Semester project TMA4215

10068 and 728387

02.10.2011

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 $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \in \mathbb{R}^2 \setminus \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, 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

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

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

$$\begin{aligned} &\left(\mathbf{b}\nabla g - (\nabla g)H\mathbf{x} - \frac{1}{3}\mathbf{x}H\nabla g\right) + \left((\nabla g)(H+C)\nabla g\right)\alpha \\ &+ \left(-\sum_{i=1}^{n} c_i x_i (\nabla g)_i^3\right)\alpha^2 + \frac{1}{3}\left(\sum_{i=1}^{n} c_i (\nabla g)_i^4\right)\alpha^3 \\ = a_0 + a_1\alpha + a_2 o^2 + a_3 c\alpha^3 = 0 \end{aligned}$$

Because the function is cubic, 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 definit.

3.2 Function, gradient and Hessian of g

```
1  function [ g ] = g ( X )
2          [ b, H, c ] = data;
3          dim = size(H,1);
4          C = zeros ( dim, dim );
5          for i = 1 : dim
6                  C(i,i) = c(i) * X(i) * X(i);
7          end
8          g = - b' * X + 0.5 * X' * H * X + 1/12 * X' * C * X;
9  end
```

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

```
1  function [ hessG ] = hessian( X )
2          [ ~, H, c ] = data;
3          dim = size ( H, 1 );
4          C = zeros ( dim, dim );
5          for i = 1 : dim
6                  C(i,i) = c(i) * X(i) * X(i);
7          end
8          hessG = H + C;
9  end
```

3.3 Minimum searching algorithm

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

```
1  X = [ 4; -1 ];
2  maxiterations = 1000;
3  tol = 1e-6;
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.

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

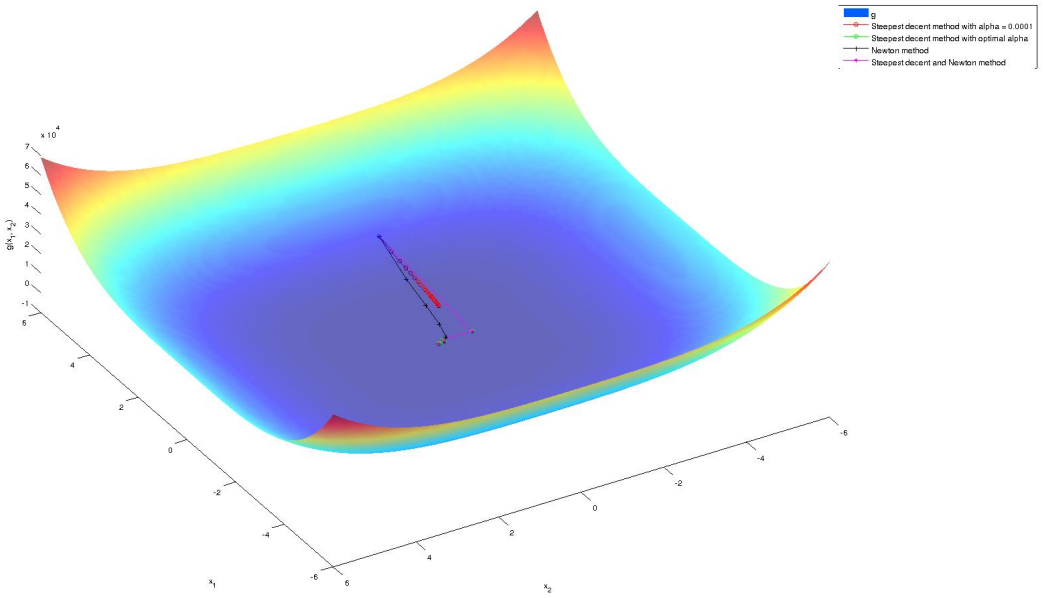

Figure 1: The function g.

```
 6  a3 = 0;
 7  a2 = 0;
 8  for i = 1 : dim
 9          C(i,i) = c(i) * X(i)^2;
10          a3 = a3 + c(i)/3 * gradg(i)^4;
11          a2 = a2 - c(i) * X(i) * gradg(i)^3;
12  end
13  a1 = gradg' * ( H + C ) * gradg;
14  a0 = b'*gradg - X' * H * gradg - 1/3 * X' * C * gradg;
15  alphas = roots( [ a3, a2, a1, a0 ] );
16  for i = 1 : 3
17          if imag ( alphas ( i ) ) == 0
18                  alpha_optimal = alphas ( i );
19          end
20  end
21  end
```

4 Structure of the project

The project can be started by executing *main()*. *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*.

5 Results

With the program presented above, several experiments can be done. A first look at the function with the defined values suggest the minimum is around $(0, 0)^T$.

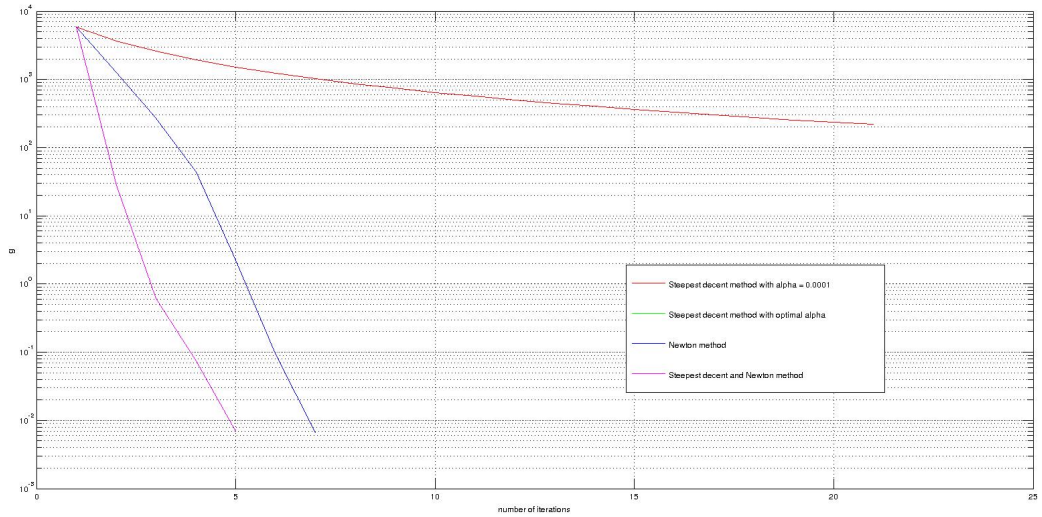

Figure 2: The value of g as a function of iterations for various methods.

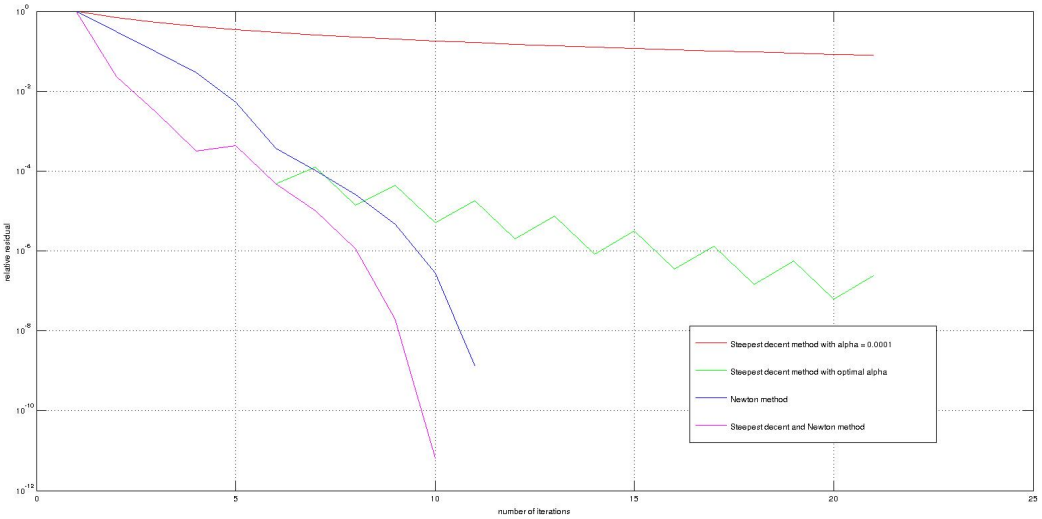

Figure 3: The relative residuals as a function of iterations for various methods.

5.1 Steepest decent method with constant α

The larger $|x|$, the smaller has to be α. If α is larger, the method does not converge any more.

The steepest decent method with constant α is alway interrupted when reaching the maximum number of iterations and not by reaching the tolerance for the residual. For the starting value of $(4, -1)^T$, the algorithm needs $0.0957s$ for 1000 iterations.

5.2 Steepest decent method with optimal α

First the maximal numbers of iteration was decreased, to have a better view for low numbers of iteration. 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.

5.3 Newton method

6 Your text, Im not sure if we should write that in our protocol

All the values of the table are average values of a 1000 values sample, calculated from the starting point [4;-1]. As we can see, regardless of the tolerance and the maximun number of iterations, Newton's method is the fastest method. When we have a large number of iterations, the combination of steepest and Newton's method is the most precise (smallest residual). For this last method, the precision is increasing while the tolerance is decreasing but obviously, the cputime increases too. The first method, steepest descent with arbitrary alpha appears to be the worst method. Indeed, the cputime and the residual are greater there. But, what is the interest of steepest method with the optimal alpha ? Actually, this method gives us a precise idea of where is the minimum of the function, without being very precise. The Newton's method can give us the same idea, but if we start far from the answer, the time of execution will be very higher. Moreover, if there are several points where the gradient equals to zero, we can't be sure that we are precisely in a global minimum or a local extremum. That is why it seems to be very useful to combine the two methods, because with the steepest descent we approach the solution to a certain point (and we are sure that is the solution) and with the Newton's method applied to this point we can go much more closer to the minimum. For instance, here with the tolerance 10e-6 and 1000 iterations, the precision of the Newton-steepest method is 1 million times more precise than the Newton's method alone.

As a conclusion, we have to choose the method accorded to the result we expect : if we just want a quick idea of the solution, we can just use the steepest descent with an alpha, arbitrary or optimal. But, if we prefer to be certain of the solution, we have to mix Newton's and steespest methods. The Newton's method is between the last ones : very fast and quite precise but we cannot be completely sure of the solution, given that it will converge to the closest zero."

6.1 Optimal residual for steepest newton

the maximal number of iterations is never reached, so only the tolerance is changed.