Newton's Method for Optimization

Caleb Maddry

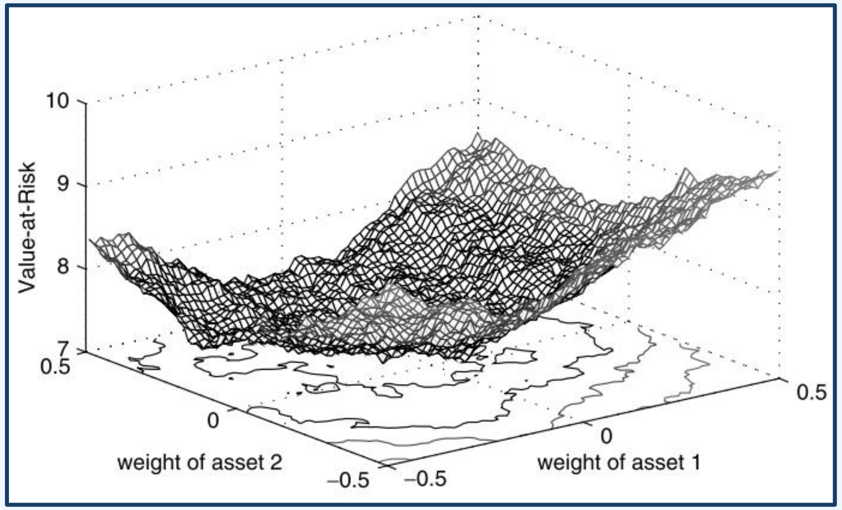
What is optimization?

Minimization of a function given a set of inputs

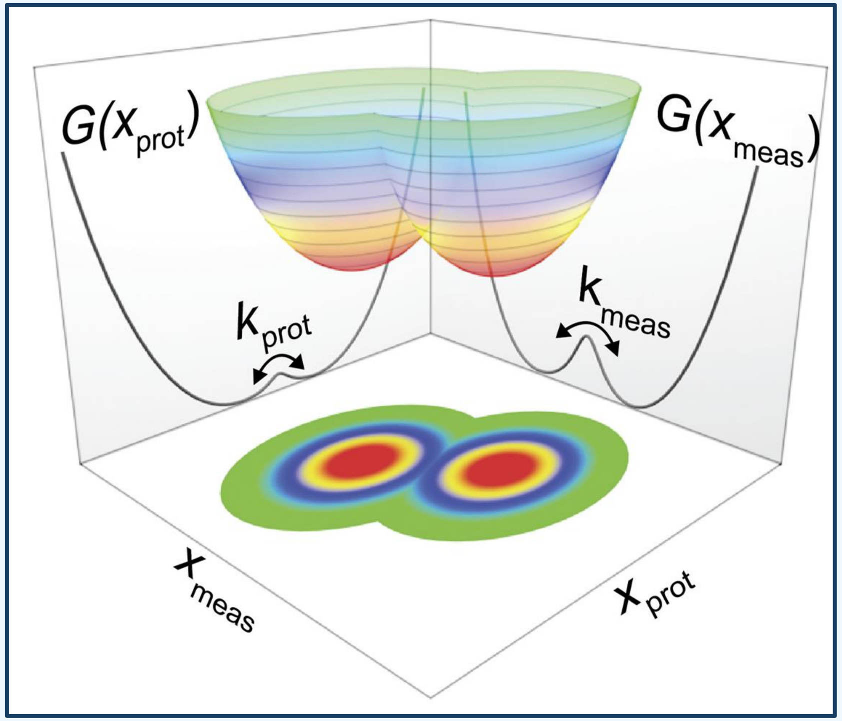
Applications of numerical optimization

- Science/Engineering
  - Molecular modeling
  - Control systems
- Finance
  - Portfolio optimization
  - Econometric models
- Policy
  - Resource allocation
  - Risk models

Applications of optimization are **everywhere**

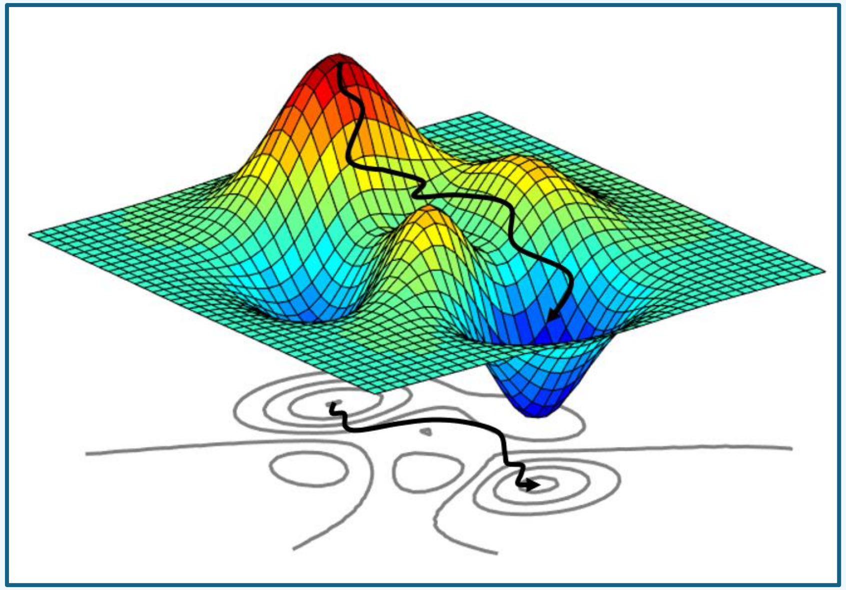


Gilli et al. (2019)



Edwards et al. (2020)

Line-search methods



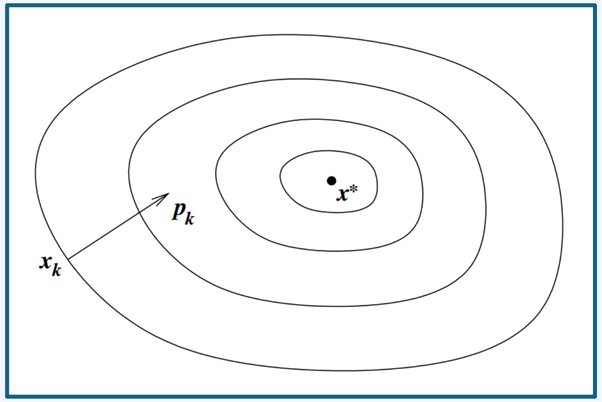
Iterative method:

$$x_{k+1} = x_k + \alpha_k d_k$$

1. Step direction, d_k
   - Determines **which direction** to take the step

2. Step length, α_k
   - Determines **how large** the step should be

The descent direction and Steepest Descent



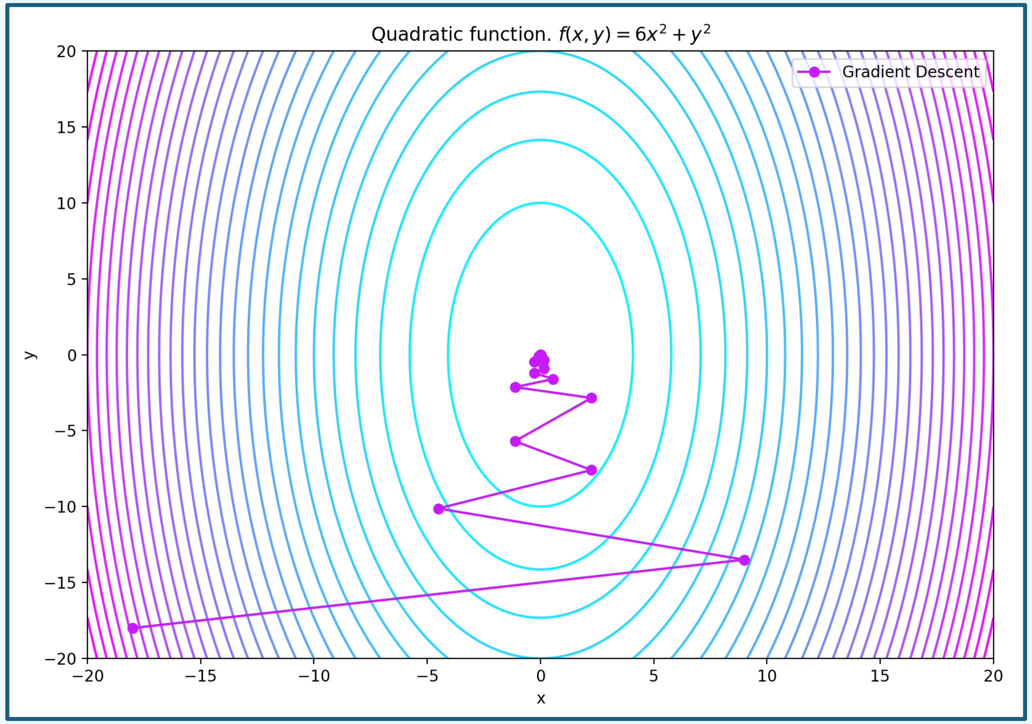
Descent direction: $\nabla f(x_k) \cdot d_k < 0$

Steepest descent is a basic optimization method

- Uses the gradient as the descent direction:

$$d_k = -\nabla f(x_k)$$

- Converges linearly

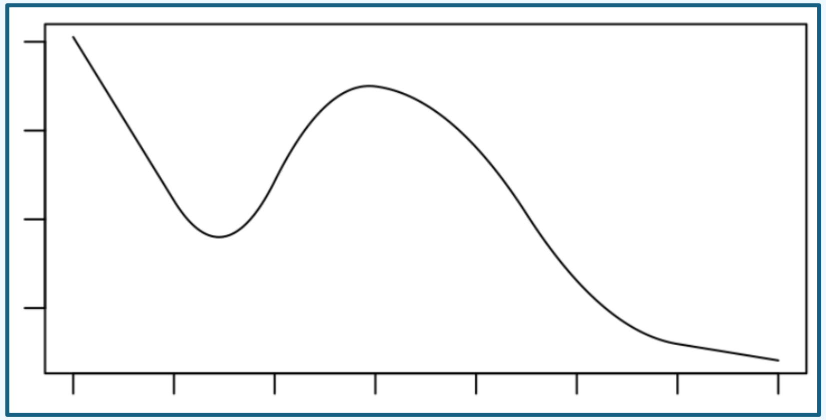


Step length is chosen with a **backtracking line search**

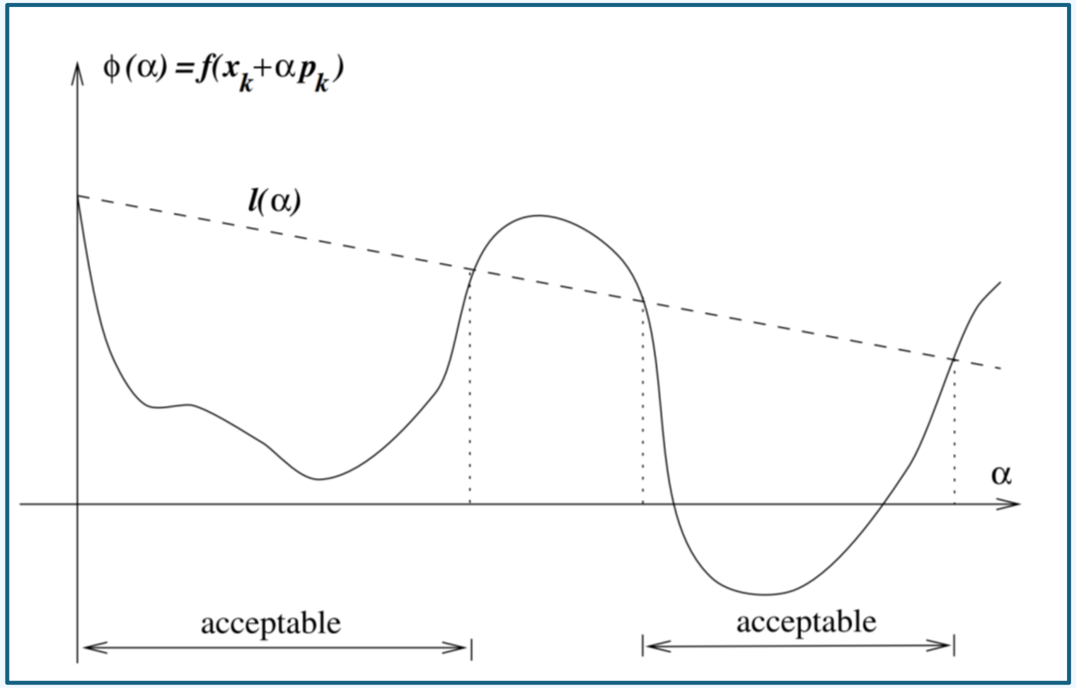
Backtracking line search to find α

Find an inexact solution to:

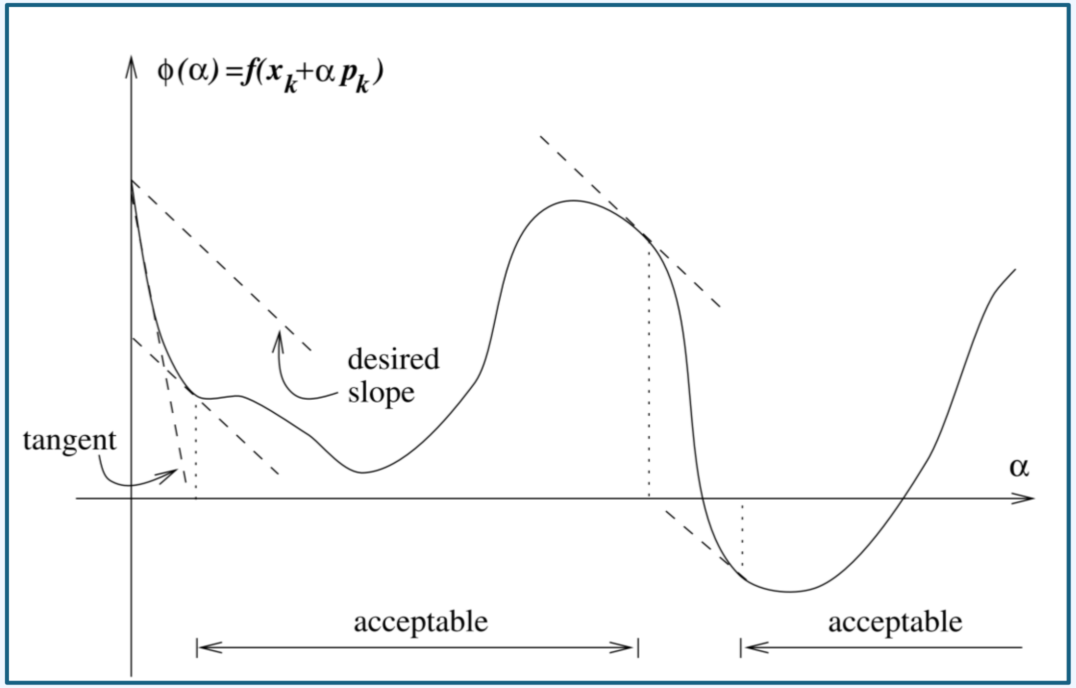
$$\min_{\alpha > 0} f\left(x_k + \alpha p_k\right)$$



Sufficient decrease condition

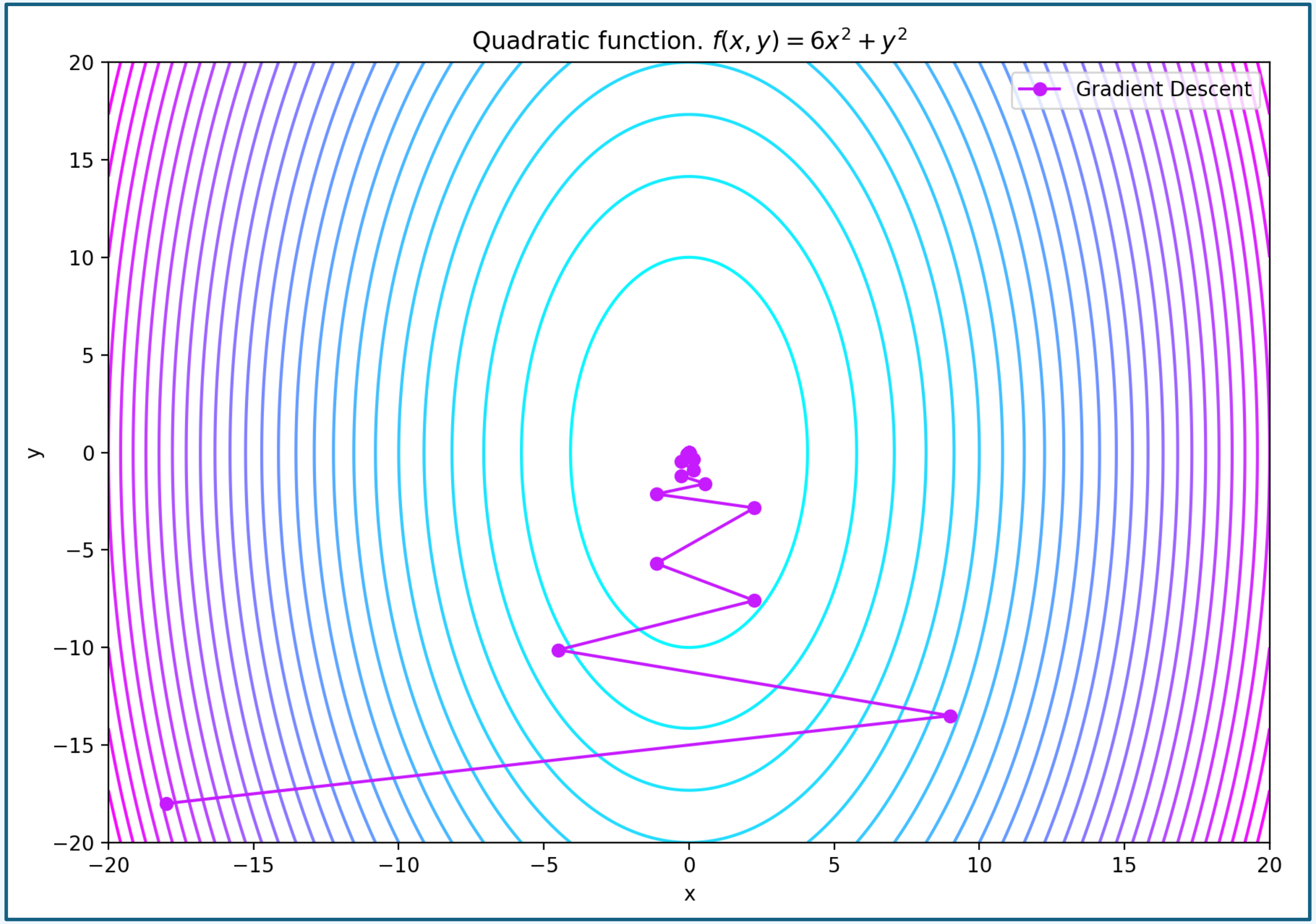


Curvature condition



Nocedal

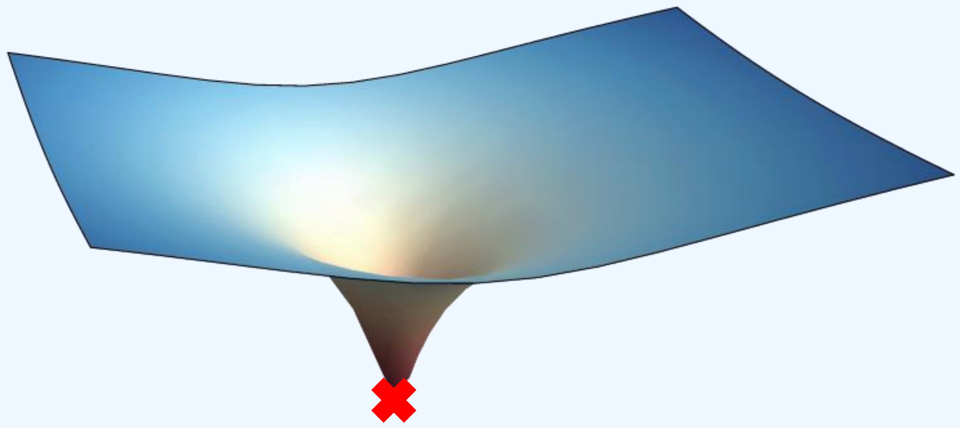
Steepest Descent



Reframing optimization as a root finding problem

Goal: find the minimum of a function

- Use the fact the **gradient is zero** at the minimum



Can use tools developed for root finding

- Newton's method
  - Enables quadratic convergence
  - Can use quasi-Newton methods
    - **Broyden-Fletcher-Goldfarb–Shanno (BFGS)**
    - **Lazy-Newton**

Newton's method for optimization

Want to find a method: $x_{k+1} = x_k + \alpha_k d_k$

1. Generate an approximation of the function at a step

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k + \alpha p) p$$

2. Solve the minimization problem to obtain the direction p_k

$$p_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

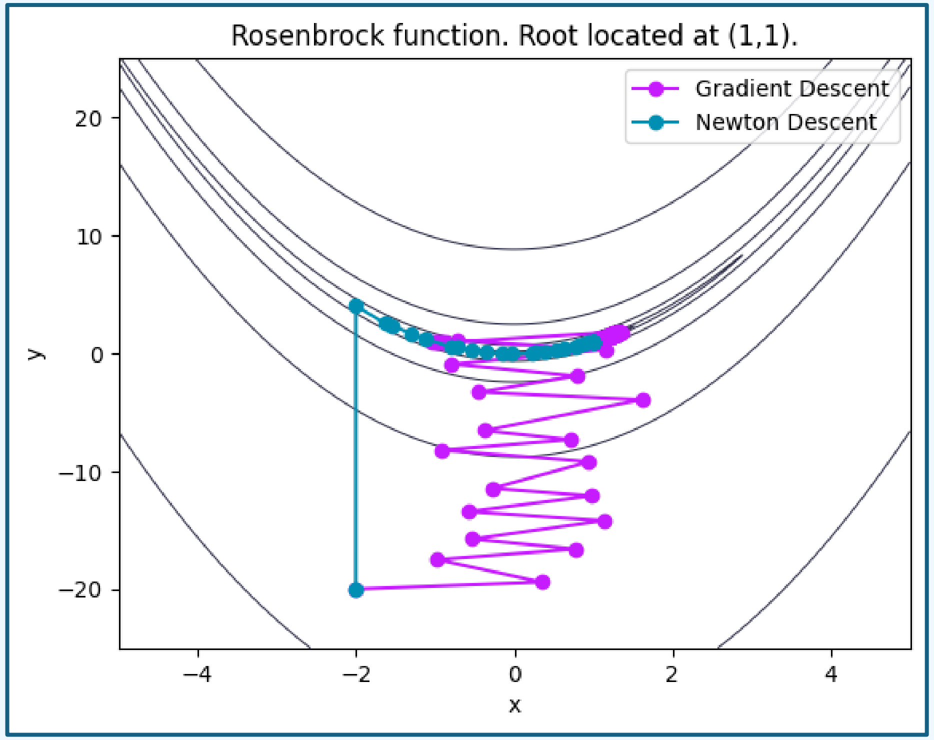
3. Newton's method:

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

Newton's method for optimization

Iteration:

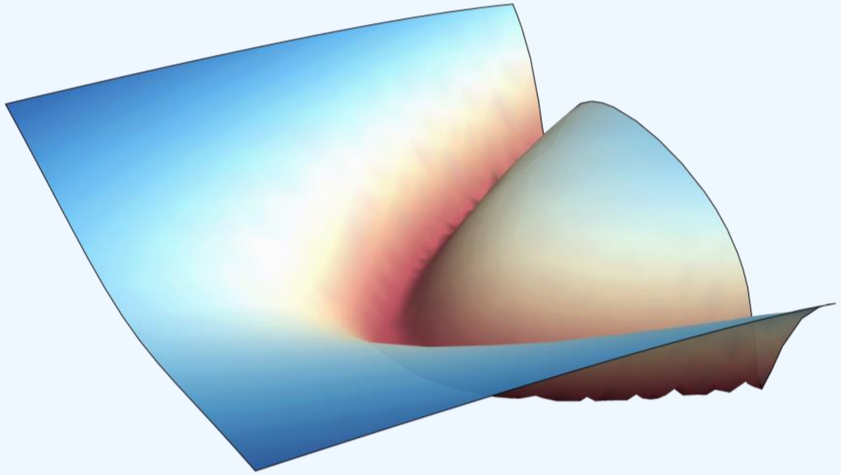
$$x_{k+1} = x_k - \alpha_k(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$$



Requirements for convergence:

1. The Hessian matrix must be SPD to satisfy the descent property
   - Initial guess must be in a region where the Hessian is SPD

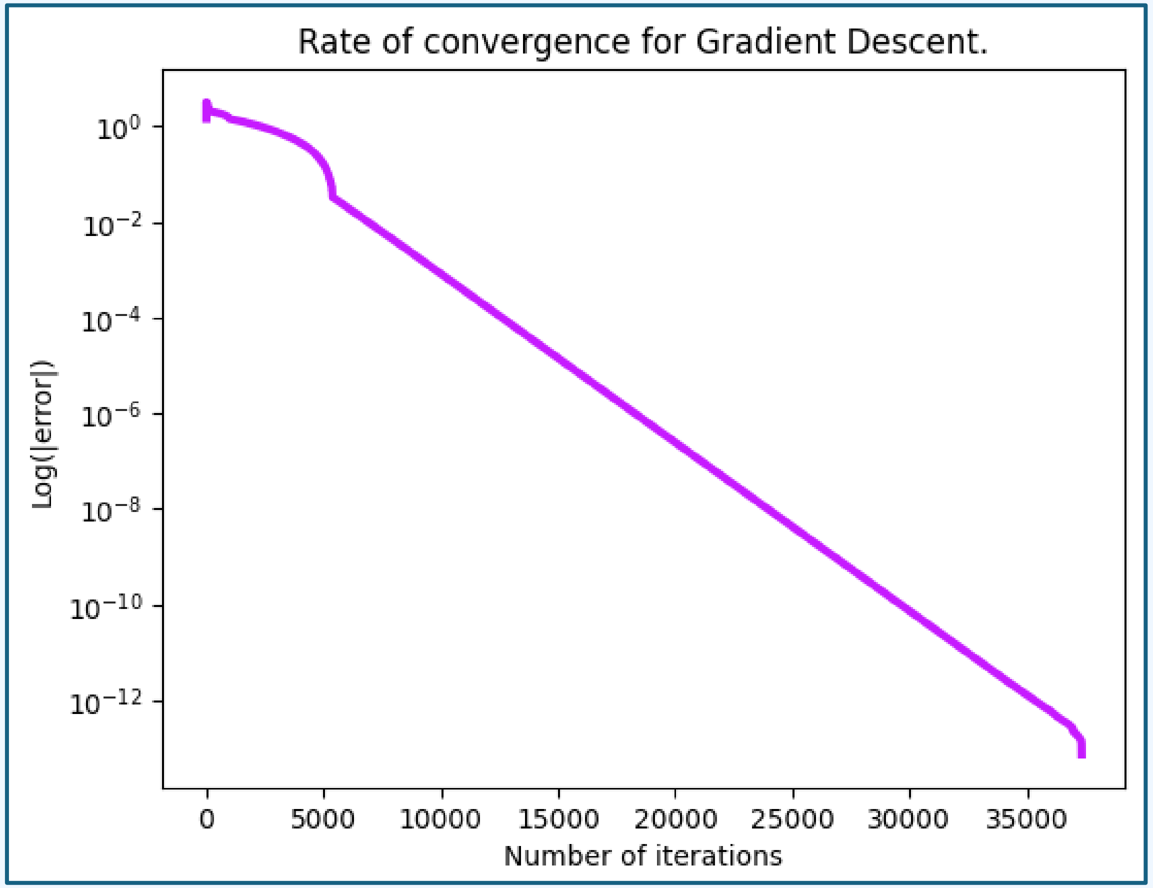
Rosenbrock Function



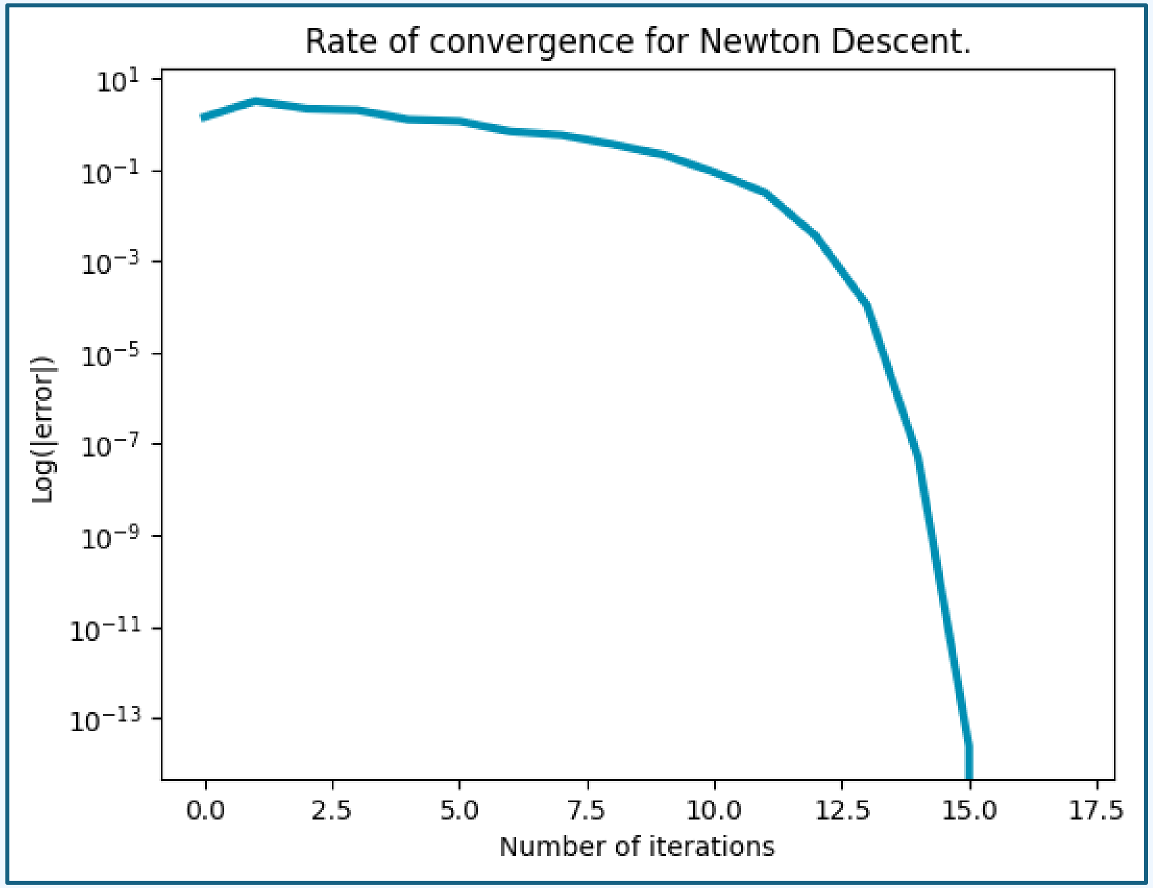
Newton’s method results in quadratic convergence

- Gradient descent converged in **32,291 iterations**
- Newton descent converges in **16 iteration**

Gradient Descent:



Newton descent:



Quasi-Newton methods

Motivation:

- Computing the Hessian matrix can be expensive
- Solving $-(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$ can also be expensive
- More important for higher dimensional functions

Can use quasi-Newton methods to **improve** algorithm

- Lazy-Newton/Shaminski
  - Converge linearly
- Broyden-Fletcher-Goldfarb–Shanno (BFGS)
  - Most robust/popular method
- DFP Algorithm
- Symmetric rank-1 update (SR1)

BFGS as an alternative to Newton’s method

BFGS

- Secant equation: $$H_{k+1} y_k = s_k$$
- Impose a condition to ensure the inverse Hessian approximation is SPD $$y_k^T s_k > 0$$
- Solve minimization problem to find the “closest” iteration

$$\min_H ||H - H_k||$$

Benefits:

1. Cost per iteration is less than Newton’s method

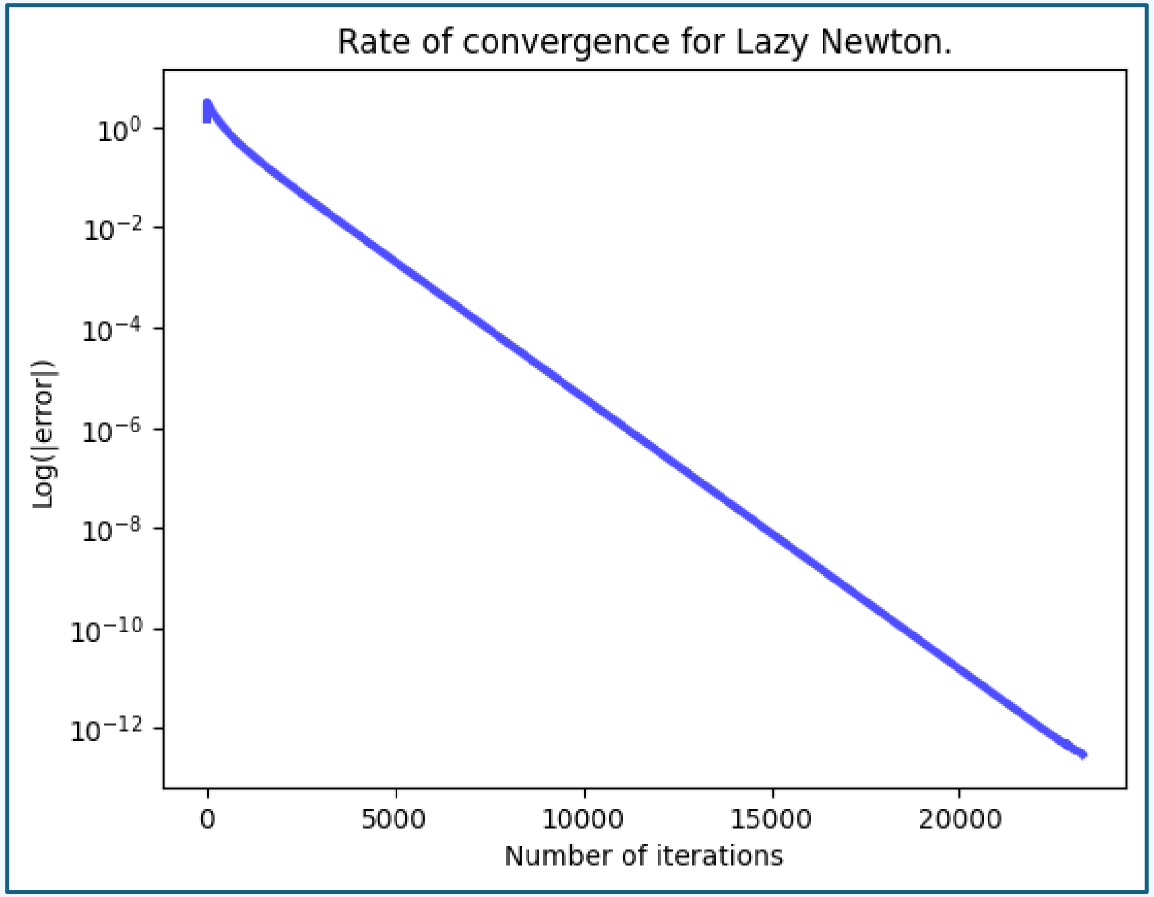
   More important for higher dimensional function

Quasi-Newton results

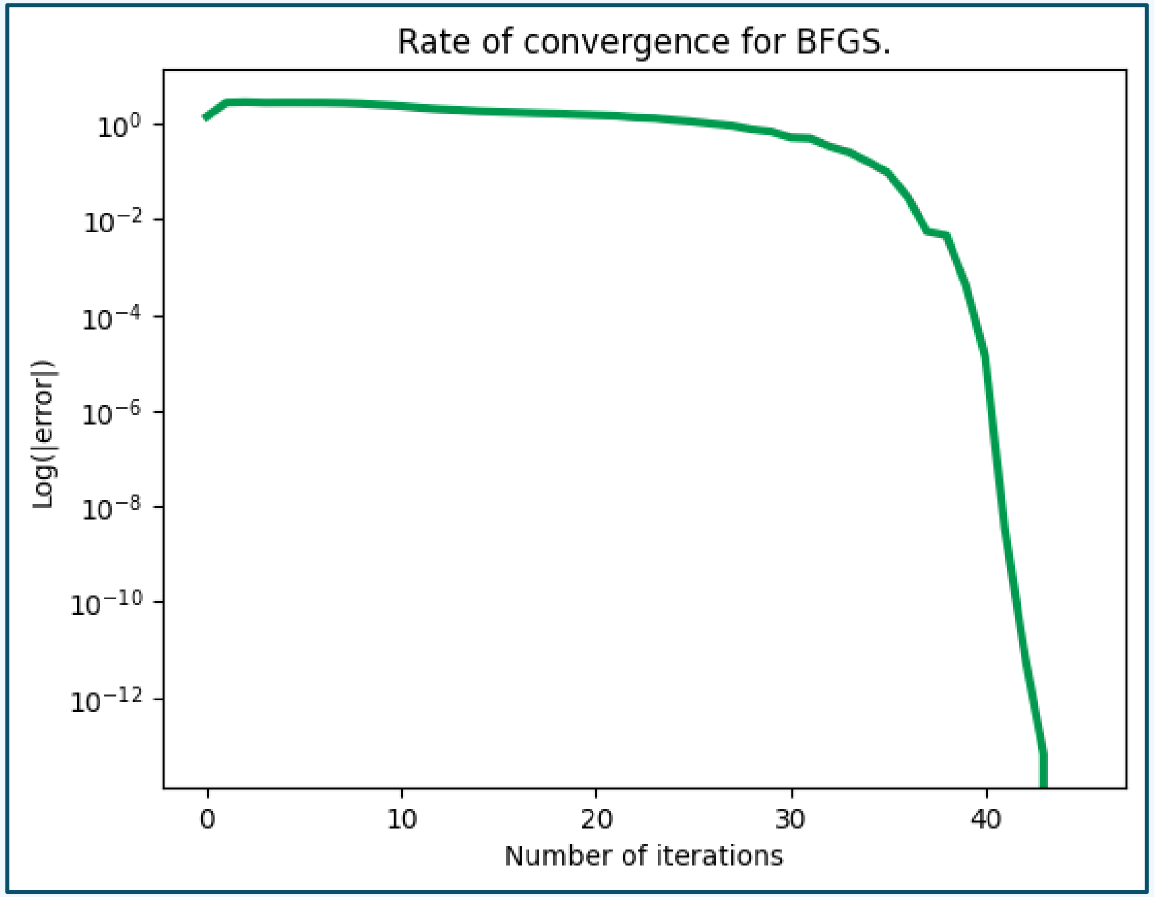
Lazy Newton computes the Hessian once

BFGS using the identity matrix as the initial choice of the approximate Hessian

Lazy Newton:



BFGS:



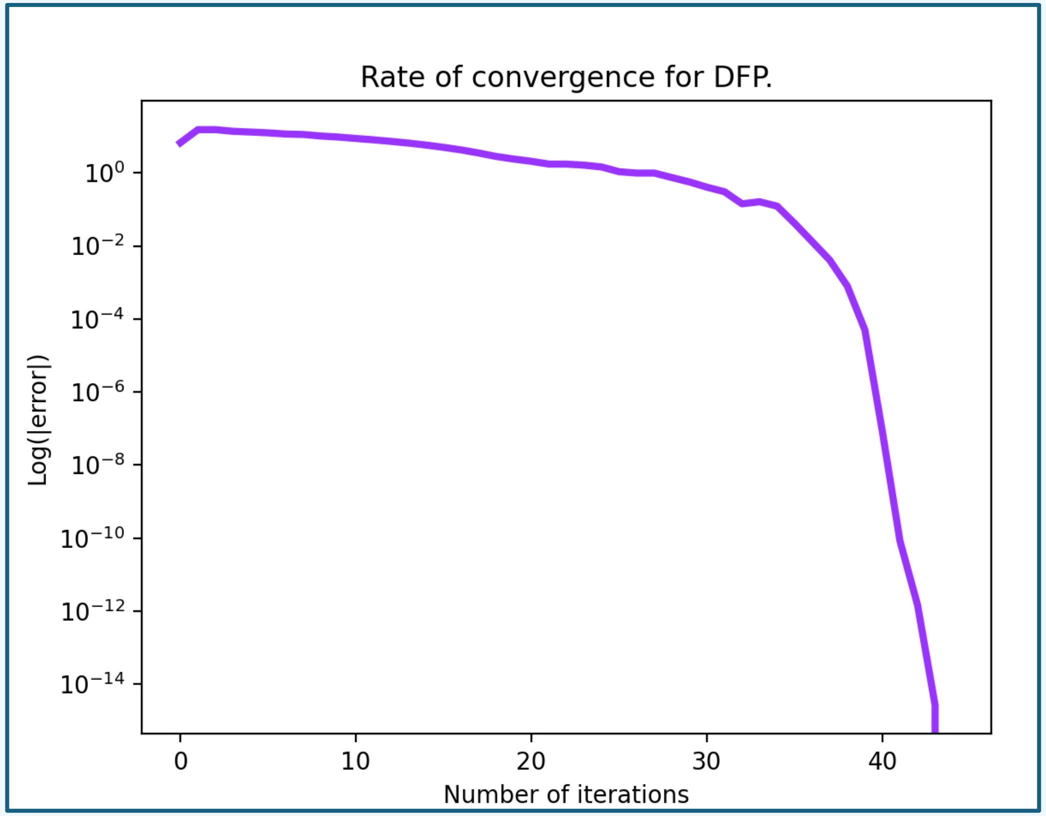
Additional quasi-Newton methods

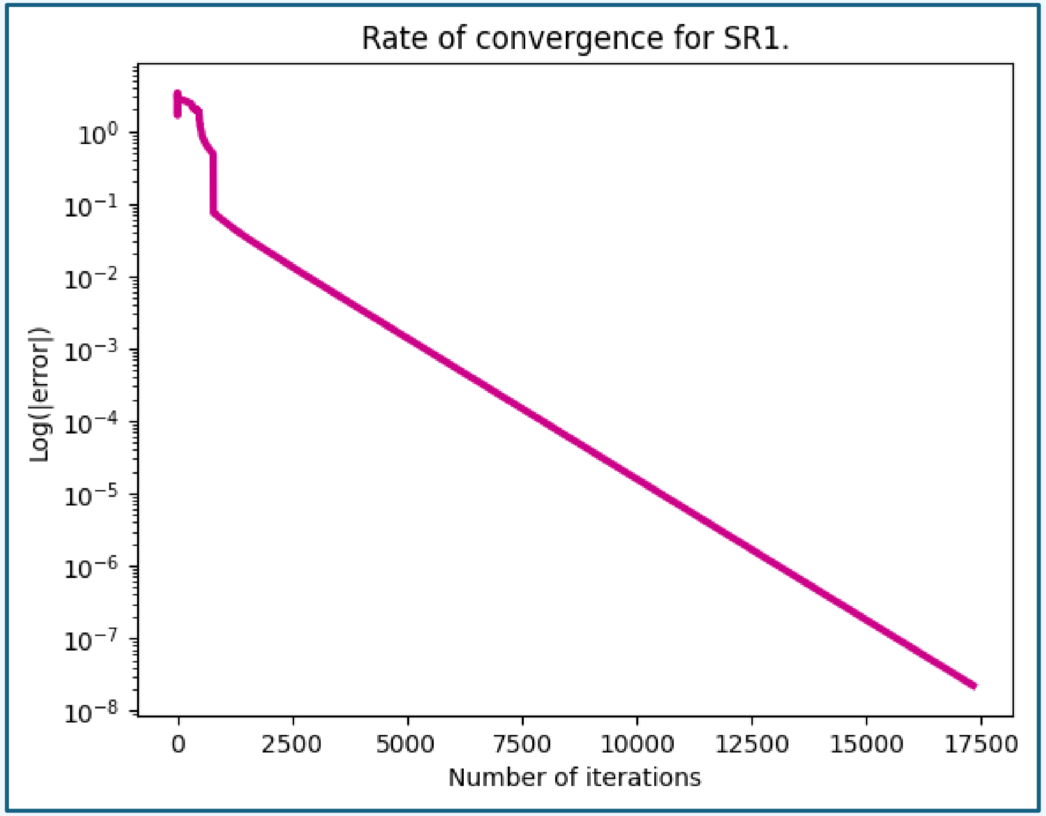
DFP Algorithm

Like BFGS but derived by solving the secant equation with the Hessian itself

Symmetric rank-1 update (SR1):

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

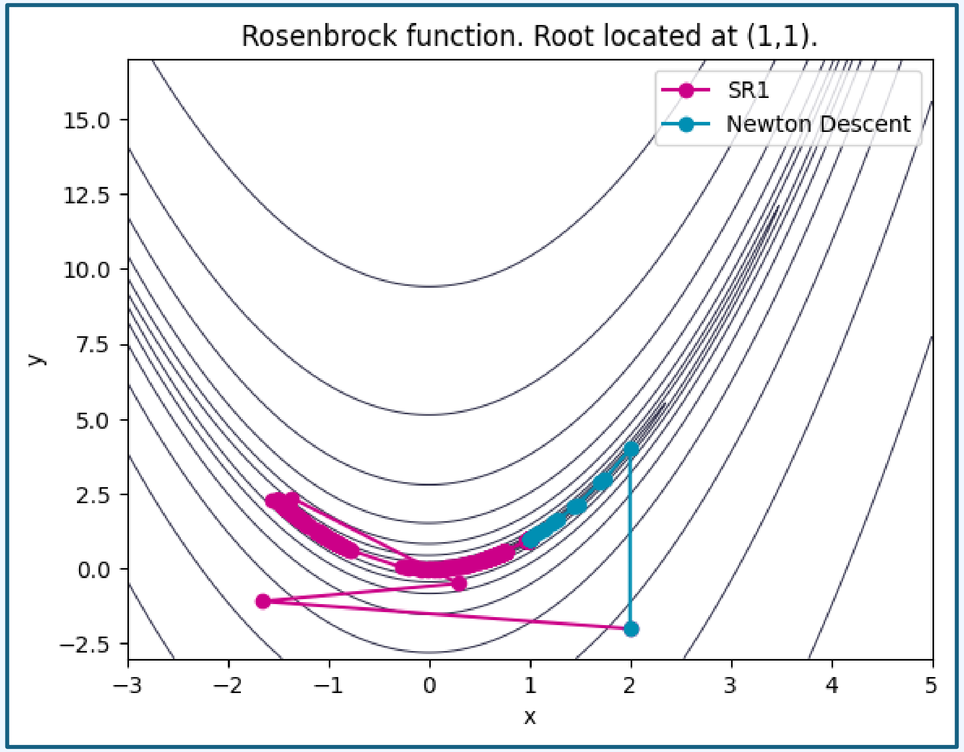


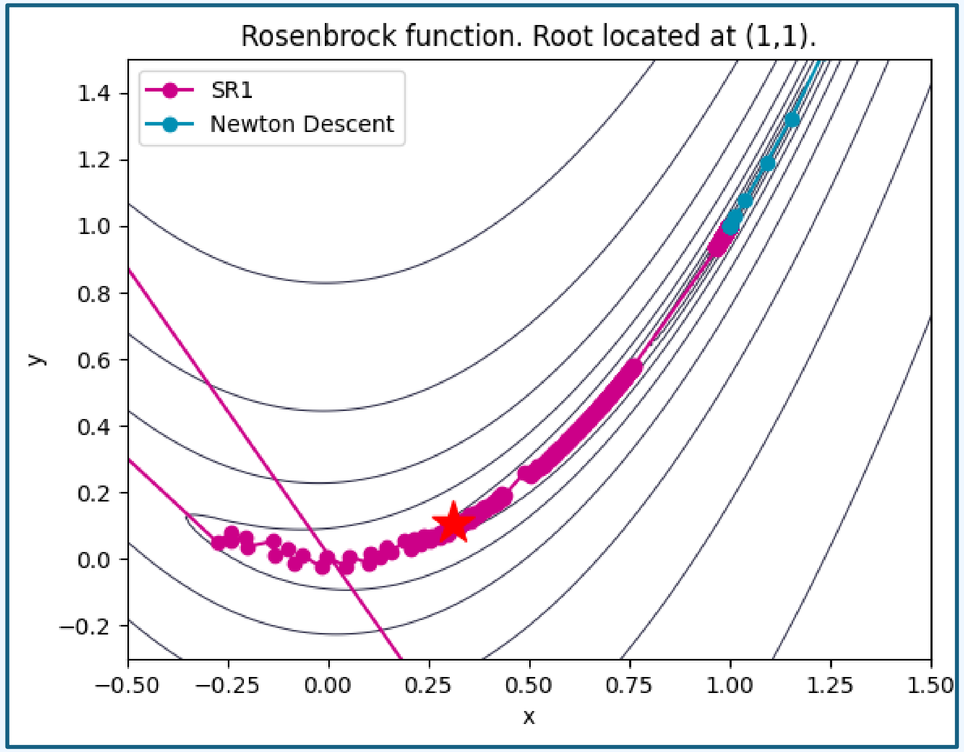


Symmetric rank-1 update (SR1): $B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$

Denominator goes to zero

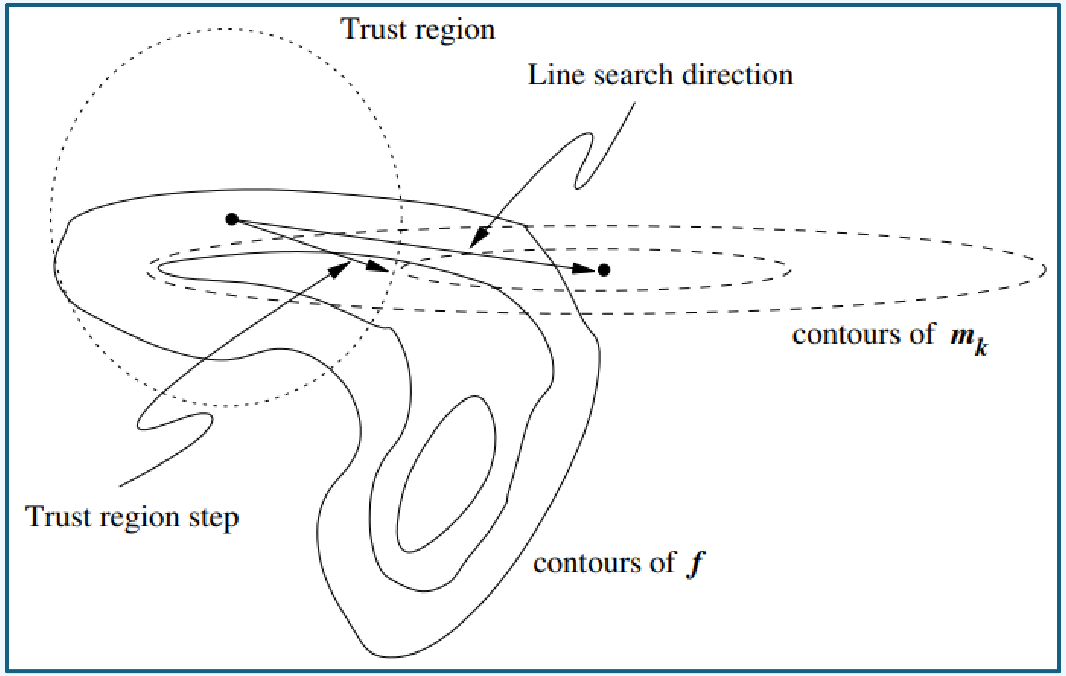
- There is no symmetric rank-one update that satisfies the secant equation
- Behaves like Lazy Newton





Trust region methods

1. Choose a maximum step length Δ_k
    - Region we "trust" our approximation
        - Based on how close the approximation is to the real function



Nocedal

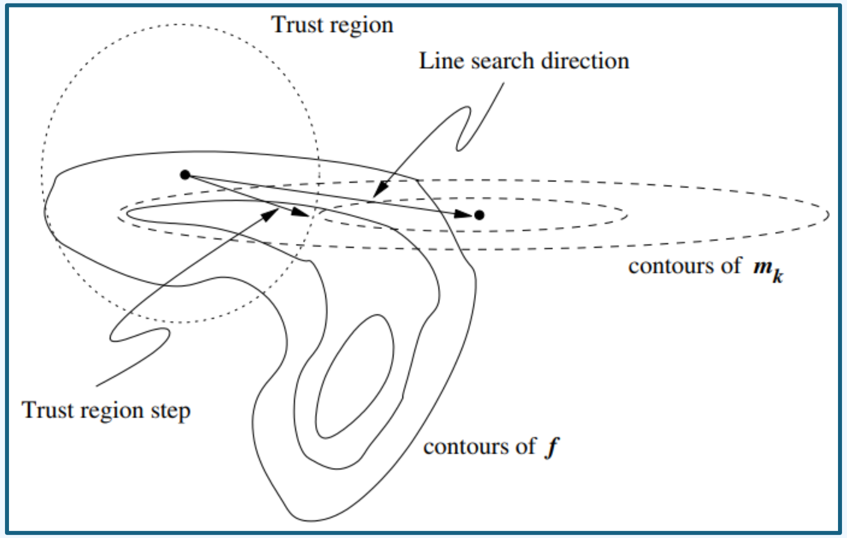
2. Minimize the **model function** within this region
    - Typically, an **inexact minimization** is performed
        - An exact approximation is overkill and too expensive for most applications

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k + \alpha p) p$$

Constraint

$$\|p\| \leq \Delta_k$$

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k + \alpha p) p$$



Nocedal

Ensure our step is within the trust region:

$$\|p\| \leq \Delta_k$$

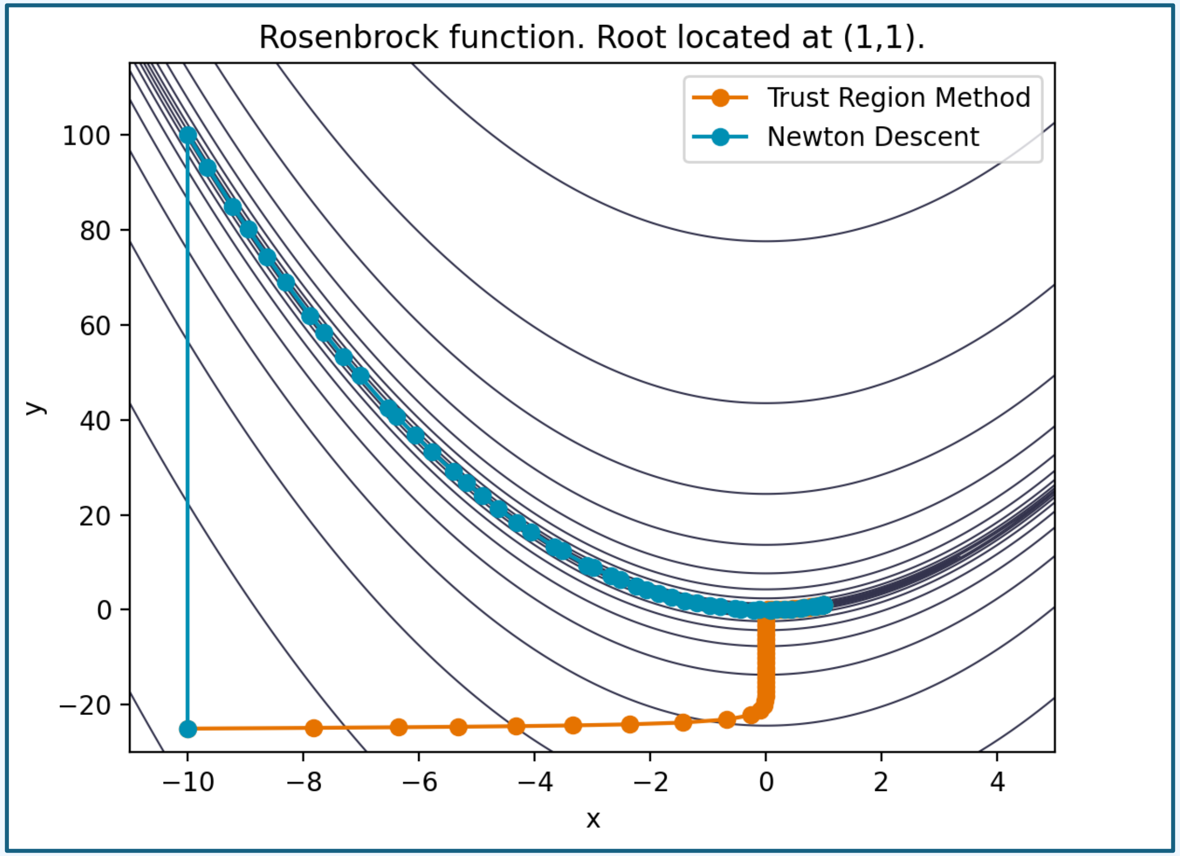
1. How do we find our trust region, Δ_k?

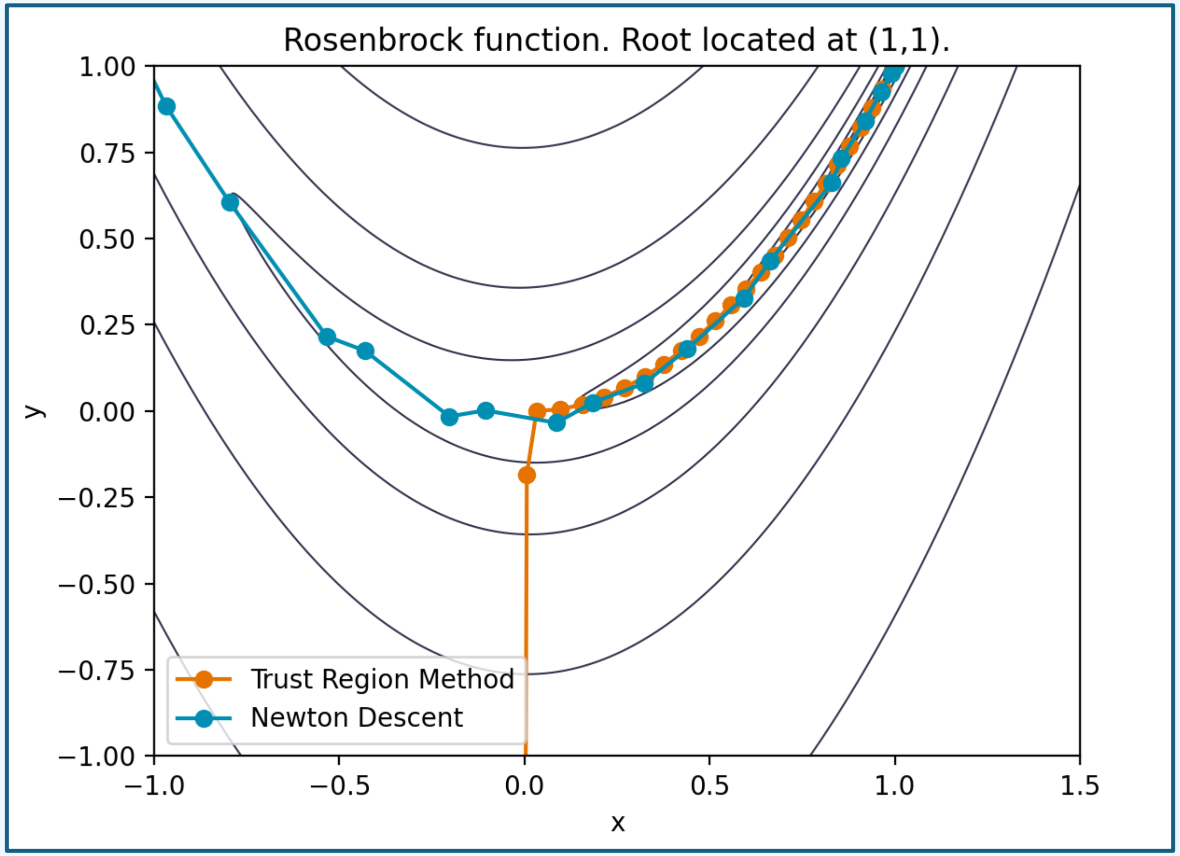
$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

2. Approximately solve the minimization problem and update the iterate!
   - Achieved with a method like the Dogleg Method

Trust region methods

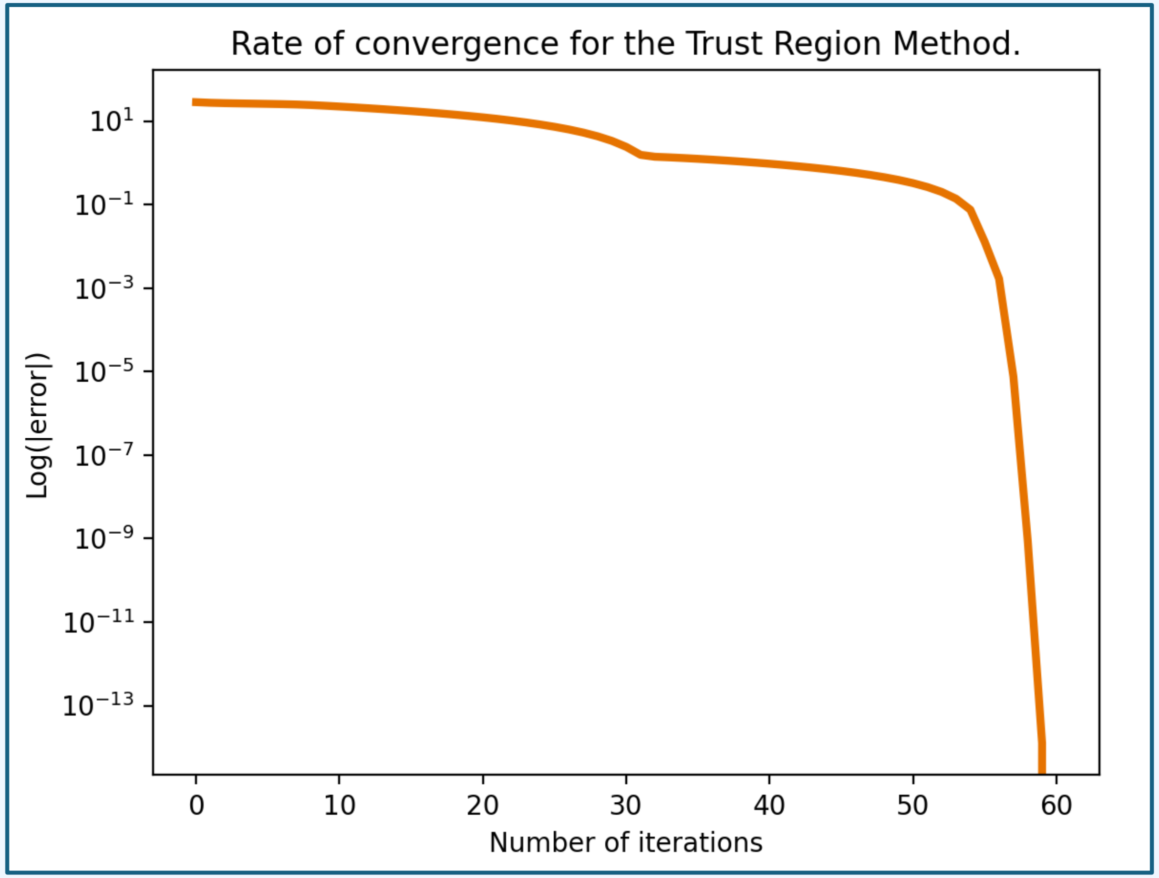
- TR converged in **64 iterations**
- Newton descent converges in **54 iteration**



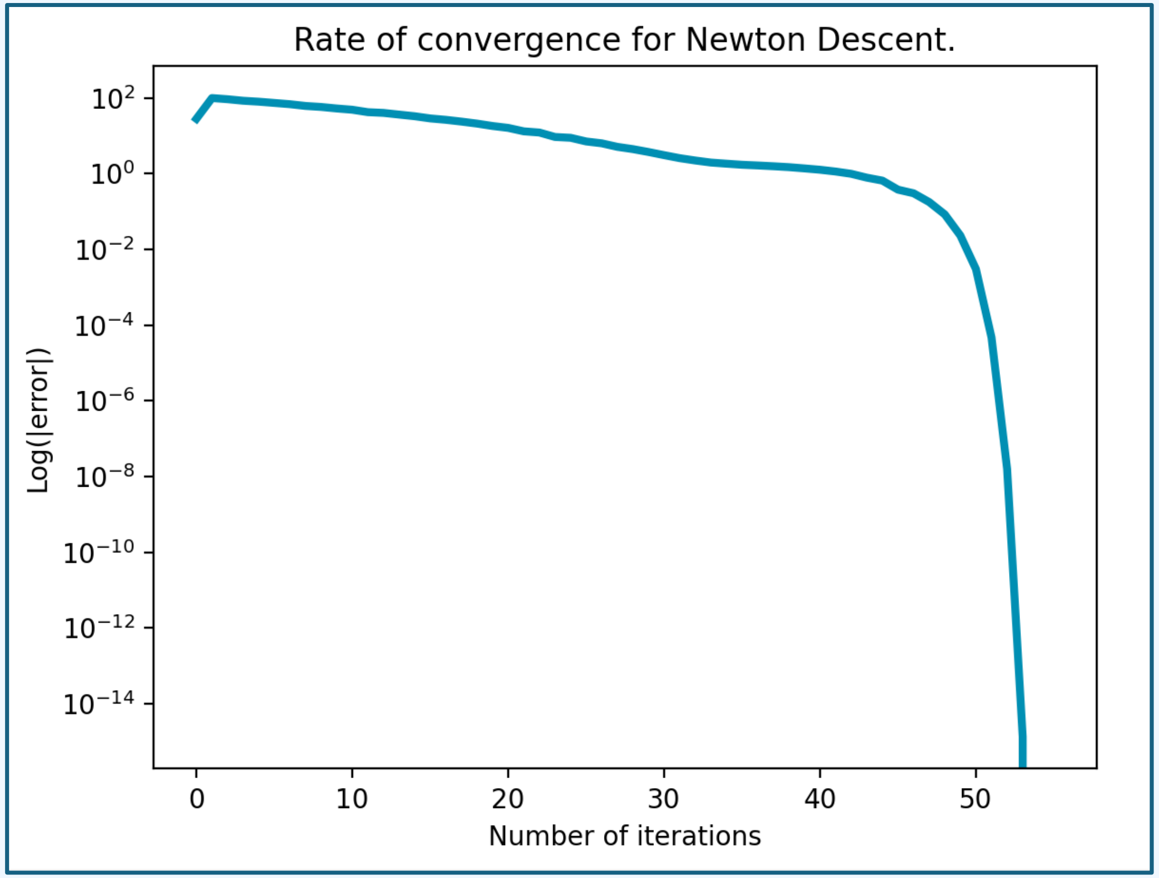


Trust region methods maintain quadratic convergence

Trust region



Newton Descent



Questions?