessian is not positive definite.

Example

Consider the following unconstrained optimization problem:

$$\min_{x_1, x_2} f = 12x_1^2 + 4x_2^2 - 12x_1x_2 + 2x_1$$

- a) Start from $x_0^T = (-1, -1)$ and carry out a Newton line search (with constant step length, $\alpha = 1$).
- b) Start from $x_0^T = (0, 0)$ and carry out a Newton line search (with constant step length, $\alpha = 1$).
- c) Calculate the minimizer analytically.
Explain the result.

Approximating the Hessian

- ▶ Second derivatives may not be easy to calculate
- ▶ Quasi-Newton methods:

Build up curvature information from observations of f_k and ∇f_k

⇒ Best general purpose methods for unconstrained
(and also constrained) gradient-based local search

Quasi-Newton method

- Approximate Hessian by means of update formula:

$$\begin{aligned}\hat{H}_{k+1} &= \hat{H}_k + \Delta H_k \\ \hat{B}_{k+1} &= \hat{B}_k + \Delta B_k\end{aligned}\quad (B_k \triangleq H_k^{-1})$$

- General update matrix:

$$\Delta A_k = auu^T + bv v^T$$

Rank one: $b = 0$

Rank two: $b \neq 0$

Quasi-Newton method

- Most widely used is rank-two BFGS update:

$$\hat{H}_{k+1} = \hat{H}_k - \frac{\hat{H}_k s_k s_k^T \hat{H}_k}{s_k^T \hat{H}_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

$$\hat{B}_{k+1} = \left\{ I - \frac{s_k y_k^T}{y_k^T s_k} \right\} \hat{B}_k \left\{ I - \frac{y_k s_k^T}{y_k^T s_k} \right\} + \frac{s_k s_k^T}{y_k^T s_k}$$

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

- BFGS allows inexact line-search with Armijo/Wolf/Goldstein sufficient decrease conditions
- Alternative DFP requires exact line-search

Example: Hands-on Demo

Consider :

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2 \quad (1)$$

- ▶ Given initial design $x_0 = (-1, -1)$, and initial guess of the Hessian matrix $\hat{H}_0 = I$ (identity matrix), calculate the BFGS Hessian estimate $\hat{H}$

Large scale optimization techniques

If the cost of factoring the Hessian becomes prohibitive:

- ▶ solve $H_k s_k = -\nabla f_k$ using iterative linear algebra techniques (preconditioned conjugate gradient or Lanczos) and terminate early to compute approximation to s_k . (inexact Newton)

If the cost of storing the Hessian becomes prohibitive:

- ▶ Use limited memory variants of quasi-Newton (L-BFGS).

Refer to Nocedal and Wright (2006) for further reading about large scale numerical optimization.

Unconstrained Optimization

$$\min_x f(x)$$

Necessary condition of optimality: $\nabla_x f(x^*) = 0$

Analytical solution for
a few special cases

$$\nabla f(x) = 0$$

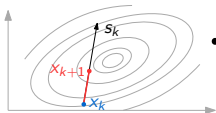
n unknowns & n equations

Solve using an
iterative procedure

$$x_{k+1} = x_k + \alpha s_k$$

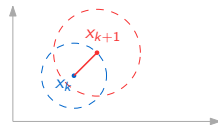
Choose step length

Choose search direction



- **Line search methods:** Choose direction first, then the step size.

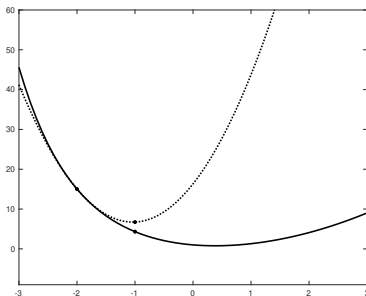
- **Trust region methods:** Choose step size first, then the step direction.



Recap: Newton method

Newton step is obtained by minimizing the second order approximation of f around x_k

$$f_{k+1} = f_k + \nabla f(x_k)^T \delta x + \frac{1}{2} \delta x^T H_k \delta x + O(\|x - x_k\|^2)$$



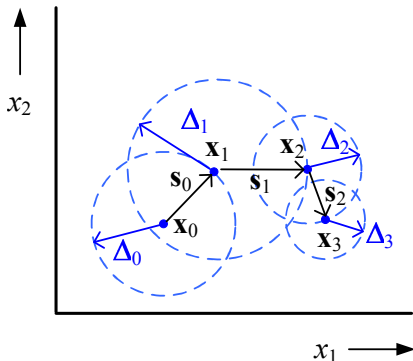
f_{k+1} is a good approximation when δx is in a small neighborhood (which we can trust).

Local exploration: trust region methods

$$s_k = \arg \min_{s \in \mathbb{R}^n} \quad m_k(s) = f_k + \nabla f_k^T s + \frac{1}{2} s^T H_k s$$

s.t. $\|s\| \leq \Delta_k$

- A ball ($\|s\|_2$) of radius Δ_k is a common trust region



Key ingredient: Strategy for choosing the trust region Δ_k

Choosing the trust region

- Measure quality of approximation:

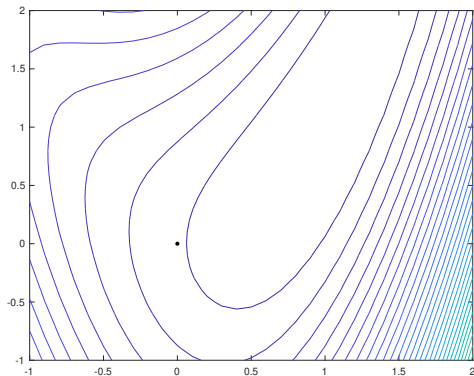
$$\begin{aligned}r_k &= \frac{\text{actual reduction}}{\text{predicted reduction}} \\&= \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)} \\&= \frac{f(x_k) - f(x_k + s_k)}{-(\nabla f_k^T s_k + \frac{1}{2} s_k^T H_k s_k)}\end{aligned}$$

- Update trust-region radius:

$$\Delta_{k+1} = \begin{cases} \text{if } 0.75 < r_k < 1.25 & \rightarrow \text{increase } \Delta \\ \text{if } r_k < 0.25 & \rightarrow \text{decrease } \Delta \\ \text{else} & \rightarrow \text{do not change } \Delta \end{cases}$$

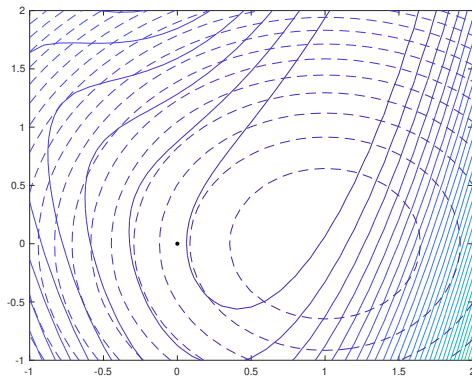
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



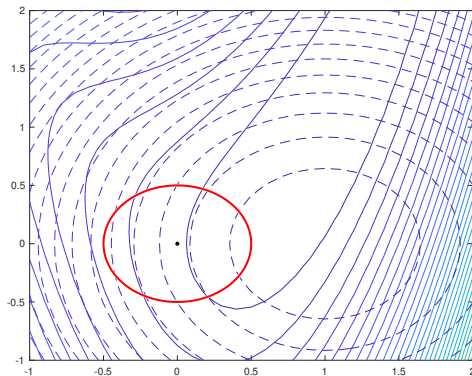
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



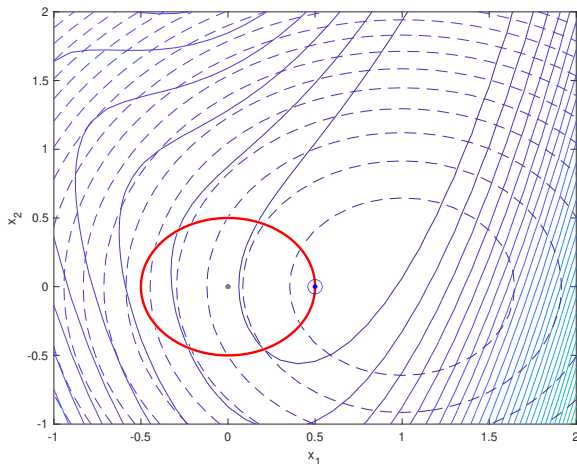
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



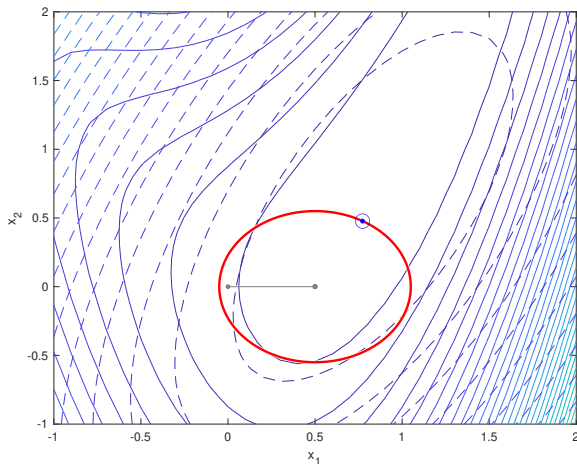
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



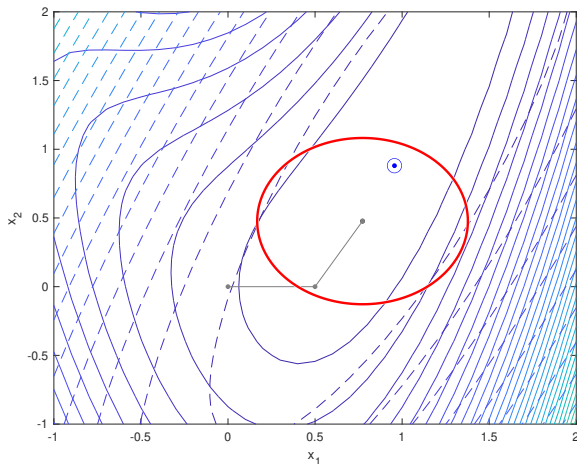
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



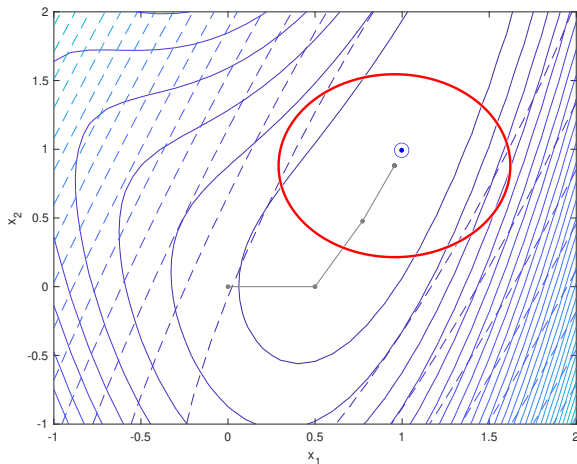
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



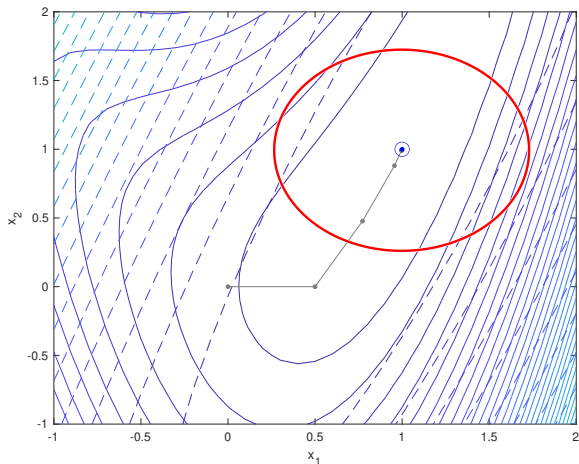
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



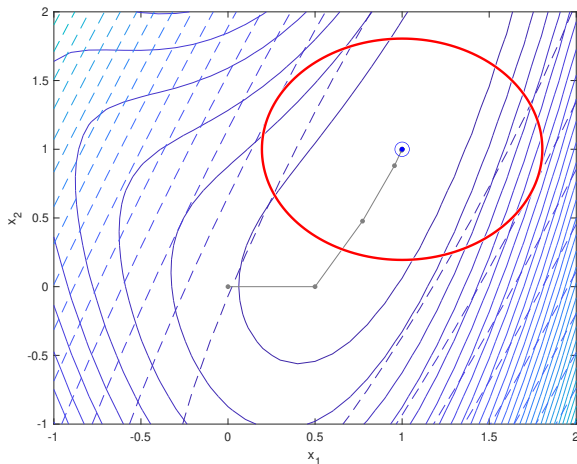
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



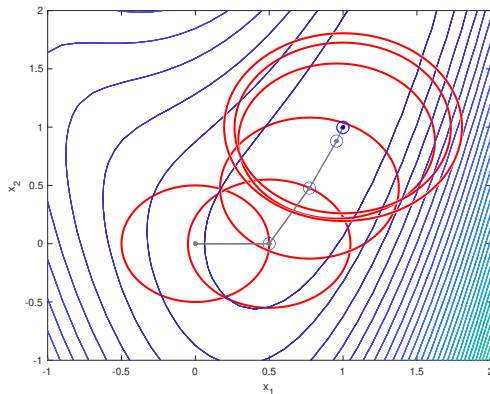
Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



Example: Rosenbrock function

$$f = (1 - x_1)^2 + (x_2 - x_1^2)^2$$



Inexact solution of the subproblem

- ▶ Solving the subproblem at x_k to high precision may be expensive and is often unnecessary
- ▶ Instead, one may compute an approximate step s_k that lies within the trust region, and gives “sufficient” reduction in the model.

$$f(x_k + s_k) \leq f(x_C)$$

quantified in terms of the so-called Cauchy point x_C of the subproblem at x_k .

- ▶ The Cauchy point is found by minimizing the quadratic model approximation m_k in the steepest descent direction subject to the trust region bound
- ▶ The Cauchy point can be cheaply calculated

Finding the Cauchy Point

- Find vector s_k that solves the linear approximation of the function, subject to trust region constraint

$$\begin{aligned} \mathbf{s}_k &= \arg \min_{\mathbf{s} \in \mathbb{R}^n} f_k + \nabla f_k^T \mathbf{s} \\ \text{s.t. } \|\mathbf{s}\| &\leq \Delta_k \end{aligned}$$

- Calculate a scalar τ_k that minimizes

$$\begin{aligned} \tau_k &= \arg \min_{\tau \in \mathbb{R}} f_k + \tau \nabla f_k^T \mathbf{s}_k + \frac{1}{2} \tau^2 \mathbf{s}_k^T H_k \mathbf{s}_k \\ \text{s.t. } \|\tau \mathbf{s}_k\| &\leq \Delta_k \end{aligned}$$

- Calculate the Cauchy point as

$$\mathbf{s}_k^c = \tau_k \mathbf{s}_k$$

What did you learn?

Unconstrained Optimization

$$\min_x f(x)$$

Necessary condition of optimality: $\nabla_x f(x^*) = 0$

Analytical solution for
a few special cases

$$\nabla f(x) = 0$$

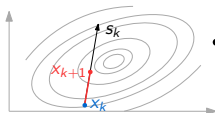
n unknowns & n equations

Solve using an
iterative procedure

$$x_{k+1} = x_k + \alpha s_k$$

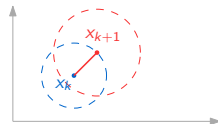
Choose step length

Choose search direction



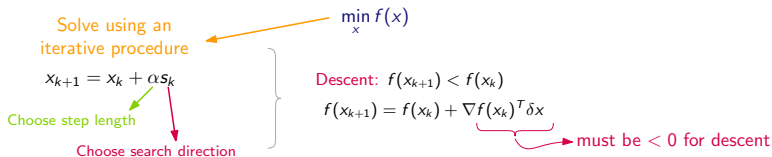
- **Line search methods:** Choose direction first, then the step size.

- **Trust region methods:** Choose step size first, then the step direction.



What did you learn?

Unconstrained Optimization



Gradient descent algorithm

1. Choose $s_k = -\nabla f(x_k)$
2. Choose step length α
 - Exact line search
 $\alpha = \arg \min_{\alpha \geq 0} f(x_k + \alpha s_k)$
 - Backtracking line search
3. Update $x_{k+1} = x_k + \alpha s_k$

Convergence can be slow!

Newton Search direction:

$$s_k = -H_k^{-1} \nabla f(x_k)$$

Rapid convergence, but requires computing/storing Hessian H_k

Use Quasi-Newton methods (e.g. BFGS)

