Lecture 5 - Newton's Method

Objective: find an optimal solution of the problem

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}.$$

- f is twice continuously differentiable over $\mathbb{R}^n$.

- Given $\mathbf{x}_k$, the next iterate $\mathbf{x}_{k+1}$ is chosen to minimize the quadratic approximation of the function around $\mathbf{x}_k$:

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\nabla^2 f(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k) \right\}.$$

  This formula is not well-defined in general.
- If $\nabla^2 f(\mathbf{x}_k) \succ \mathbf{0}$,

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

- The vector $-(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$ is called Newton's direction

Pure Newton's Method

Pure Newton's Method

Input: $\varepsilon > 0$ - tolerance parameter.

Initialization: pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily.
General step: for any $k = 0, 1, 2, \ldots$ execute the following steps:

(a) Compute the Newton direction $\mathbf{d}_k$, which is the solution to the linear system $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.

(b) Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$.

(c) if $\|\nabla f(\mathbf{x}_{k+1})\| \leq \varepsilon$, then STOP and $\mathbf{x}_{k+1}$ is the output.

(non)Convergence of Newton's method

- At the very least, Newton's method requires that $\nabla^2 f(\mathbf{x}) \succ \mathbf{0}$ for every $\mathbf{x} \in \mathbb{R}^n$, which in particular implies that there exists a unique optimal solution $\mathbf{x}^*$. However, this is not enough to guarantee convergence.

Example: $f(x) = \sqrt{1+x^2}$. The minimizer of f over $\mathbb{R}$ is of course $x = 0$. The first and second derivatives of f are:

$$f'(x) = \frac{x}{\sqrt{1+x^2}}, \; f''(x) = \frac{1}{(1+x^2)^{3/2}}.$$

Therefore, (pure) Newton's method has the form

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} = x_k - x_k(1+x_k^2) = -x_k^3.$$

Divergence when $|x_0| \geq 1$, fast convergence when $|x_0| < 1$.

convergence of Newton's method

- A lot of assumptions are required to be made in order to guarantee convergence of the method.
- However, Newton's method does have one very attractive feature – under certain assumptions one can prove local quadratic rate of convergence, which means that near the optimal solution the errors $e_k = \|\mathbf{x}_k - \mathbf{x}^*\|$ satisfy an inequality $e_{k+1} \leq Me_k^2$ for some positive $M > 0$.
- This property essentially means that the number of accuracy digits is doubled at each iteration.
- This is in contrast to the gradient method in which the convergence theorems are rather independent in the starting point, but only "relatively" slow linear convergence is assured.

Thm: Quadratic Convergence of Newton's Method

Theorem. Let f be a twice continuously differentiable function defined over $\mathbb{R}^n$. Assume that

- There exists $m > 0$ for which $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$ for any $\mathbf{x} \in \mathbb{R}^n$.
- There exists $L > 0$ for which $\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the sequence generated by Newton's method and let $\mathbf{x}^*$ be the unique minimizer of f over $\mathbb{R}^n$. Then for any $k = 0, 1, \ldots$ the inequality

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \frac{L}{2m}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2$$

holds. In addition, if $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq \frac{m}{L}$, then:

$$\|\mathbf{x}_k - \mathbf{x}^*\| \leq \frac{2m}{L}\left(\frac{1}{4}\right)^{2^k}, \quad k = 0, 1, 2, \ldots$$

See proof of Theorem 5.2 on page 85 of the book.

Numerical Example

Consider the minimization problem

$$\min 100x^4 + 0.01y^4,$$

- optimal solution: $(x, y) = (0, 0)$.
- poorly scaled problem

```
>> f=@(x)100*x(1)^4+0.01*x(2)^4;
>> g=@(x)[400*x(1)^3;0.04*x(2)^3];
>> [x,fun_val]=gradient_method_backtracking(f,g,[1;1],1,0.5,0.5,1e-6)
iter_number =   1 norm_grad = 90.513620 fun_val = 13.799181
iter_number =   2 norm_grad = 32.381098 fun_val = 3.511932
iter_number =   3 norm_grad = 11.472585 fun_val = 0.887929
      :                    :                      :
iter_number = 14611 norm_grad = 0.000001 fun_val = 0.000000
iter_number = 14612 norm_grad = 0.000001 fun_val = 0.000000
```

Numerical Example Contd.

Invoking pure Newton's method we obtain convergence after only 17 iterations.

```
>>h=@(x)[1200*x(1)^2,0;0,0.12*x(2)^2];
>>pure_newton(f,g,h,[1;1],1e-6)
iter=  1 f(x)=19.7550617284
iter=  2 f(x)=3.9022344155
iter=  3 f(x)=0.7708117364
       :              :
iter= 15 f(x)=0.0000000027
iter= 16 f(x)=0.0000000005
iter= 17 f(x)=0.0000000001
```

Numerical Example 2

Consider the minimization problem

$$\min \sqrt{x_1^2+1}+\sqrt{x_2^2+1},$$

- Optimal solution $\mathbf{x}=\mathbf{0}$.
- The Hessian of the function is

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{1}{(x_1^2+1)^{3/2}} & 0 \\ 0 & \frac{1}{(x_2^2+1)^{3/2}} \end{pmatrix} \succ \mathbf{0},$$

  but there does not exists an $m>0$ for which $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$.

```
>>f=@(x)sqrt(1+x(1)^2)+sqrt(1+x(2)^2)
>>g=@(x)[x(1)/sqrt(x(1)^2+1);x(2)/sqrt(x(2)^2+1)];
>>h=@(x)diag([1/(x(1)^2+1)^1.5,1/(x(2)^2+1)^1.5]);
>>pure_newton(f,g,h,[1;1],1e-8)
iter=  1 f(x)=2.8284271247
iter=  2 f(x)=2.8284271247
   :              :
iter= 30 f(x)=2.8105247315
iter= 31 f(x)=2.7757389625
iter= 32 f(x)=2.6791717153
iter= 33 f(x)=2.4507092918
iter= 34 f(x)=2.1223796622
iter= 35 f(x)=2.0020052756
iter= 36 f(x)=2.0000000081
iter= 37 f(x)=2.0000000000
```

Numerical Example 2 Contd.

Gradient method with backtracking and parameters $(s, \alpha, \beta) = (1, 0.5, 0.5)$ converges after only 7 iterations.

```
>>[x,fun_val]=gradient_method_backtracking(f,g,[1;1],1,0.5,0.5,1e-8);
iter_number =   1 norm_grad = 0.397514 fun_val = 2.084022
iter_number =   2 norm_grad = 0.016699 fun_val = 2.000139
iter_number =   3 norm_grad = 0.000001 fun_val = 2.000000
iter_number =   4 norm_grad = 0.000001 fun_val = 2.000000
iter_number =   5 norm_grad = 0.000000 fun_val = 2.000000
iter_number =   6 norm_grad = 0.000000 fun_val = 2.000000
iter_number =   7 norm_grad = 0.000000 fun_val = 2.000000
```

Numerical Example 2 Contd. Starting from (10; 10)

```
>>[x,fun_val]=gradient_method_backtracking(f,g,[10;10],1,0.5,0.5,1e-8);
iter_number =   1 norm_grad = 1.405573 fun_val = 18.120635
iter_number =   2 norm_grad = 1.403323 fun_val = 16.146490
     :                 :                      :
iter_number =  12 norm_grad = 0.000049 fun_val = 2.000000
iter_number =  13 norm_grad = 0.000000 fun_val = 2.000000

>>pure_newton(f,g,h,[10;10],1e-8);
iter=  1 f(x)=2000.0009999997
iter=  2 f(x)=1999999999.9999990000
iter=  3 f(x)=1999999999999997300000000000.0000000
iter=  4 f(x)=1999999999999999230000000000000000000....
iter=  5 f(x)=       Inf
```

- Newton's method seem to be unreliable – partly since no stepsize was defined.

Damped Newton's Method

Damped Newton's Method

Input: (α, β) - parameters for the backtracking procedure $(\alpha \in (0,1), \beta \in (0,1))$
$\varepsilon > 0$ - tolerance parameter.

Initialization: pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily.
General step: for any $k = 0, 1, 2, \ldots$ execute the following steps:

(a) compute the Newton direction $\mathbf{d}_k$, which is the solution to the linear system $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.

(b) set $t_k = 1$. While

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + t_k \mathbf{d}_k) < -\alpha t_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k.$$

set $t_k := \beta t_k$

(c) $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k$.

(c) if $\|\nabla f(\mathbf{x}_{k+1})\| \leq \varepsilon$, then STOP and $\mathbf{x}_{k+1}$ is the output.

Numerical Example 2 Contd. Starting from $(10; 10)$

Using damped Newton's method:

```
>>newton_backtracking(f,g,h,[10;10],0.5,0.5,1e-8);
iter=  1 f(x)=4.6688169339
iter=  2 f(x)=2.4101973721
iter=  3 f(x)=2.0336386321
   :             :
iter= 16 f(x)=2.0000000005
iter= 17 f(x)=2.0000000000
```