Non-Linear Optimization Methods - Part II

Optimization Methods in Management Science
Master in Management
HEC Lausanne

Dr. Rodrigue Oeuvray

Fall 2019 Semester

Non-Linear Optimization Methods

Unconstrained optimization:

- Descent methods
- Linesearch
- Newton's method

Context

- We now consider the more general problem described below:

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

  where f is typically twice continuously differentiable

- We don't assume any more that $f(\boldsymbol{x})$ is a quadratic function
- It can be **any** twice continuously differentiable function
- A descent method is an **iterative** optimization algorithm for finding a **local optimum** of such a function

Descent Direction

- A direction $\boldsymbol{d}_k$ for which $\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k < 0$ is called a descent direction
- If $\boldsymbol{d}_k$ is a **descent** direction, then we have the guarantee that

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) < f(\boldsymbol{x}_k)$$

  if $\alpha > 0$ is sufficiently small
- This just means that we can **decrease** the objective function if we move into the direction $\boldsymbol{d}_k$
- $\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k < 0$ can be interpreted as the **directional derivative** of f at $\boldsymbol{x}_k$ along a vector $\boldsymbol{d}_k$

Descent Direction (2)

Proposition

We assume that $\boldsymbol{Q}$ is a positive definite matrix and that $\nabla f(\boldsymbol{x}_k) \neq \mathbf{0}$. Then any direction $\boldsymbol{d}_k$ given by $\boldsymbol{d}_k = -\boldsymbol{Q}\nabla f(\boldsymbol{x}_k)$ is a descent direction.

Proof : $-\nabla f(\boldsymbol{x}_k)^T \boldsymbol{Q} \nabla f(\boldsymbol{x}_k) < 0.$

Steepest Descent Direction

- The steepest descent is the direction given by the **opposite** of the gradient $-\nabla f(\boldsymbol{x})$
- This is a descent direction that is optimal in the sense that

$$-\nabla f(\boldsymbol{x})^T \nabla f(\boldsymbol{x}) \leq \boldsymbol{d}^T \nabla f(\boldsymbol{x})$$

for any $\boldsymbol{d}$ such that $\|\boldsymbol{d}\| = \|\nabla f(\boldsymbol{x})\|$

Descent Method Framework

Descent methods comprise the following steps:

- Find a direction $\boldsymbol{d}_k$ such that $\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k < 0$
- Find a **step** α_k such that $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k)$
- Compute $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $k = k + 1$, and repeat the process until a **stopping criterion** is satisfied

Example

We would like to **minimize** the following function f:

$$f(x_1, x_2) = \frac{1}{2}(x_1^2 + 10x_2^2)$$

The **solution** is the point $(0, 0)$. The gradient of f is given by:

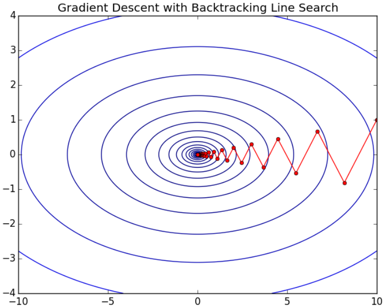
$$\nabla f(x_1, x_2) = (x_1 \ 10x_2)^T$$

We would like to apply the **steepest descent** method with the initial point given by $\boldsymbol{x} = (10, 1)$. At this point, the descent direction is

$$\boldsymbol{d}_1 = -\nabla f(10, 1) = (\text{-}10 \ \text{-}10)^T$$

Example (Cont'd)

Here is the sequence of iterates produced by the steepest descent method:



We can see that this sequence of iterates **converge** to the **optimal** solution by **zig-zaging**. We will see later in this presentation much performant algorithms than the steepest descent method

Length of the Step

- Indeed, it is not necessary to solve the following problem:

$$\alpha_k = argmin_{\alpha \geq 0} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$$

- We just need to find a step α_k that reduces f **sufficiently** in order to have an algorithm that **converges** to the **optimal** solution
- This step should be **easy** to compute
- The challenges in finding a good α_k are both in avoiding that the step length is **too long**, or **too short**
- If the step is too long or too short, then this can **prevent** the convergence of the descent method to the **optimal** solution

Example

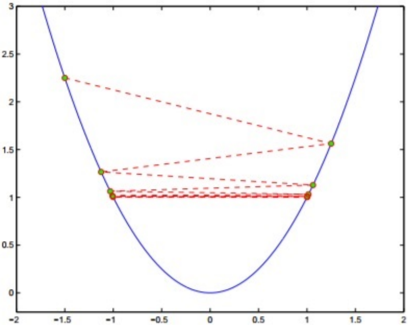
- We would like to **minimize** $f(x) = x^2$ with a descent method
- The **minimum** is at $x = 0$ and its value is 0
- The sequence of iterates is generated by the following formula:

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

- We would like that this sequence of iterates **converge** to $x = 0$
- With a problem with one variable, there is only **two possible directions** d_k: $d_k =$ -1 (we move to the left from the current iterate) and $d_k = 1$ (we move to the right)
- Let's assume that the **initial point** is given by $x_0 = 2$

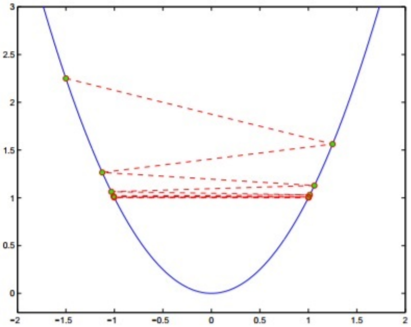
Example: Steps Too Long...

- Let's consider the case where $d_k = (\text{-}1)^{k+1}$ and $\alpha_k = 2 + 3(2^{\text{-}k-1})$
- $k = 0, d_0 = \text{-}1, \alpha_0 = 3.5, k = 1, d_1 = 1, \alpha_1 = 2.75, k = 2, d_2 = \text{-}1, \alpha_2 = 2 + 3/8, \ldots$
- Starting from $x_0 = 2$, the descent method generates the following sequence of iterates: $x_0 = 2, x_1 = \text{-}1.5, x_2 = 1.25, \ldots$



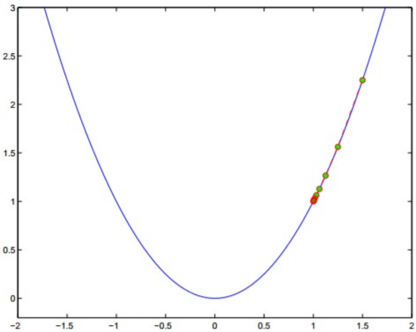
Example: Steps Too Long... (Cont'd)

- **The objective function decreases at each iteration but $\{x_k\}$ never converge to 0 !**
- When k is even (resp. odd), the sequence of iterates converge to 1 (resp. to -1). But the **optimum** is at $x = 0$!



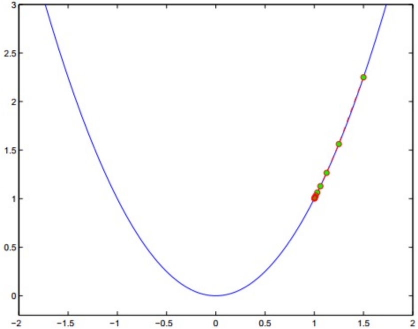
Example: Steps Too Short...

- Same example as before but we consider the case where $d_k = \text{-}1$ and $\alpha_k = 2^{\text{-}k-1}$
- $k = 0, d_0 = \text{-}1, \alpha_0 = 0.5, k = 1, d_1 = \text{-}1, \alpha_0 = 0.25, k = 2, d_2 = \text{-}1, \alpha_0 = 0.125$
- The descent method generates the following sequence of iterates: $x_0 = 2, x_1 = 1.5, x_2 = 1.25, \ldots$



Example: Steps Too Short...(Cont'd)

- **The objective function decreases at each iteration but $\{x_k\}$ converge to 1 and not to 0 !**



- **These examples just show that the steps α_k must satisfy some conditions to guarantee that the algorithm converges to an optimal solution**

Wolfe Conditions

The conditions that f must satisfy in terms of decrease to converge to the optimum are called the Wolfe conditions

Theorem

*Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, a point $\boldsymbol{x} \in \mathbb{R}^n$, a descent direction $\boldsymbol{d}_k$ such that $\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k < 0$ and a step $\alpha_k > 0$. We say that the step length α_k satisfies the **Wolfe conditions** if*

$$(1) \quad f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x_k}) + \alpha_k \beta_1 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \quad \textit{(Armijo rule)}$$

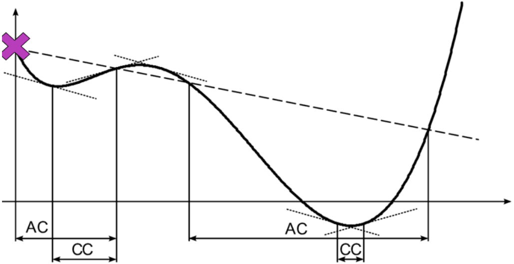
$$(2) \quad \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)^T \boldsymbol{d}_k \geq \beta_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \quad \textit{(curvature condition)}$$

with $0 < \beta_1 < \beta_2 < 1$

Armijo rule ensures that the step length α_k decreases f sufficiently, and the curvature condition ensures that the slope has been reduced sufficiently

Illustration of the Wolfe Conditions

- Let's define the function ϕ as follows: $\phi(\alpha) = f(\boldsymbol{x_k} + \alpha \boldsymbol{d_k})$



- $\phi(0) = f(\boldsymbol{x_k})$ corresponds to the violet cross
- The dotted line starting from the violet cross is given by $g(\alpha) = f(\boldsymbol{x_k}) + \alpha\beta_1 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k$
- AC represents the set of α satisfying the Armijo condition
- CC corresponds to set of α satisfying the curvature condition
- The set of α satisfying both conditions are given by the intersection between AC and CC

Wolfe Conditions: Exercise

We consider the following function:

$$\begin{aligned} f \ : \ & \mathbb{R}^2 \to \mathbb{R} \\ & (x, y) \mapsto f(x, y) = x^4 + x^2 + y^2 \end{aligned}$$

and the initial data:

$$\boldsymbol{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \beta_1 = 0.1 \quad \text{and} \quad \beta_2 = 0.5$$

a) Compute $\nabla f(\boldsymbol{x}_1)$

b) Show that $\boldsymbol{d}_1 = \begin{pmatrix} \text{-3} \\ \text{-1} \end{pmatrix}$ is a descent direction at $\boldsymbol{x}_1$

c) Determine if the steps $\alpha_1 = 1$, $\alpha_2 = 0.1$ and $\alpha_3 = 0.5$ satisfy Wolfe's conditions

Wolfe Conditions: Exercise (Cont'd)

a) $\nabla f(\boldsymbol{x}) = \begin{pmatrix} 4x^3 + 2x \\ 2y \end{pmatrix} \quad \forall\, \boldsymbol{x} \in \mathbb{R}^2, \quad \nabla f(1,1) = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$

b) $\boldsymbol{d}_1$ is a descent direction:

$$\nabla f(\boldsymbol{x}_1)^T \boldsymbol{d}_1 = \begin{pmatrix} 6 & 2 \end{pmatrix} \begin{pmatrix} -3 \\ -1 \end{pmatrix} = -20 < 0$$

Wolfe Conditions: Exercise (Cont'd)

c) $f(x,y) = x^4 + x^2 + y^2, \nabla f(\boldsymbol{x}) = (4x^3 + 2x, 2y)^T$. For $\alpha_1 = 1$, let's check Wolfe conditions with $\beta_1 = 0.1$ and $\beta_2 = 0.5$

$$\begin{array}{lll} (1) & f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \alpha_k \beta_1 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k & \text{(Armijo rule)} \\ (2) & \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)^T \boldsymbol{d}_k \geq \beta_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k & \text{(curvature condition)} \end{array}$$

Application : $\boldsymbol{x}_1 = (1,1), \nabla f(1,1) = (6,2)^T$, $\boldsymbol{d}_1 = (\text{-}3, \text{-}1)^T$, $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \alpha \boldsymbol{d}_1 = (\text{-}2, 0)$ and $\nabla f(\boldsymbol{x}_2) = (\text{-}36, 0)^T$

- It satisfies the curvature condition but not the Armijo rule:

(1) $f(\text{-}2,0) = 20 \nleq 1 = 3 + 1 \cdot 0.1 \cdot \begin{pmatrix} 6 & 2 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix}$

(2) $\begin{pmatrix} \text{-}36 & 0 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix} = 108 \geq \text{-}10 = 0.5 \cdot \begin{pmatrix} 6 & 2 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix}$

Wolfe Conditions: Exercise (Cont'd)

c) $f(x,y) = x^4 + x^2 + y^2, \nabla f(\boldsymbol{x}) = (4x^3 + 2x, 2y)^T$. For $\alpha_2 = 0.1$, let's check Wolfe conditions with $\beta_1 = 0.1$ and $\beta_2 = 0.5$

$$
\begin{array}{lll}
(1) & f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \alpha_k \beta_1 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k & \text{(Armijo rule)} \\
(2) & \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)^T \boldsymbol{d}_k \geq \beta_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k & \text{(curvature condition)}
\end{array}
$$

Application : $\boldsymbol{x}_1 = (1,1), \nabla f(1,1) = (6,2)^T$, $\boldsymbol{d}_1 = (\text{-}3, \text{-}1)^T$, $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \alpha \boldsymbol{d}_1 = (0.7, 0.9)$ and $\nabla f(\boldsymbol{x}_2) = (\text{-}2.772, 1.8)^T$

- It satisfies Armijo rule but not the curvature condition:

(1) $f(0.7, 0.9) = 1.54 \leq 2.8 = 3 + 0.1 \cdot 0.1 \cdot (\text{-}20)$

(2) $\begin{pmatrix} 2.772 & 1.8 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix} = \text{-}10.116 \ngeq \text{-}10 = 0.5 \cdot \begin{pmatrix} 6 & 2 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix}$

Wolfe Conditions: Exercise (Cont'd)

c) $f(x,y) = x^4 + x^2 + y^2, \nabla f(\boldsymbol{x}) = (4x^3 + 2x, 2y)^T$. For $\alpha_3 = 0.5$, let's check Wolfe conditions with $\beta_1 = 0.1$ and $\beta_2 = 0.5$

$$
\begin{aligned}
&(1) \quad f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + \alpha_k \beta_1 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \quad \text{(Armijo rule)} \\
&(2) \quad \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)^T \boldsymbol{d}_k \geq \beta_2 \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \quad \text{(curvature condition)}
\end{aligned}
$$

Application : $\boldsymbol{x}_1 = (1,1), \nabla f(1,1) = (6,2)^T$, $\boldsymbol{d}_1 = (\text{-}3, \text{-}1)^T$, $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \alpha \boldsymbol{d}_1 = (\text{-}0.5, 0.5)$ and $\nabla f(\boldsymbol{x}_2) = (\text{-}1.5, 1)^T$

- It satisfies both conditions

(1) $f(\text{-}0.5, 0.5) = 0.5625 \leq 2 = 3 + 0.5 \cdot 0.1 \cdot (\text{-}20)$

(2) $\begin{pmatrix} \text{-}1.5 & 1 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix} = 3.5 \geq \text{-}10 = 0.5 \cdot \begin{pmatrix} 6 & 2 \end{pmatrix} \begin{pmatrix} \text{-}3 \\ \text{-}1 \end{pmatrix}$

Linesearch

- A linesearch is a method that produces a step α^* satisfying the Wolfe conditions **at each iteration**
- Let's assume that we are at **iteration** k. How to generate a step α_k satisfying these conditions ?
  - We start from a initial step α_0
  - If this step violates the Armijo rule, then the step is too long and we reduce it
  - If this step violates the curvature rule, then the step is too short and we increase it

Backtracking Linesearch

- In practice, we use a backtracking linesearch
- The **backtracking linesearch** starts with a **large** step α_0 and iteratively shrinks it to satisfy the Armijo rule if it is necessary
- The curvature condition can be **dispensed**
- This makes this algorithm very **efficient** to find a step satisfying the Wolfe conditions

Newton's Method

We consider the following problem:

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

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is twice continuously differentiable

Newton's method:

- We **approximate** f by a **quadratic** function given by its **Taylor series expansion**
- The **minimization** of this quadratic function generates the **next** iterate
- We repeat these steps until a stopping criterion is reached

Newton's Method (Cont'd)

Important Remark

The **quadratic** approximation $q(\boldsymbol{x})$ of $f(\boldsymbol{x})$ around the current iterate $\boldsymbol{x}_k$ is given by its Taylor series expansion:

$$f(\boldsymbol{x}) \approx q(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

The first order condition to minimize the quadratic function is given by $\nabla q(\boldsymbol{x}) = 0$, i.e.:

$$\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) = 0$$

Newton's Method (Cont'd)

- If the matrix $\nabla^2 f(\boldsymbol{x}_k)$ is invertible, then the solution $\boldsymbol{x}$ of this equation is given by

$$\boldsymbol{x} = \boldsymbol{x}_k - \nabla^2 f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)$$

- The next iterate $\boldsymbol{x}_{k+1}$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x} = \boldsymbol{x}_k - \nabla^2 f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)$$

- We repeat these steps until a **stopping criterion** is reached

Newton's Method: Algorithm

- **Input**: a given point $\boldsymbol{x}_0 \in \mathbb{R}^n$ and a convergence parameter $\epsilon > 0$
- **Output**: an approximation $\boldsymbol{x}^*$of the solution
- **Initialization**: $k = 0$
- **Iterations**:
  - $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \nabla^2 f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)$
  - $k = k + 1$
- **Stopping criterion**: if $\|\nabla f(\boldsymbol{x})\| \leq \epsilon$, then $\boldsymbol{x}^* = \boldsymbol{x}_k$

Newton's Method: Direction

Important Remark

- With the **steepest descent** method, the direction $\boldsymbol{d}_k$ is given by

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

  Then we determine a step α_k and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$

- With the **Newton's method**, the direction $\boldsymbol{d}_k$ is given by

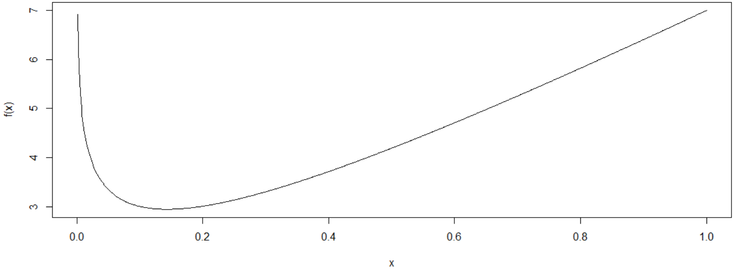
$$\boldsymbol{d}_k = -\nabla^2 f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)$$

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

- When the method converges to the optimal solution, the Newton's direction is often a much better direction than the one provided by the steepest descent

Newton's Method: Example

- We would like to minimize the function given by $f(x) = 7x - ln(x)$
- The domain of f is $x > 0$
- Its first derivative is given by $f'(x) = 7 - \frac{1}{x}$ and its second derivative by $f''(x) = \frac{1}{x^2}$
- It is not difficult to check that $x^* = \frac{1}{7}$ is the **unique global minimizer**



Newton's Method: Example (Cont'd)

- The Newton's direction at x is

$$d = -\nabla^2 f(x)^{-1} \nabla f(x) = -\frac{f'(x)}{f''(x)} = -x^2(7 - \frac{1}{x}) = x - 7x^2$$

- So, the Newton's method generates the sequence of iterates $\{x_k\}$ with

$$x_{k+1} = x_k + d_k = x_k + (x_k - 7x_k^2) = 2x_k - 7x_k^2$$

Newton's Method: Example (Cont'd)

- The table below gives some examples of the sequences generated by this method for **different** starting points $x_0 = 1, y_0 = 0.1, z_0 = 0.01$:

k	x_k	y_k	z_k
0	1	0.1	0.01
1	**-5**	0.13	0.0193
2		0.1417	0.0359925
3		0.14284777	0.062916884
4		0.142857142	0.098124028
5		**0.142857143**	0.128849782
6			0.1414837
7			0.142843938
8			0.142857142
9			0.142857143
10			**0.142857143**

- When we start at $x_0 = 1$, the next iterate is $x_1 =$ -5. But $f(x)$ is not defined for $x \leq 0$! The algorithm has failed !
- The sequences $\{y_k\}$ and $\{z_k\}$ have converged to $1/7$ but not $\{x_k\}$

Issues with the Newton's Method

- When the method converges, the iterates of Newton's method are attracted to **critical points**. Indeed, the method is just trying to solve the system of equations $\nabla f(\boldsymbol{x}) = 0$
- There is **no guarantee** that the sequence of iterates will converge to the optimum
- The **hessian** is assumed to be **nonsingular** at each iteration. Indeed, even if $\nabla^2 f(\boldsymbol{x}_k)$ is nonsingular, it may converge to a non-singular matrix
- $\boldsymbol{d}_k$ is not guaranteed to be a **descent** direction, unless $\nabla^2 f(\boldsymbol{x}_k)$ is positive definite
- It may be **too expensive** to compute second order partial derivatives in particular when the dimension of the system is high