



## **Newton method. Quasi-Newton methods**

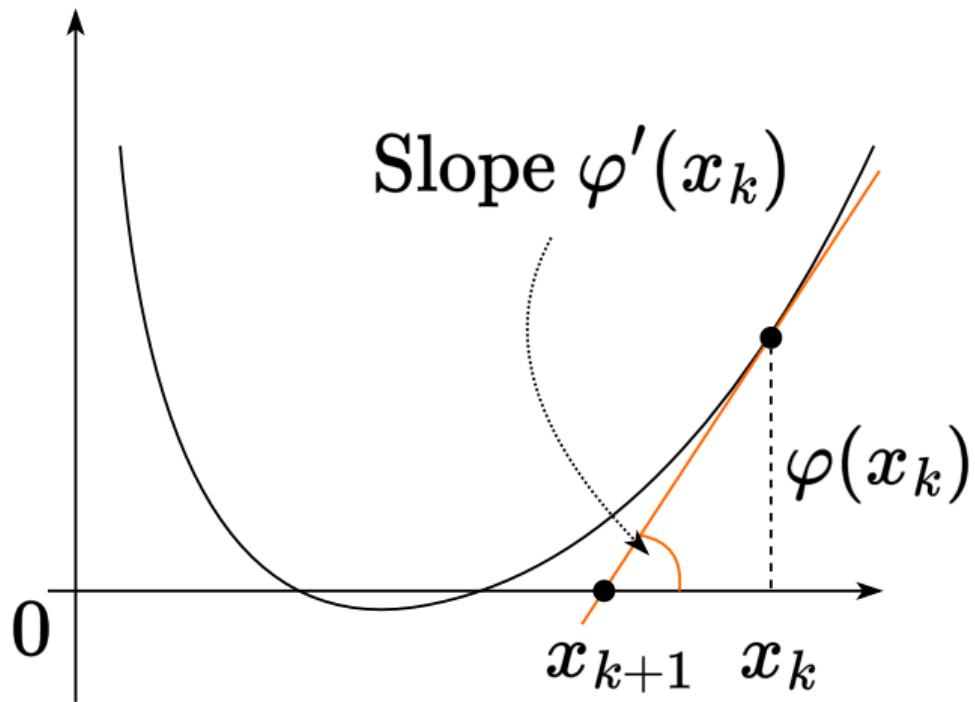
**Daniil Merkulov**

Optimization methods. MIPT

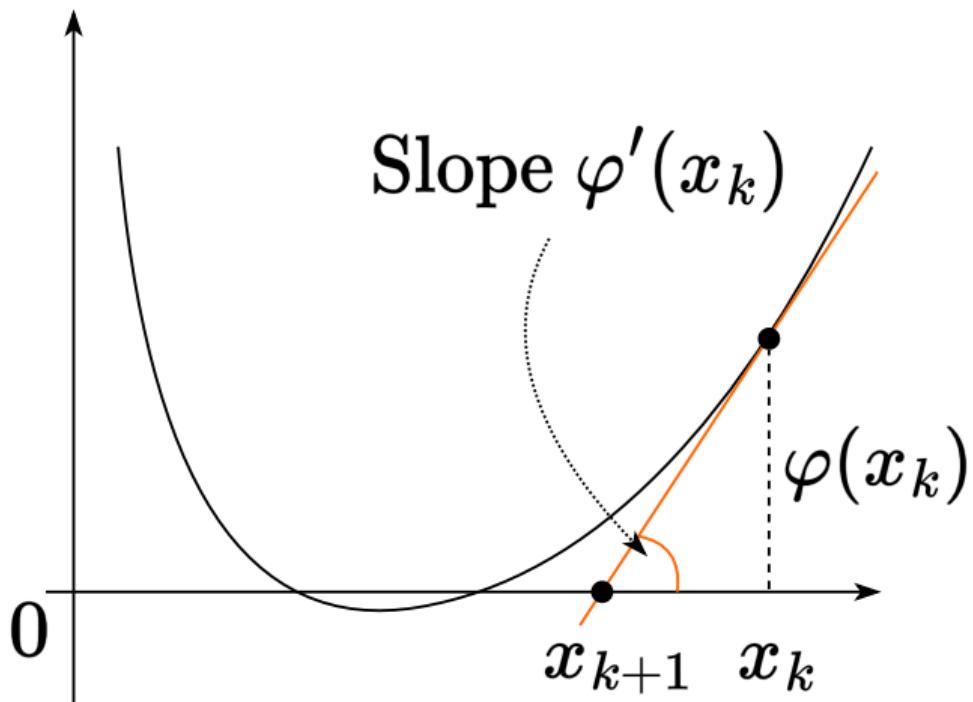
## Newton method

## Idea of Newton method of root finding

Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ .

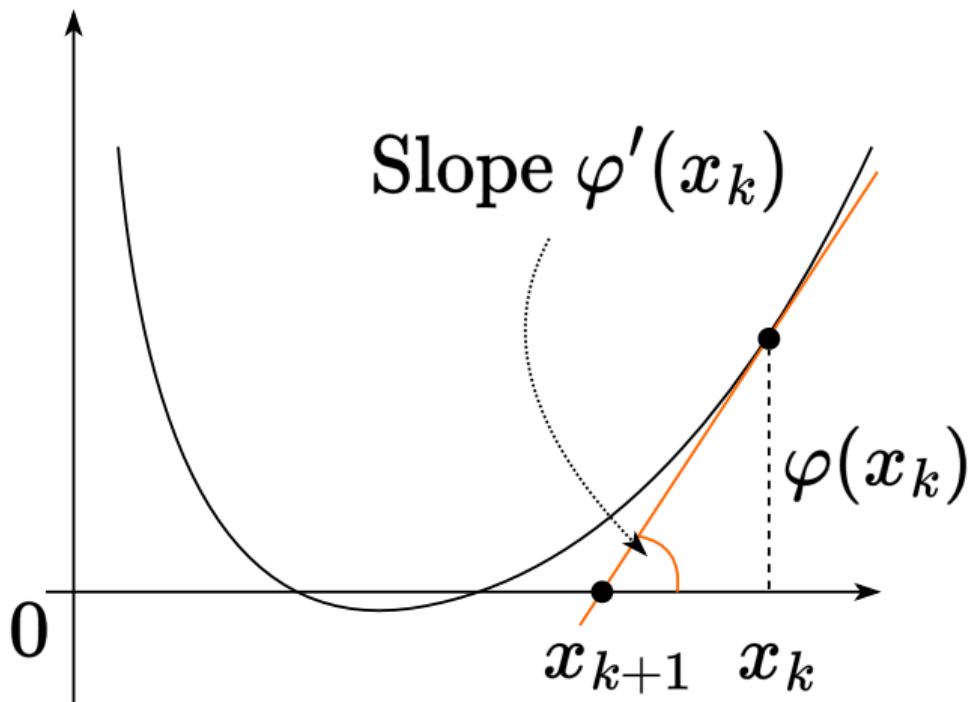


## Idea of Newton method of root finding



Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ .  
The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

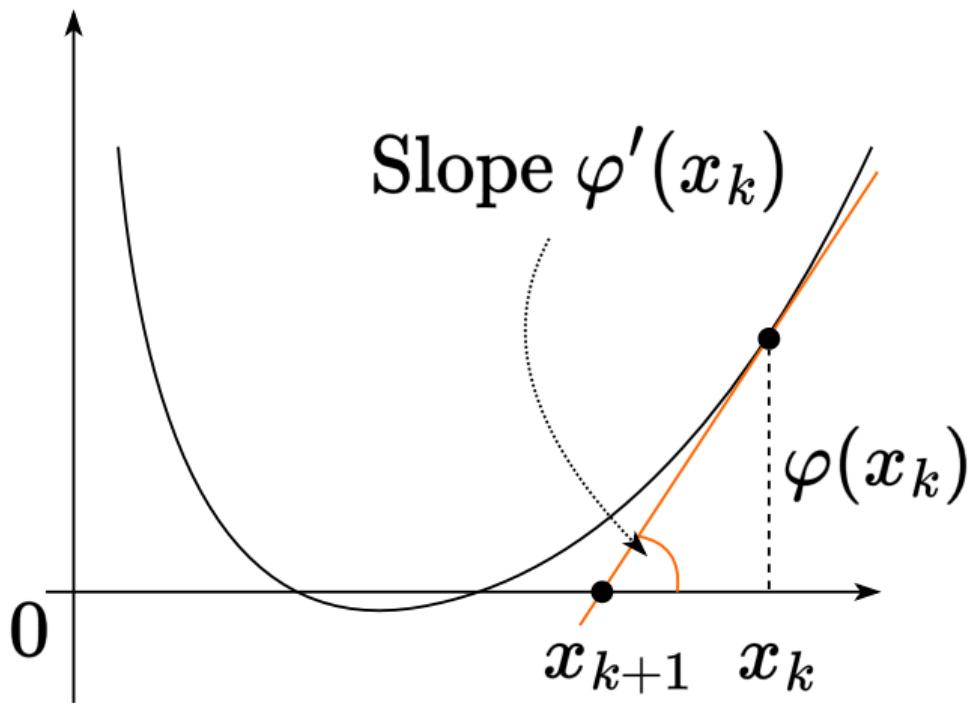
## Idea of Newton method of root finding



Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ . The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

## Idea of Newton method of root finding

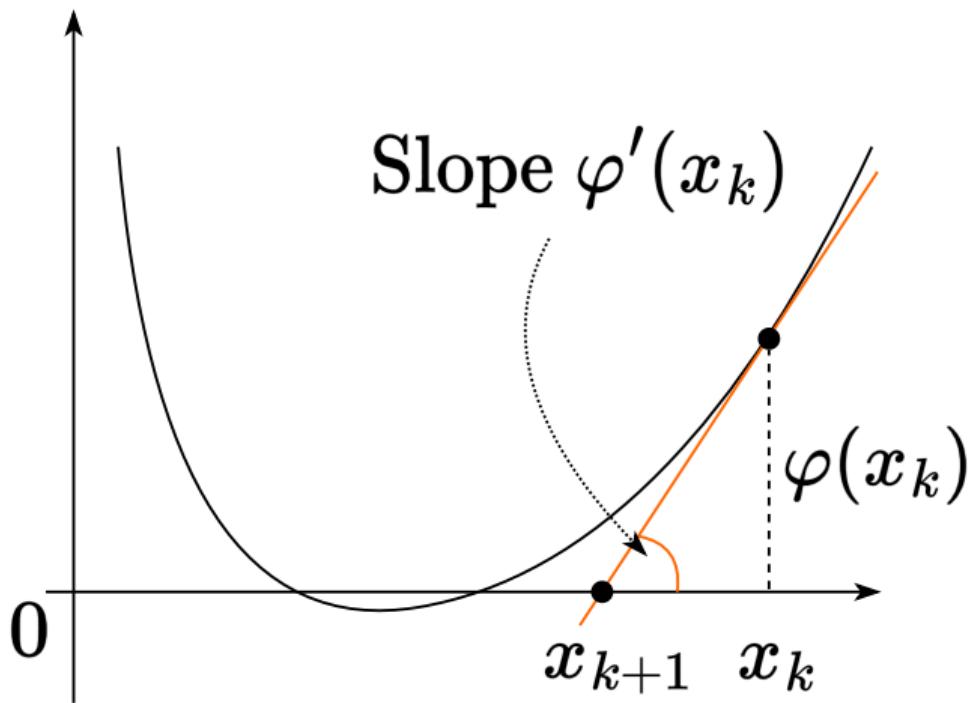


Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ .  
The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

## Idea of Newton method of root finding



Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ .  
The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

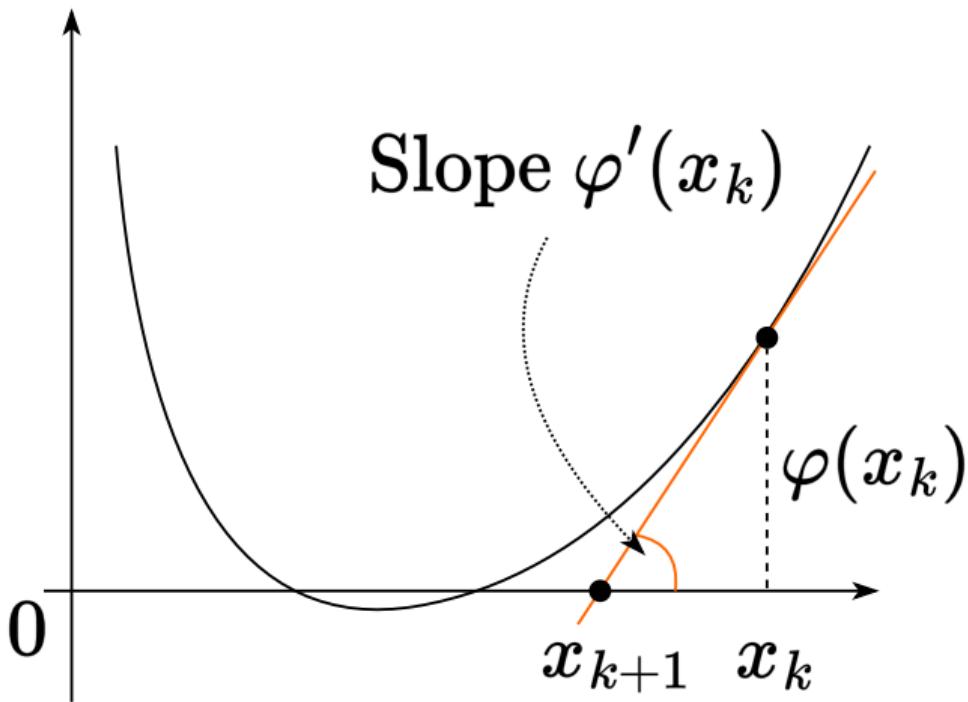
$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

<sup>1</sup>Literally we aim to solve the problem of finding stationary points  $\nabla f(x) = 0$

## Idea of Newton method of root finding



Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ .  
The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

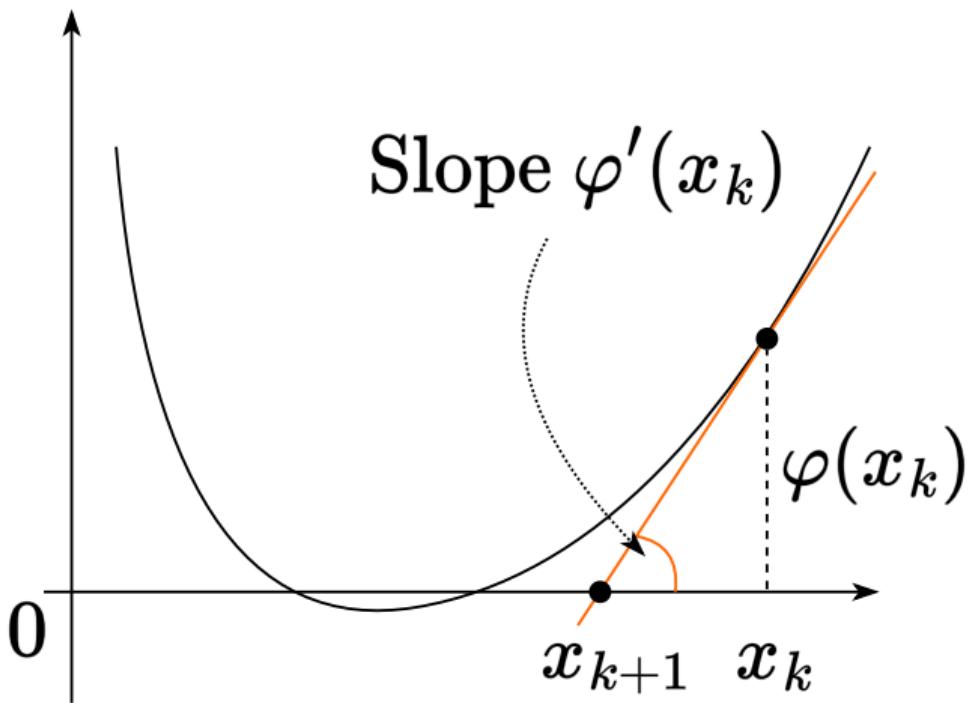
We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Which will become a Newton optimization method in case  $f'(x) = \varphi(x)$ <sup>1</sup>:

<sup>1</sup>Literally we aim to solve the problem of finding stationary points  $\nabla f(x) = 0$

## Idea of Newton method of root finding



Consider the function  $\varphi(x) : \mathbb{R} \rightarrow \mathbb{R}$ . The whole idea came from building a linear approximation at the point  $x_k$  and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Which will become a Newton optimization method in case  $f'(x) = \varphi(x)$ <sup>1</sup>:

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

<sup>1</sup>Literally we aim to solve the problem of finding stationary points  $\nabla f(x) = 0$

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

$$\nabla f_{x_k}^{II}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

$$\nabla f_{x_k}^{II}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

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

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

$$\nabla f_{x_k}^{II}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

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

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

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

$$\nabla f_{x_k}^{II}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

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

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

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

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

$$\nabla f_{x_k}^{II}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

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

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

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

## Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function  $f(x)$  and a certain point  $x_k$ . Let us consider the quadratic approximation of this function near  $x_k$ :

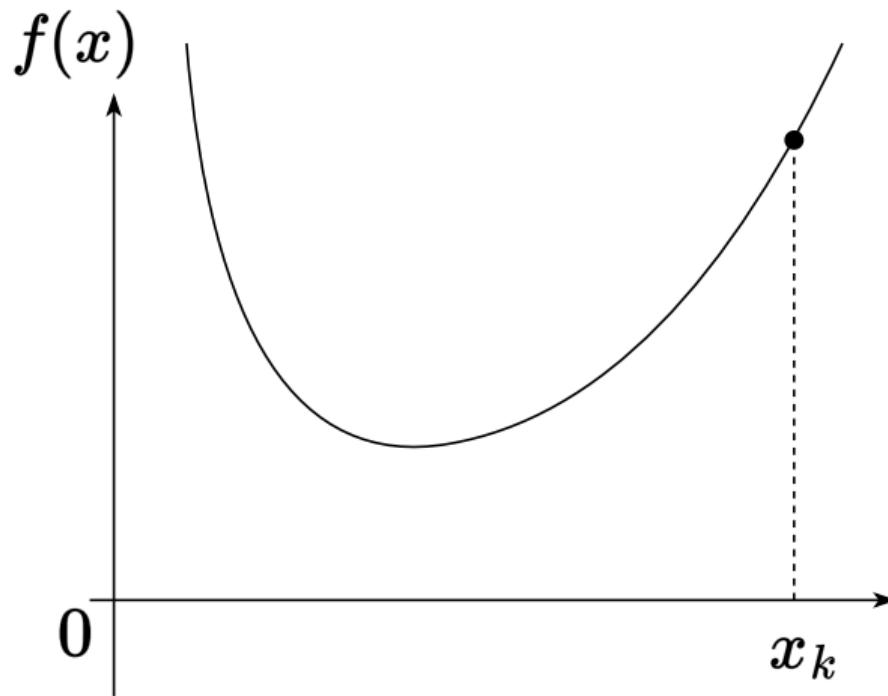
$$f_{x_k}^{II}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle.$$

The idea of the method is to find the point  $x_{k+1}$ , that minimizes the function  $f_{x_k}^{II}(x)$ , i.e.  $\nabla f_{x_k}^{II}(x_{k+1}) = 0$ .

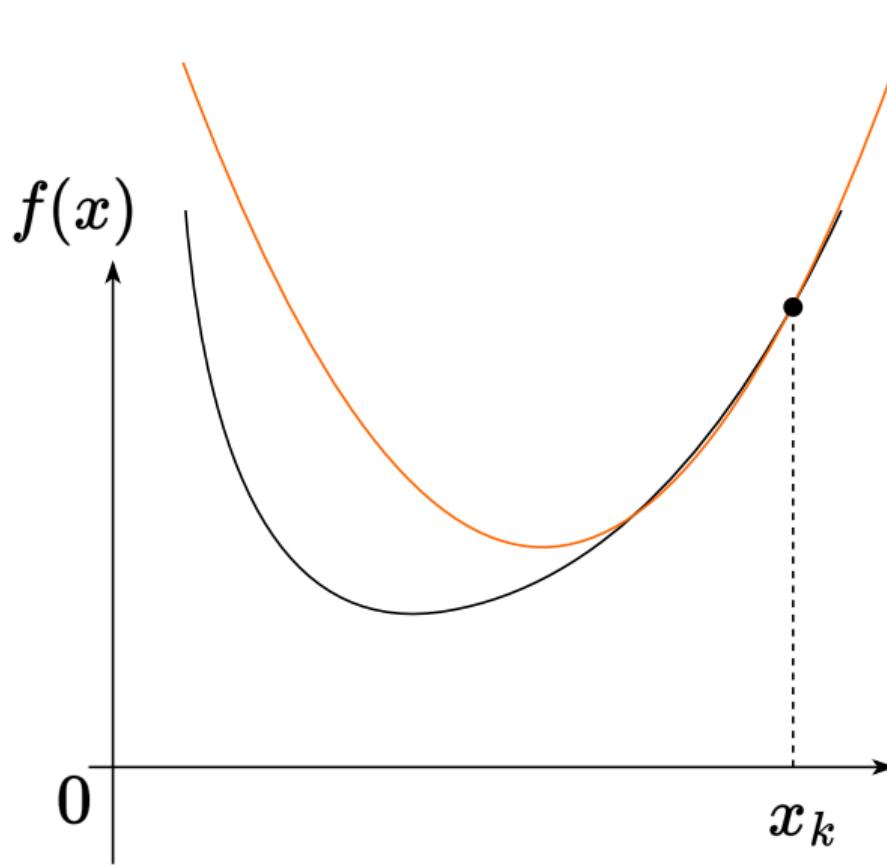
$$\begin{aligned}\nabla f_{x_k}^{II}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\ \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\ [\nabla^2 f(x_k)]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \\ x_{k+1} &= x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k).\end{aligned}$$

Let us immediately note the limitations related to the necessity of the Hessian's non-degeneracy (for the method to exist), as well as its positive definiteness (for the convergence guarantee).

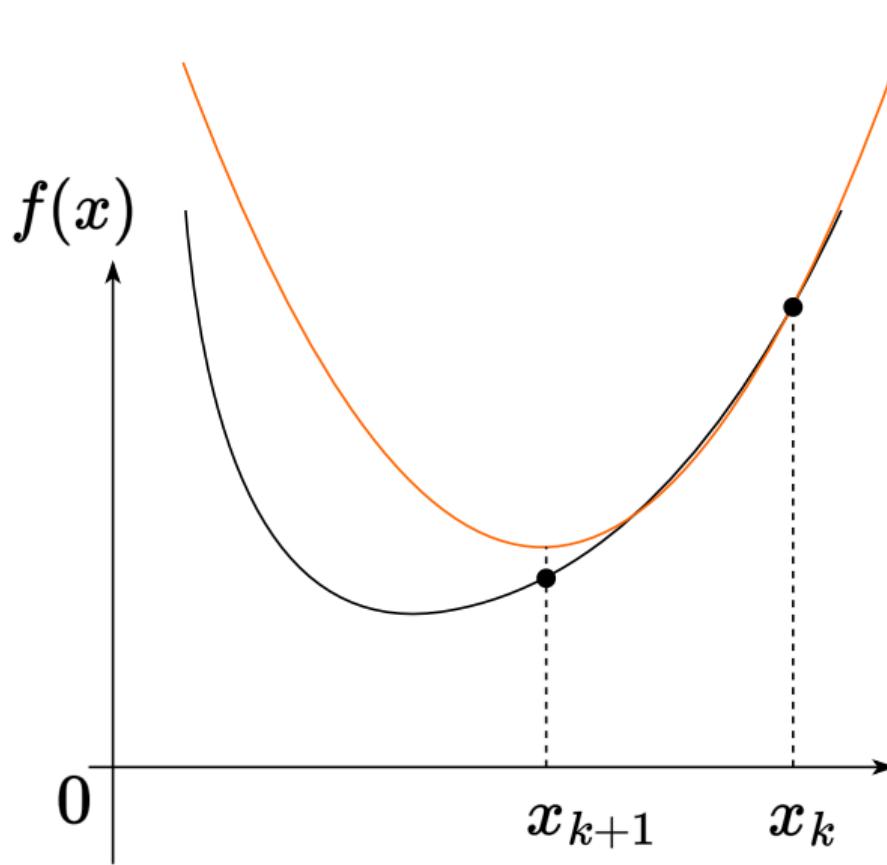
## Newton method as a local quadratic Taylor approximation minimizer



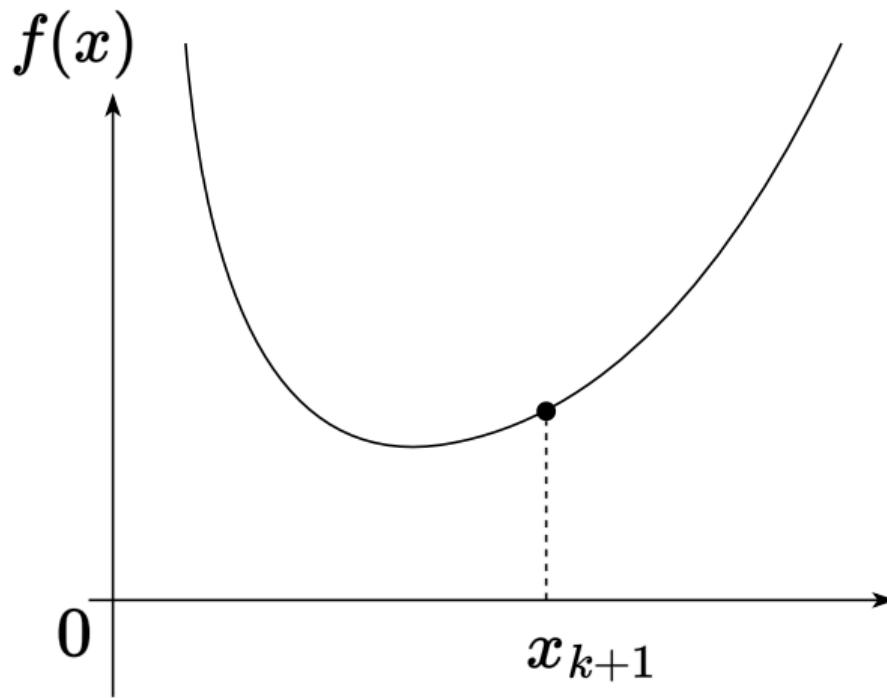
## Newton method as a local quadratic Taylor approximation minimizer



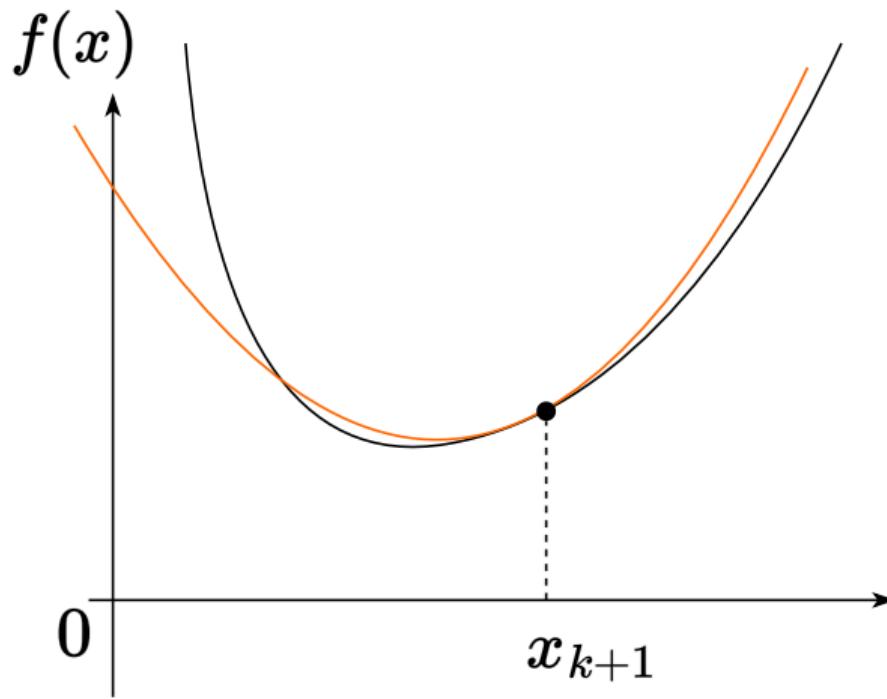
## Newton method as a local quadratic Taylor approximation minimizer



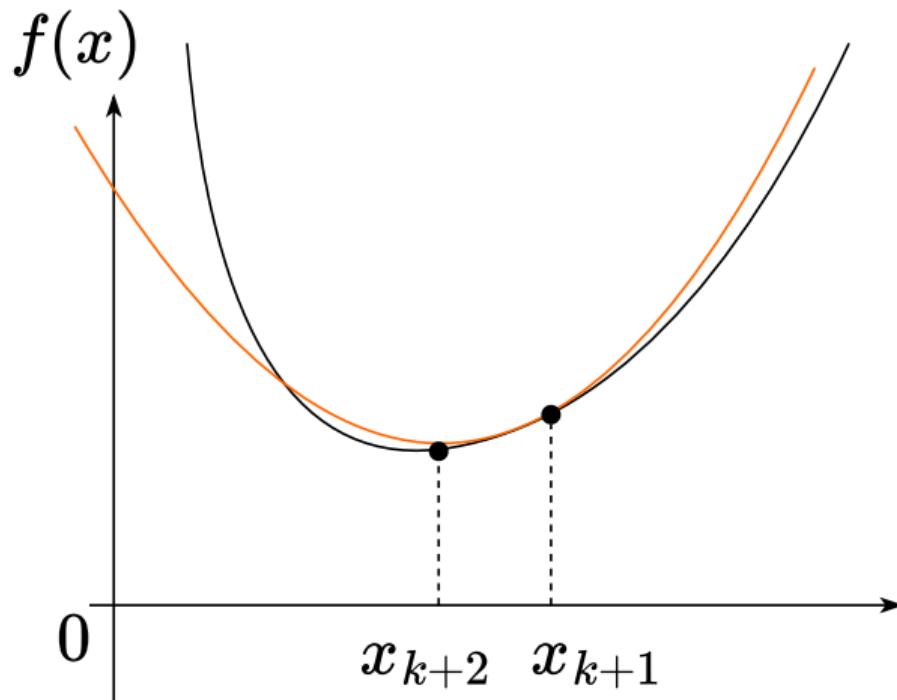
## Newton method as a local quadratic Taylor approximation minimizer



## Newton method as a local quadratic Taylor approximation minimizer



## Newton method as a local quadratic Taylor approximation minimizer



# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

## Proof

# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

## Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

## Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

2. Then we track the distance to the solution

# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

## Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

2. Then we track the distance to the solution

$$x_{k+1} - x^* = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) - x^* = x_k - x^* - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) =$$

# Convergence

## i Theorem

Let  $f(x)$  be a strongly convex twice continuously differentiable function at  $\mathbb{R}^n$ , for the second derivative of which inequalities are executed:  $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$ . Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is  $M$ -Lipschitz continuous, then this method converges locally to  $x^*$  at a quadratic rate.

## Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

2. Then we track the distance to the solution

$$\begin{aligned} x_{k+1} - x^* &= x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) - x^* = x_k - x^* - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) = \\ &= x_k - x^* - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau \end{aligned}$$

## Convergence

3.

$$= \left( I - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) =$$

## Convergence

3.

$$\begin{aligned} &= \left( I - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \end{aligned}$$

## Convergence

3.

$$\begin{aligned} &= \left( I - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))) d\tau \right) (x_k - x^*) = \end{aligned}$$

## Convergence

3.

$$\begin{aligned} &= \left( I - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))) d\tau \right) (x_k - x^*) = \\ &\quad = [\nabla^2 f(x_k)]^{-1} G_k(x_k - x^*) \end{aligned}$$

## Convergence

3.

$$\begin{aligned} &= \left( I - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) (x_k - x^*) = \\ &= [\nabla^2 f(x_k)]^{-1} \left( \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))) d\tau \right) (x_k - x^*) = \\ &\quad = [\nabla^2 f(x_k)]^{-1} G_k (x_k - x^*) \end{aligned}$$

4. We have introduced:

$$G_k = \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))) d\tau .$$

## Convergence

5. Let's try to estimate the size of  $G_k$ :

where  $r_k = \|x_k - x^*\|$ .

## Convergence

5. Let's try to estimate the size of  $G_k$ :

$$\|G_k\| = \left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right\| \leq$$

where  $r_k = \|x_k - x^*\|$ .

## Convergence

5. Let's try to estimate the size of  $G_k$ :

$$\begin{aligned}\|G_k\| &= \left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right\| \leq \\ &\leq \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))\| d\tau \leq \quad \text{(Hessian's Lipschitz continuity)}\end{aligned}$$

where  $r_k = \|x_k - x^*\|$ .

## Convergence

5. Let's try to estimate the size of  $G_k$ :

$$\begin{aligned}\|G_k\| &= \left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right\| \leq \\ &\leq \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))\| d\tau \leq \quad \text{(Hessian's Lipschitz continuity)} \\ &\leq \int_0^1 M \|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M \|x_k - x^*\| (1 - \tau) d\tau = \frac{r_k}{2} M,\end{aligned}$$

where  $r_k = \|x_k - x^*\|$ .

## Convergence

5. Let's try to estimate the size of  $G_k$ :

$$\begin{aligned}\|G_k\| &= \left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right\| \leq \\ &\leq \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))\| d\tau \leq \quad \text{(Hessian's Lipschitz continuity)} \\ &\leq \int_0^1 M \|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M \|x_k - x^*\| (1 - \tau) d\tau = \frac{r_k}{2} M,\end{aligned}$$

where  $r_k = \|x_k - x^*\|$ .

6. So, we have:

$$r_{k+1} \leq \left\| [\nabla^2 f(x_k)]^{-1} \right\| \cdot \frac{r_k}{2} M \cdot r_k$$

and we need to bound the norm of the inverse hessian

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

## Convergence

- Because of Hessian's Lipschitz continuity and symmetry:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -Mr_k I_n$$

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$\begin{aligned}\nabla^2 f(x_k) - \nabla^2 f(x^*) &\succeq -Mr_k I_n \\ \nabla^2 f(x_k) &\succeq \nabla^2 f(x^*) - Mr_k I_n\end{aligned}$$

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \mu I_n - Mr_k I_n$$

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \mu I_n - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq (\mu - Mr_k) I_n$$

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \mu I_n - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq (\mu - Mr_k) I_n$$

Convexity implies  $\nabla^2 f(x_k) \succ 0$ , i.e.  $r_k < \frac{\mu}{M}$ .

$$\left\| [\nabla^2 f(x_k)]^{-1} \right\| \leq (\mu - Mr_k)^{-1}$$

$$r_{k+1} \leq \frac{r_k^2 M}{2(\mu - Mr_k)}$$

## Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$\nabla^2 f(x_k) - \nabla^2 f(x^*) \succeq -Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq \mu I_n - Mr_k I_n$$

$$\nabla^2 f(x_k) \succeq (\mu - Mr_k) I_n$$

Convexity implies  $\nabla^2 f(x_k) \succ 0$ , i.e.  $r_k < \frac{\mu}{M}$ .

$$\left\| [\nabla^2 f(x_k)]^{-1} \right\| \leq (\mu - Mr_k)^{-1}$$

$$r_{k+1} \leq \frac{r_k^2 M}{2(\mu - Mr_k)}$$

8. The convergence condition  $r_{k+1} < r_k$  imposes additional conditions on  $r_k$ :  $r_k < \frac{2\mu}{3M}$

Thus, we have an important result: Newton's method for the function with Lipschitz positive-definite Hessian converges **quadratically** near ( $\|x_0 - x^*\| < \frac{2\mu}{3M}$ ) to the solution.

## Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function  $f$  and a nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , let  $x = Ay$ , and define  $g(y) = f(Ay)$ . Note, that  $\nabla g(y) = A^T \nabla f(x)$  and  $\nabla^2 g(y) = A^T \nabla^2 f(x)A$ . The Newton steps on  $g$  are expressed as:

$$y_{k+1} = y_k - (\nabla^2 g(y_k))^{-1} \nabla g(y_k)$$

## Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function  $f$  and a nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , let  $x = Ay$ , and define  $g(y) = f(Ay)$ . Note, that  $\nabla g(y) = A^T \nabla f(x)$  and  $\nabla^2 g(y) = A^T \nabla^2 f(x)A$ . The Newton steps on  $g$  are expressed as:

$$y_{k+1} = y_k - (\nabla^2 g(y_k))^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - (A^T \nabla^2 f(Ay_k) A)^{-1} A^T \nabla f(Ay_k)$$

## Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function  $f$  and a nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , let  $x = Ay$ , and define  $g(y) = f(Ay)$ . Note, that  $\nabla g(y) = A^T \nabla f(x)$  and  $\nabla^2 g(y) = A^T \nabla^2 f(x)A$ . The Newton steps on  $g$  are expressed as:

$$y_{k+1} = y_k - (\nabla^2 g(y_k))^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - (A^T \nabla^2 f(Ay_k) A)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse  $(AB)^{-1} = B^{-1}A^{-1}$ , this simplifies to:

$$\begin{aligned} y_{k+1} &= y_k - A^{-1} (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \\ Ay_{k+1} &= Ay_k - (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \end{aligned}$$

## Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function  $f$  and a nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , let  $x = Ay$ , and define  $g(y) = f(Ay)$ . Note, that  $\nabla g(y) = A^T \nabla f(x)$  and  $\nabla^2 g(y) = A^T \nabla^2 f(x)A$ . The Newton steps on  $g$  are expressed as:

$$y_{k+1} = y_k - (\nabla^2 g(y_k))^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - (A^T \nabla^2 f(Ay_k) A)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse  $(AB)^{-1} = B^{-1}A^{-1}$ , this simplifies to:

$$\begin{aligned} y_{k+1} &= y_k - A^{-1} (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \\ Ay_{k+1} &= Ay_k - (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \end{aligned}$$

Thus, the update rule for  $x$  is:

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

## Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function  $f$  and a nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , let  $x = Ay$ , and define  $g(y) = f(Ay)$ . Note, that  $\nabla g(y) = A^T \nabla f(x)$  and  $\nabla^2 g(y) = A^T \nabla^2 f(x)A$ . The Newton steps on  $g$  are expressed as:

$$y_{k+1} = y_k - (\nabla^2 g(y_k))^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - (A^T \nabla^2 f(Ay_k) A)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse  $(AB)^{-1} = B^{-1}A^{-1}$ , this simplifies to:

$$\begin{aligned} y_{k+1} &= y_k - A^{-1} (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \\ Ay_{k+1} &= Ay_k - (\nabla^2 f(Ay_k))^{-1} \nabla f(Ay_k) \end{aligned}$$

Thus, the update rule for  $x$  is:

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

This shows that the progress made by Newton's method is independent of problem scaling. This property is not shared by the gradient descent method!

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration:  $\mathcal{O}(n^2)$  memory

## Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration:  $\mathcal{O}(n^2)$  memory
- it is necessary to solve linear systems:  $\mathcal{O}(n^3)$  operations

# Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration:  $\mathcal{O}(n^2)$  memory
- it is necessary to solve linear systems:  $\mathcal{O}(n^3)$  operations
- the Hessian can be degenerate at  $x^*$

# Summary

What's nice:

- quadratic convergence near the solution  $x^*$
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration:  $\mathcal{O}(n^2)$  memory
- it is necessary to solve linear systems:  $\mathcal{O}(n^3)$  operations
- the Hessian can be degenerate at  $x^*$
- the hessian may not be positively determined → direction  $-(f''(x))^{-1}f'(x)$  may not be a descending direction

## Newton method problems

# Newton

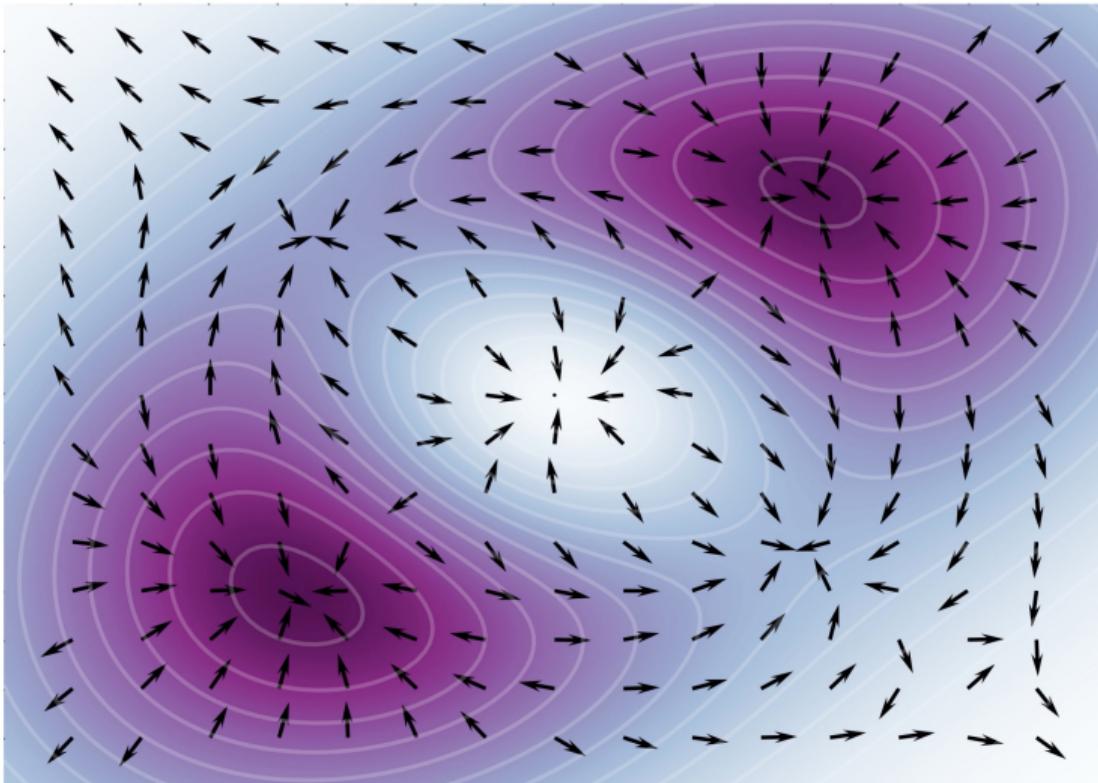


Figure 7: Animation

## Newton method problems

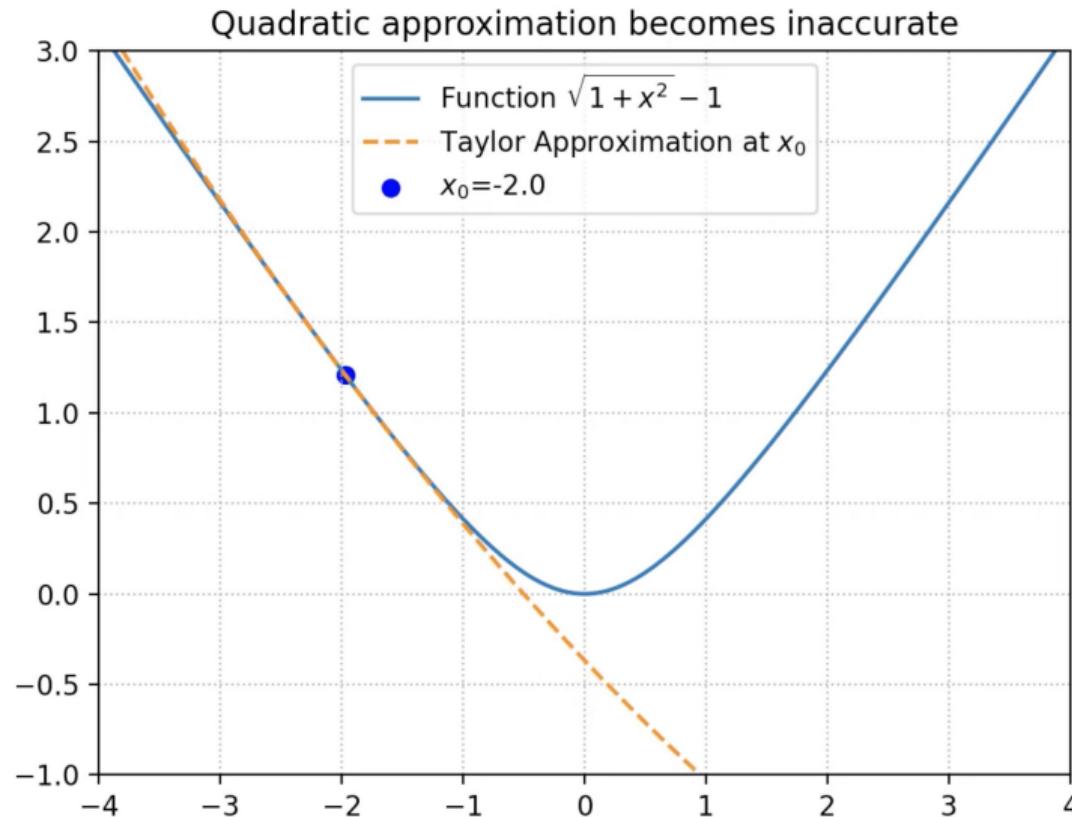


Figure 8: Animation

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define

$B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points  
with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a  
distance function  $d(x, x_0)$ .

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define

$B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points  
with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a  
distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define

$B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define

$B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define

$B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

Let us assume that the distance is defined locally by some metric  $A$ :

$$d(x, x_0) = (x - x_0)^\top A(x - x_0)$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define  $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

Let us assume that the distance is defined locally by some metric  $A$ :

$$d(x, x_0) = (x - x_0)^\top A(x - x_0)$$

Let us also consider first order Taylor approximation of a function  $f(x)$  near the point  $x_0$ :

$$f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \quad (1)$$

Now we can explicitly pose a problem of finding  $s$ , as it was stated above.

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} f(x_0 + \delta x) \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define  $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

Let us assume that the distance is defined locally by some metric  $A$ :

$$d(x, x_0) = (x - x_0)^\top A(x - x_0)$$

Let us also consider first order Taylor approximation of a function  $f(x)$  near the point  $x_0$ :

$$f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \quad (1)$$

Now we can explicitly pose a problem of finding  $s$ , as it was stated above.

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} f(x_0 + \delta x) \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

Using equation (1) it can be written as:

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} \nabla f(x_0)^\top \delta x \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define  $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

Let us assume that the distance is defined locally by some metric  $A$ :

$$d(x, x_0) = (x - x_0)^\top A(x - x_0)$$

Let us also consider first order Taylor approximation of a function  $f(x)$  near the point  $x_0$ :

$$f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \quad (1)$$

Now we can explicitly pose a problem of finding  $s$ , as it was stated above.

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} f(x_0 + \delta x) \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

Using equation (1) it can be written as:

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} \nabla f(x_0)^\top \delta x \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

Using Lagrange multipliers method, we can easily conclude, that the answer is:

$$\delta x = -\frac{2\varepsilon^2}{\nabla f(x_0)^\top A^{-1} \nabla f(x_0)} A^{-1} \nabla f$$

## The idea of adaptive metrics

Given  $f(x)$  and a point  $x_0$ . Define  $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$  as the set of points with distance  $\varepsilon$  to  $x_0$ . Here we presume the existence of a distance function  $d(x, x_0)$ .

$$x^* = \arg \min_{x \in B_\varepsilon(x_0)} f(x)$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$s = \lim_{\varepsilon \rightarrow 0} \frac{x^* - x_0}{\varepsilon}$$

Let us assume that the distance is defined locally by some metric  $A$ :

$$d(x, x_0) = (x - x_0)^\top A(x - x_0)$$

Let us also consider first order Taylor approximation of a function  $f(x)$  near the point  $x_0$ :

$$f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \quad (1)$$

Now we can explicitly pose a problem of finding  $s$ , as it was stated above.

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} f(x_0 + \delta x) \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

Using equation (1) it can be written as:

$$\begin{aligned} & \min_{\delta x \in \mathbb{R}^n} \nabla f(x_0)^\top \delta x \\ \text{s.t. } & \delta x^\top A \delta x = \varepsilon^2 \end{aligned}$$

Using Lagrange multipliers method, we can easily conclude, that the answer is:

$$\delta x = -\frac{2\varepsilon^2}{\nabla f(x_0)^\top A^{-1} \nabla f(x_0)} A^{-1} \nabla f$$

Which means, that new direction of steepest descent is nothing else, but  $A^{-1} \nabla f(x_0)$ .

Indeed, if the space is isotropic and  $A = I$ , we immediately have gradient descent formula, while Newton method uses local Hessian as a metric matrix.

## Quasi-Newton methods

## Quasi-Newton methods intuition

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  the general scheme of iteration method is written as:

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

## Quasi-Newton methods intuition

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  the general scheme of iteration method is written as:

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

In the Newton method, the  $d_k$  direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

## Quasi-Newton methods intuition

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  the general scheme of iteration method is written as:

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

In the Newton method, the  $d_k$  direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

## Quasi-Newton methods intuition

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  the general scheme of iteration method is written as:

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

In the Newton method, the  $d_k$  direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

Note here that if we take a single matrix of  $B_k = I_n$  as  $B_k$  at each step, we will exactly get the gradient descent method.

The general scheme of quasi-Newton methods is based on the selection of the  $B_k$  matrix so that it tends in some sense at  $k \rightarrow \infty$  to the truth value of the Hessian  $\nabla^2 f(x_k)$ .

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

**Basic Idea:** As  $B_k$  already contains information about the Hessian, use a suitable matrix update to form  $B_{k+1}$ .

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

**Basic Idea:** As  $B_k$  already contains information about the Hessian, use a suitable matrix update to form  $B_{k+1}$ .

**Reasonable Requirement for  $B_{k+1}$**  (motivated by the secant method):

$$\begin{aligned}\nabla f(x_{k+1}) - \nabla f(x_k) &= B_{k+1}(x_{k+1} - x_k) = B_{k+1}d_k \\ \Delta y_k &= B_{k+1}\Delta x_k\end{aligned}$$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

**Basic Idea:** As  $B_k$  already contains information about the Hessian, use a suitable matrix update to form  $B_{k+1}$ .

**Reasonable Requirement for  $B_{k+1}$**  (motivated by the secant method):

$$\begin{aligned}\nabla f(x_{k+1}) - \nabla f(x_k) &= B_{k+1}(x_{k+1} - x_k) = B_{k+1}d_k \\ \Delta y_k &= B_{k+1}\Delta x_k\end{aligned}$$

In addition to the secant equation, we want:

- $B_{k+1}$  to be symmetric

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

**Basic Idea:** As  $B_k$  already contains information about the Hessian, use a suitable matrix update to form  $B_{k+1}$ .

**Reasonable Requirement for  $B_{k+1}$**  (motivated by the secant method):

$$\begin{aligned}\nabla f(x_{k+1}) - \nabla f(x_k) &= B_{k+1}(x_{k+1} - x_k) = B_{k+1}d_k \\ \Delta y_k &= B_{k+1}\Delta x_k\end{aligned}$$

In addition to the secant equation, we want:

- $B_{k+1}$  to be symmetric
- $B_{k+1}$  to be “close” to  $B_k$

## Quasi-Newton Method Template

Let  $x_0 \in \mathbb{R}^n$ ,  $B_0 \succ 0$ . For  $k = 1, 2, 3, \dots$ , repeat:

1. Solve  $B_k d_k = -\nabla f(x_k)$
2. Update  $x_{k+1} = x_k + \alpha_k d_k$
3. Compute  $B_{k+1}$  from  $B_k$

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute  $(B_{k+1})^{-1}$  from  $(B_k)^{-1}$ .

**Basic Idea:** As  $B_k$  already contains information about the Hessian, use a suitable matrix update to form  $B_{k+1}$ .

**Reasonable Requirement for  $B_{k+1}$**  (motivated by the secant method):

$$\begin{aligned}\nabla f(x_{k+1}) - \nabla f(x_k) &= B_{k+1}(x_{k+1} - x_k) = B_{k+1}d_k \\ \Delta y_k &= B_{k+1}\Delta x_k\end{aligned}$$

In addition to the secant equation, we want:

- $B_{k+1}$  to be symmetric
- $B_{k+1}$  to be “close” to  $B_k$
- $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$

## Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

## Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation  $B_{k+1}d_k = \Delta y_k$  yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

## Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation  $B_{k+1}d_k = \Delta y_k$  yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

This only holds if  $u$  is a multiple of  $\Delta y_k - B_k d_k$ . Putting  $u = \Delta y_k - B_k d_k$ , we solve the above,

$$a = \frac{1}{(\Delta y_k - B_k d_k)^T d_k},$$

## Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation  $B_{k+1}d_k = \Delta y_k$  yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

This only holds if  $u$  is a multiple of  $\Delta y_k - B_k d_k$ . Putting  $u = \Delta y_k - B_k d_k$ , we solve the above,

$$a = \frac{1}{(\Delta y_k - B_k d_k)^T d_k},$$

which leads to

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

called the symmetric rank-one (SR1) update or Broyden method.

## Symmetric Rank-One Update with inverse

How can we solve

$$B_{k+1}d_{k+1} = -\nabla f(x_{k+1}),$$

in order to take the next step? In addition to propagating  $B_k$  to  $B_{k+1}$ , let's propagate inverses, i.e.,  $C_k = B_k^{-1}$  to  $C_{k+1} = (B_{k+1})^{-1}$ .

### Sherman-Morrison Formula:

The Sherman-Morrison formula states:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

Thus, for the SR1 update, the inverse is also easily updated:

$$C_{k+1} = C_k + \frac{(d_k - C_k \Delta y_k)(d_k - C_k \Delta y_k)^T}{(d_k - C_k \Delta y_k)^T \Delta y_k}$$

In general, SR1 is simple and cheap, but it has a key shortcoming: it does not preserve positive definiteness.

## Davidon-Fletcher-Powell Update

We could have pursued the same idea to update the inverse  $C$ :

$$C_{k+1} = C_k + auu^T + bvv^T.$$

## Davidon-Fletcher-Powell Update

We could have pursued the same idea to update the inverse  $C$ :

$$C_{k+1} = C_k + auu^T + bvv^T.$$

Multiplying by  $\Delta y_k$ , using the secant equation  $d_k = C_k \Delta y_k$ , and solving for  $a, b$ , yields:

$$C_{k+1} = C_k - \frac{C_k \Delta y_k \Delta y_k^T C_k}{\Delta y_k^T C_k \Delta y_k} + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

## Woodbury Formula Application

Woodbury then shows:

$$B_{k+1} = \left( I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k} \right) B_k \left( I - \frac{d_k \Delta y_k^T}{\Delta y_k^T d_k} \right) + \frac{\Delta y_k \Delta y_k^T}{\Delta y_k^T d_k}$$

This is the Davidon-Fletcher-Powell (DFP) update. Also cheap:  $O(n^2)$ , preserves positive definiteness. Not as popular as BFGS.

## Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

## Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

The secant equation  $\Delta y_k = B_{k+1}d_k$  yields:

$$\Delta y_k - B_k d_k = (au^T d_k)u + (bv^T d_k)v$$

## Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

The secant equation  $\Delta y_k = B_{k+1}d_k$  yields:

$$\Delta y_k - B_k d_k = (au^T d_k)u + (bv^T d_k)v$$

Putting  $u = \Delta y_k$ ,  $v = B_k d_k$ , and solving for a, b we get:

$$B_{k+1} = B_k - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k} + \frac{\Delta y_k \Delta y_k^T}{d_k^T \Delta y_k}$$

called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

## Broyden-Fletcher-Goldfarb-Shanno update with inverse

### Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

## Broyden-Fletcher-Goldfarb-Shanno update with inverse

### Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Applied to our case, we get a rank-two update on the inverse  $C$ :

$$C_{k+1} = C_k + \frac{(d_k - C_k \Delta y_k) d_k^T}{\Delta y_k^T d_k} + \frac{d_k (d_k - C_k \Delta y_k)^T}{\Delta y_k^T d_k} - \frac{(d_k - C_k \Delta y_k)^T \Delta y_k}{(\Delta y_k^T d_k)^2} d_k d_k^T$$

$$C_{k+1} = \left( I - \frac{d_k \Delta y_k^T}{\Delta y_k^T d_k} \right) C_k \left( I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k} \right) + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

This formulation ensures that the BFGS update, while comprehensive, remains computationally efficient, requiring  $O(n^2)$  operations. Importantly, BFGS update preserves positive definiteness. Recall this means  $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$ . Equivalently,  $C_k \succ 0 \Rightarrow C_{k+1} \succ 0$

## Code

- Open In Colab

## Code

- Open In Colab
- Comparison of quasi Newton methods

## Code

- Open In Colab
- Comparison of quasi Newton methods
- Some practical notes about Newton method