Semester project TMA4215

10 026 and 10 068

03.10.2011

1 Task

We consider minimization problems of the type

$$\min_{\mathbf{x}\in\mathbb{R}^n} g(\mathbf{x}),\ g(\mathbf{x}) := -\mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \frac{1}{12}\mathbf{x}^T C(\mathbf{x})\mathbf{x},$$

here $\mathbf{b} \in \mathbb{R}^n$ and, H is a $n \times n$ symmetric and positive definite matrix and $C(\mathbf{x})$ is a diagonal matrix with diagonal entries $c_i x_i^2, i = 1, ..., n$. Here $c_i > 0$ are the components of a vector $\mathbf{c} \in \mathbb{R}^n$ and x_i are the components of $\mathbf{x}$.

2 Mathematical calculations

At first some calculations are done.

2.1 Positive definition of H

Since we use only one H, the proof is not generally, but only for one special matrix. Let $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \in \mathbb{R}^2 \backslash \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and $H = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, with $a, c > 0$.

$$\mathbf{u}^T H\mathbf{u} = au_1^2 + cu_2^2 + 2bu_1u_2$$

If we choose $b = \sqrt{a}\sqrt{c}$, we get

$$= (\sqrt{a}u_1 + \sqrt{c}u_2)^2 > 0,$$

no matter what $\mathbf{u}$ is. H is positive definite with this choice of b.

2.2 Gradient

The gradient can easily be calculated with sums.

$$\nabla g = \nabla\left(-\sum_{i=1}^{n} b_i x_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} H_{ij}x_i x_j + \frac{1}{12}\sum_{i=1}^{n} c_i x_i^4\right)$$

The two sums in the middle are divided into the diagonal element and the not diagonal elements. All not diagonal elements are there twice, because H is symmetric and therefore $H_{ij} = H_{ji}$.

$$= \nabla\left(-\sum_{i=1}^{n} b_i x_i + \frac{1}{2}\sum_{i=1}^{n} H_{ii}x_i^2 + \sum_{i=1}^{n}\sum_{j=1}^{i-1} H_{ij}x_i x_j + \frac{1}{12}\sum_{i=1}^{n} c_i x_i^4\right)$$

$$= -\mathbf{b} + H\mathbf{x} + \frac{1}{3}C\mathbf{x}$$

2.3 Hessian

The Hessian of g is easily calculable:

$$\nabla^2 g = H + C$$

2.4 Existence of minimum

Let $u \in R^n$ be an arbitrary vector, except $\vec{0}$ then

$$u^T(\nabla^2 g(x))u = u^T(H+C)u = u^T H u + u^T C u = u^T H u + \sum_{i=1}^{n} c_i u_i^2 x_i^2.$$

Since H is positive definite, $u^T H u > 0$ and because $c_i > 0$, $u^T C u > 0$, so

$$u^T(\nabla^2 g(x))u > 0$$

That means that the Hessian of g is positive definite and therefore strictly convex and has at most one local minimum.

2.5 Equivalence of steepest decent method and forward Euler method

2.6 Optimal α in the steepest decent method

To find the optimal α,

$$g\left(\mathbf{x}^{(k+1)}\right) = g\left(\mathbf{x}^{(k)} - \alpha^{(k)} \nabla g(\mathbf{x}^{(k+1)})\right)$$

has to be minimal, so $\frac{\partial}{\partial \alpha} g(\mathbf{x}^{(k+1)})$ has to be zero. This leads to the equation

$$\begin{aligned} f = &\left(\mathbf{b}\nabla g - \mathbf{x}^T\left(H + \frac{C}{3}\right)\nabla^T g\right) + \left((\nabla g)(H+C)\nabla g\right)\alpha \\ &+ \left(-\sum_{i=1}^{n} c_i x_i (\nabla g)_i^3\right)\alpha^2 + \frac{1}{3}\left(\sum_{i=1}^{n} c_i (\nabla g)_i^4\right)\alpha^3 \end{aligned}$$

Since the function is a cubic polynomial, the limit for $\lim_{\alpha \to \pm\infty} f = \pm\infty$ and because the function is continuous, there has to be at least one real zero.

The function is always rising. So only one zero is possible.

3 Main algorithms

3.1 Generation of the data

The data is generated in the function *data*.

```
1 function [ b, H, c ] = data
2         b = [ 1; 0 ];
3         c = [ 200; 400 ];
4         H = [ 200, 20 ; 20, 2 ];
5 end
```

With the calculations in chapter 2.1 can be easily seen that H is positive definite.

3.2 Function, gradient and Hessian of g

```
function [ g ] = g ( X )
        [ b, H, c ] = data;
        dim = size(H,1);
        C = zeros ( dim, dim );
        for i = 1 : dim
                C(i,i) = c(i) * X(i) * X(i);
        end
        g = - b' * X + 0.5 * X' * H * X + 1/12 * X' * C * X;
end
```

```
function [ nablaG ] = grad( X )
        [ b, H, c ] = data;
        dim = size ( H, 1 );
        for i = 1 : dim
            C(i,i) = c(i) * X(i) * X(i);
        end
        nablaG = - b + H * X + 1/3 * C * X;
end
```

```
function [ hessG ] = hessian( X )
        [ ~, H, c ] = data;
        dim = size ( H, 1 );
        C = zeros ( dim, dim );
        for i = 1 : dim
                C(i,i) = c(i) * X(i) * X(i);
        end
        hessG = H + C;
end
```

3.3 Minimum searching algorithm

All methods work similar, only line 9 has to be exchanged. The real implementation is more complicated and is shown in the next chapter.

```
1  X = [ 4; -1 ];
2  maxiterations = 20;
3  tol = 1e-8;
4
5  norm_old = norm ( grad ( X) );
6  condition = 1;
7  while condition
8          maxiterations = maxiterations - 1;
9          X = X - alpha * grad ( X ); % steepest decent method with constant alpha
10         % X = X - optimalAlpha ( X ) * grad ( X ); % with optimal alpha
11         % X = X - linsolve ( hessian ( X ), grad ( X ) ); % Newton method
12         residual = norm ( grad ( X ) ) / norm_old;
13         condition = (maxiterations > 0) && ( residual > tol);
14 end
```

3.4 Computation of α

For the calculation of α see chapter 2.6.

```
function [ alpha_optimal ] = optimalAlpha( X )
[ b, H, c] = data;
dim = size ( H, 2 );
C = zeros ( dim, dim );
gradg = grad ( X );
```

```
a3 = 0;
a2 = 0;
for i = 1 : dim
        C(i,i) = c(i) * X(i)^2;
        a3 = a3 + c(i) / 3 * gradg(i)^4;
        a2 = a2 - c(i) * X(i) * gradg(i)^3;
end
a1 = gradg' * ( H + C ) * gradg;
a0 = b'*gradg - X' * H * gradg - 1/3 * X' * C * gradg;
alphas = roots( [ a3, a2, a1, a0 ] );
for i = 1 : 3
        if imag ( alphas ( i ) ) == 0
                alpha_optimal = alphas ( i );
        end
end
end
```

4 Structure of the project

The project can be started by executing *main()*. There the initial guess, the tolerance and the maximal number of iterations is defined. *main* first calls *plotmethod*, which contains a variant of the code in chapter 3.3 and returns the relative residuals, and the points $\mathbf{x}^k$. The change of $\mathbf{x}$ in each iteration is calculated in *delta* for various methods. At the end of *main*, all plots are generated using the functions *drawfunction* and *drawcurve*.

To count the time a process needs, there are the functions *meantime*, which makes many experiments and *fastmethod*, which is a smaller variant of *plotmethod*.

5 Results

With the program presented above, several experiments can be done. If not other stated, $(4,-1)^T$ is used as initial guess, 10^{-8} as tolerance and the number of iterations is limited to 30. A first look at the function with the defined values suggest the minimum is around $(0,0)^T$.

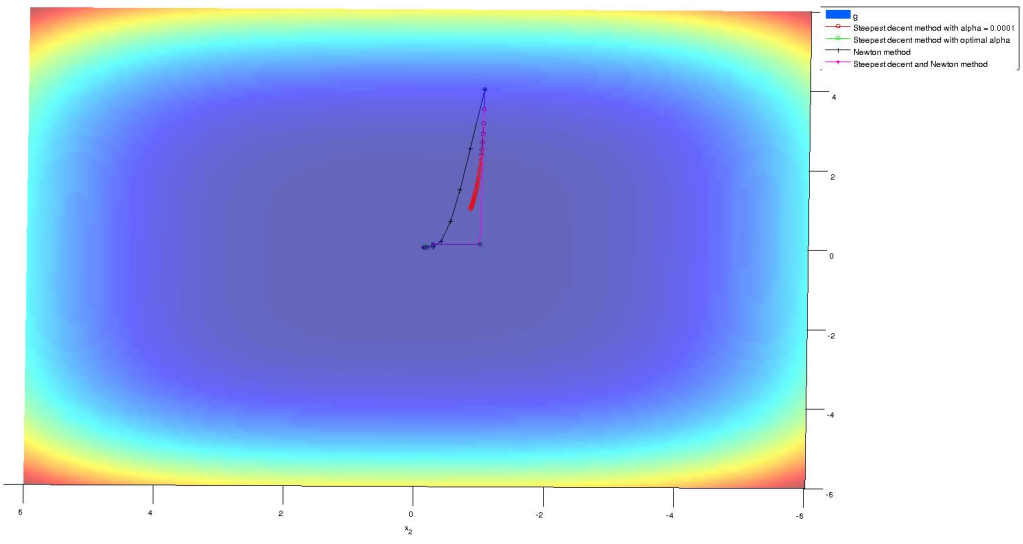

Figure 1: The function g.

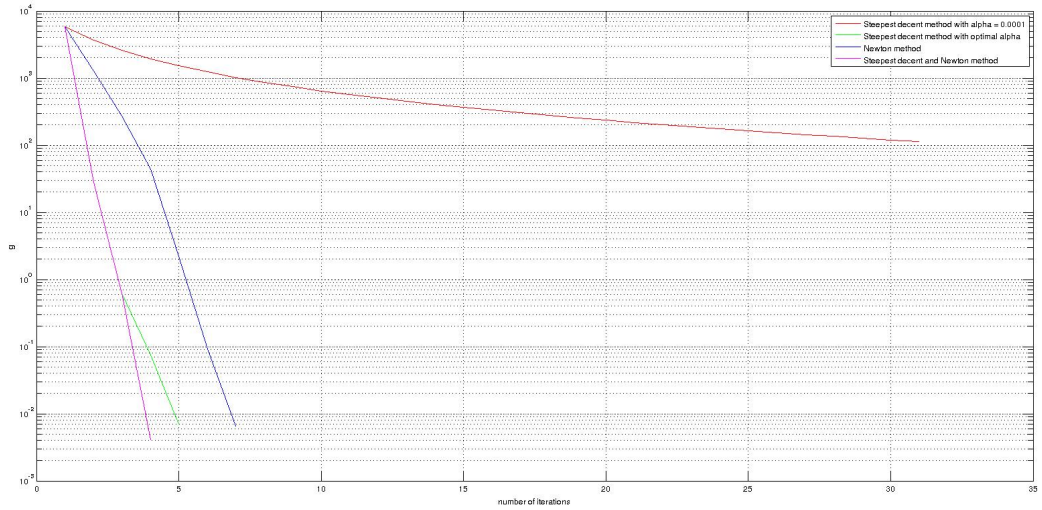

Figure 2: The value of g as a function of iterations for various methods.

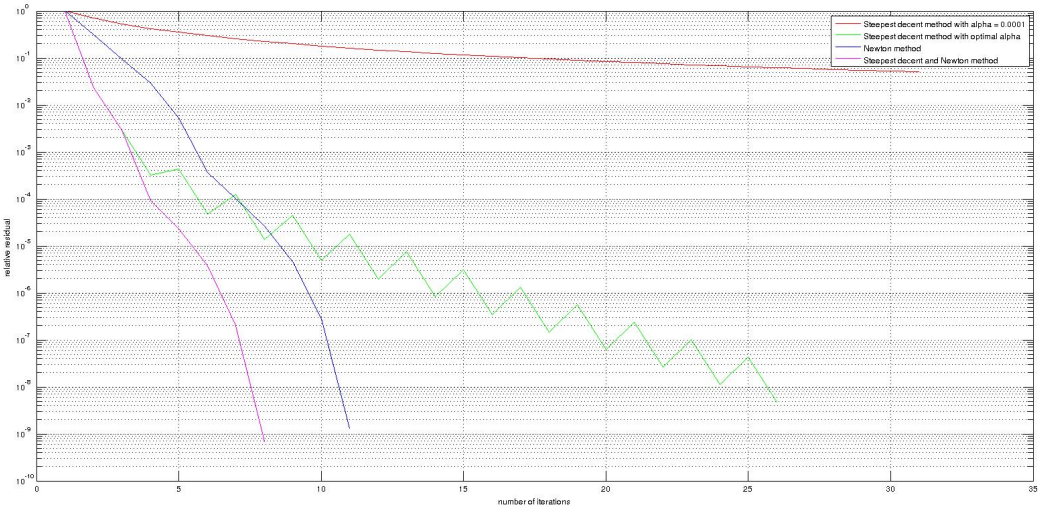

Figure 3: The relative residuals as a function of iterations for various methods.

5.1 Steepest decent method with constant α

If the initial guess is far away from the solution, the gradient at this point is very large. To get $\mathbf{x}^2$, this gradient has to be multiplied by α. If α is not small enough, the value of g for the next iteration gets infinity and the program breaks down. With $\alpha = 0.0001$, the algorithm works, but the minimum is only slowly approached. The tolerance is reached after $744\,\mathrm{ms}$ and $25\,473$ iterations, so in most cases the limiting factor is the maximal number of iterations.

5.2 Steepest decent method with optimal α

The algorithm works better than with constant α, but when the relative residual drops below 10^{-3}, the relative residual rises and falls alternately, but with a dropping drift. It reaches the tolerance for the relative residual after 30 iterations and needs $4.33\,\mathrm{ms}$.

5.3 Newton method

At the beginning the Newton method reaches the minimum more slowly than the steepest decent method with optimal α. After the relative residual drop below 10^{-3}, the Newton method converges faster. It reaches the tolerance after $1.69\,\mathrm{ms}$ and 11 iterations.

If the initial guess for the minimum is far away from the minimum, the Newton method converges more slowly than the steepest decent method with optimal alpha.

5.4 Steepest decent method and Newton method

First the steepest decent method is applied and after a certain tolerance limit, the algorithm switches to the Newton method.

To find this tolerance limit, the total time for the algorithm is watched with various tolerance limits. The program is run 1000 times for each limit and the mean is given in tabular 5.4. To run the function with minimal time, the steepest decent method is applied till the relative residual is below 10^{-2} and the Newton method is applied till the default tolerance (10^{-8}) is reached.

tolerance limit	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
time $[ms]$	1.73	1.72	1.79	1.92	4.24	4.27

Table 1: Times for the mixture of steepest decent and Newton method for various changing conditions.

We decided to keep the number of iterations constant, because 20 iterations is not much.