MTH437: Advanced Calsulus and Application 1

Jie He

December 2016

1 Abstract

The task of this project is to develop the Steepest Descent Method for solving linear system. For example, solving equation of the form

$$Ax = b \tag{1}$$

where A is an $n \times n$ matrix and b is a vector in R^n, which means that the solution for this problem is a vector $x \in R^n$.

For this kind of problem, the thought is converting the multiple-dimension(R^n) into one-dimension (R^1), like:

$$F(x) = \frac{1}{2}x^T Ax - x^T b \tag{2}$$

So that f(x): $R^n \to R$. What we should do is just to find the minimal value fitting to the accuracy requirement.

At the same time,

$$\begin{aligned}
\nabla f(x) =&(\frac{1}{2}X^T)' Ax + \frac{1}{2}x^T(Ax)' - (x^T b)' \\
=&\frac{1}{2}x^T A^T + (\frac{1}{2}X^T)A^T - b^T \\
=&x^T A^T - b^T \\
\nabla f(x)^T =&Ax - b
\end{aligned}$$

This report will focus on the requirement of A and how to iterately find the point that makes the function value minimal.

2 Steepest Descent Method

2.1 Theory used in this essay

Theory 1: if f: $R^n \to R.$ is differentiable at X_0, $R^n \to R.$ $\exists p \in R^n$, let

$$\nabla f(x_0)^T P < 0$$

so that vector P is the descenting direction of f at point x_0.

Theory 2: if f: $R^n \to R.$ is differentiable at $X_0 R^n \to R.$ if the x_0 is the local optimal solution to unconstrained problems, so that

$$\nabla f(x_0) = 0$$

From the First Derivative Test for Local Extrema, we know that even $\nabla f(x_0) = 0$ X_0 can be critical point or saddle point. so we need further restriction to A.

Theory 3: According to Second Derivative Test for Local Extrema, we know that if Hessian matrix exist and and this matrix is positive-definite, the value of point is local minimum.

$$Hf(x_0)(x) = X^T AX > 0 \tag{3}$$

Theory 4: According the definition of positive definite matrix if $Hf(x_0)(x) = X^T AX > 0$, the A is a positive definite matrix.

$$Hf(x_0)(x) = X^T AX > 0 \Longleftrightarrow \text{A is positive definite} \tag{4}$$

According to theory (2),(3),(4), we know that if A is symmetric and positive definite matrix, we will get a minimum of f(x).

2.2 Algorithm

Direction:Because we should do iteration, so that our thought is that let:

$$x_{k+1} = x_k + t_k * d_k \tag{5}$$

in which, t_k is the length of each step, and the d_k Plug (5) into f(x), and use Tyler theory:

$$f(x_{k+1}) - f(x_k) = t_k \nabla f(x_k)^T d_k + o(t_k d_k) \tag{6}$$

Compared with first order item $-t_k d_k \nabla f(x_k)$, the reminder $o(t_k d_k)$ is so small that it can be canceled.

$t_k d_k \nabla f(x_k)$ is the variable value of f in each iteration. what we should do is to make sure that the reduction should be the biggest.we can easily find that the steepes descent direction is

$$d_k = -\nabla f(x_k) \tag{7}$$

Length of Step: Set

$$g_k = \nabla f(x_k) = Ax - b$$

which represents the residual of each iteration. hence, wheng_k is less or equal than the accuracy requirement, the algorithm ends up. The formula of Steepest Descent Method is

$$x_{k+1} = x_k - t_k \nabla f(x_k); t_k > 0 \tag{8}$$

in which

$$t_k = arg \min_{t_k > 0} f(x_k - t_k \nabla f(x_k))$$

We can set a new function

$$\varphi(t_k) = f(x_k - t_k \nabla f(x_k)$$

When g_k is bigger than accuracy requirement, according to Theory 2, The Necessary Conditions for minimum value:

$$\varphi^{'}(t_k) = (x_k - t_k \nabla f(x_k)^T) A(-\nabla f(x_k)) - b^T \nabla f(x_k) = 0$$

$$t_k \nabla f(x_k)^T A \nabla f(x_k) = (x_k^T A - b^T) \nabla f(x_k)$$

Because A is symmetric metrix,$A = A^T$, so

$$\nabla f(x_k)^T = x_k^T A - b^T$$

hence:

$$t_k \nabla f(x_k)^T A \nabla f(x_k) = \nabla f(x_k)^T \nabla f(x_k)$$

$$t_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T A \nabla f(x_k)} \tag{9}$$

Ending condition: There are two ending condition. one of them is that we find the X, in which the ending condition is

$$|\nabla f(x_k)^T| \leq \varepsilon$$

one is that we we don't find the X, in which the minimum value of $\nabla f(x_k)$ is bigger than the accuracy requirement.

$$|\nabla f(x_{k+1})^T| - |\nabla f(x_k)^T| > 0$$

So that the iteration of Steepest Descent Method is

$$\begin{cases} x_{k+1} = x_k - t_k \nabla f(x_k); t_k > 0 \\ \\ t_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T A \nabla f(x_k)} \\ \\ Ending : |\nabla f(x_k)^T| \leq \varepsilon or |\nabla f(x_{k+1})^T| - |\nabla f(x_k)^T| > 0 \end{cases} \tag{10}$$

if input: A $\in$ $\mathrm{R}^{2\times 2}$ and b $\in\mathrm{R}^2$

Step 1: set an original point

$$X_k{}^T = (0,0), k = 0$$

Step 2: plug X_0 into $g_k = \nabla f(x_k) = Ax - b$ if ($g_k <= 10^{-5}$), end and return X_k; else compute

$$t_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T A \nabla f(x_k)}$$

And go next

step 3:

$$x_{k+1} = x_k - t_k \nabla f(x_k); K = K + 1$$

return to step 2;

3 Application

3.1 Matlab

3.1.1 Source code

```
function [x,iteration]=steepestDescent(A,b,x)
r=A*x-b; % initial step length
iteration=0;
while abs(r) > 10^(-5)   % check the ending condition
  iteration=iteration+1;
  a=r'*r/(r'*A*r);      %iterate length of stepth
  x=x-a*r;              %iterate x
  r=A*x-b;              %iterate residual
  disp(x);              %print out each x
end
```

3.1.2 Iteration of x

$(0.0800000,-0.6133333) \to (1.0044444, -2.0000000) \to (1.5221333, -1.6548741) \to$

$(1.7522173,-2.0000000) \to (1.8810643, -1.9141020) \to (1.9383296, -2.0000000) \to$

$(1.9703982,-1.9786209) \to (1.9846509, -2.0000000 \to (1.9926324, -1.9946790) \to$

$(1.9961798,-2.0000000) \to (1.9981663, -1.9986757) \to (1.9990492, -2.0000000) \to$

$(1.9995436,-1.9996704) \to (1.9997634, -2.0000000) \to (1.9998864, -1.9999180) \to$

$(1.9999411,-2.0000000) \to (1.9999717, -1.9999796) \to (1.9999853, -2.0000000) \to$

$(1.9999930,-1.9999949) \to (1.9999964, -2.0000000)$

3.2 Evaluation

3.2.1 Surface plot

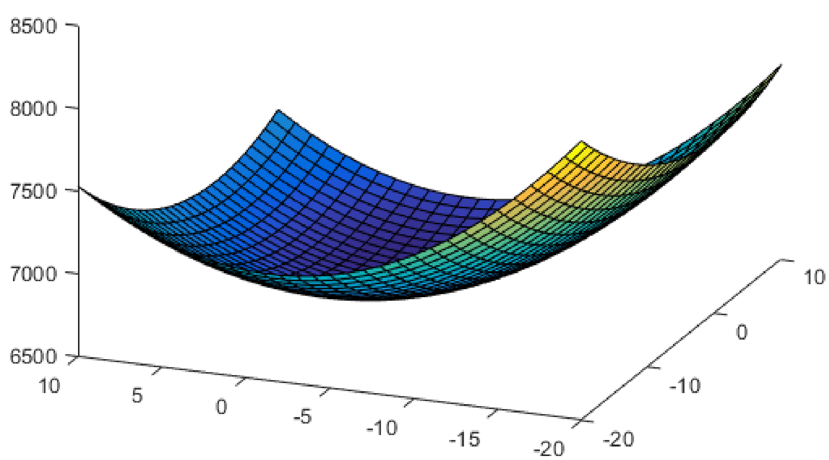

Figure 1: Surface plot

3.2.2 curve plot

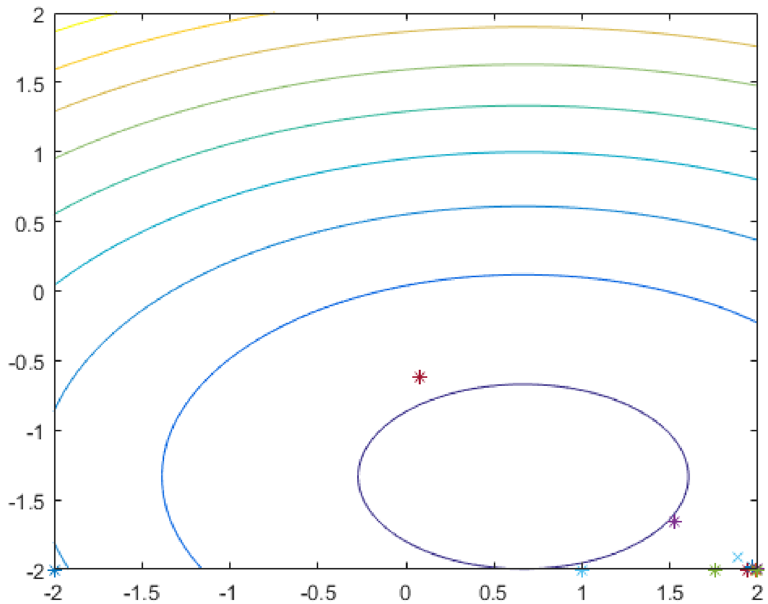

Figure 2: Curve plot

3.2.3 Path

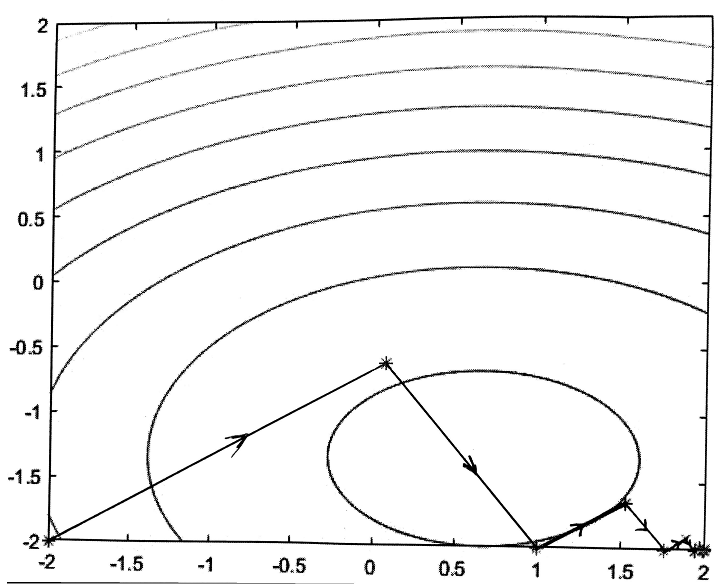

Figure 3: Path

3.2.4 Evaluation

The descent speed method of this method is very fast at the first few iteration, but when we iteration is much closer to the minimal value and we want to find percise solution, it becomes very slow, which is not efficient enough. The reason is that when

$$\nabla f(x_k) \to 0, t_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T A \nabla f(x_k)} \to 0$$

The depth t_K will be closed to 0, which makes iteration much more complicated, especially when the iteration is around the critical point.

However, compared with Gaussian Elimination, Steepest Descent Method can be really fast, when if the accuracy is not so serious(the accuracy is not a very small number.