优化实用算法: 第一次作业

2022 年 5 月 28 日

指导老师：徐翔

徐圣泽 3190102721

Problem 1

1. Prove that $\|Bx\| \geq \|x\|/\|B^{-1}\|$, for any non singular matrix B.
证明： 当 $x=0$ 时，$0=\|Bx\|=\|x\|/\|B^{-1}\|=0$，命题成立。
当 $x \neq 0$ 时，根据矩阵范数的定义，我们有 $\|B^{-1}\| = \max_{x\neq 0}\frac{\|B^{-1}x\|}{\|x\|}$，因此我们有

$$\|B^{-1}\| = \max_{x\neq 0}\frac{\|B^{-1}x\|}{\|x\|} \geq \frac{\|B^{-1}\cdot Bx\|}{\|Bx\|} = \frac{\|x\|}{\|Bx\|}$$

需要注意的是，因为 B 为非奇异阵，因此 $\|Bx\| \neq 0$，故由上式可得 $\|Bx\| \geq \|x\|/\|B^{-1}\|$。
综上所述，原命题成立。

Problem 2

2. Given a square nonsingular matrix A. Consider its rank-one update $\overline{A} = A + ab^T$, where $a, b \in R^n$
(a) Verify that when $\overline{A}$ is nonsingular, we have

$$\overline{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1+b^TA^{-1}a}$$

证明：

$$\begin{aligned}
(A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1+b^TA^{-1}a})\overline{A} &= (A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1+b^TA^{-1}a})(A+ab^T) \\
&= I + \frac{A^{-1}ab^Tb^TA^{-1}a - A^{-1}ab^TA^{-1}ab^T}{1+b^TA^{-1}a} \\
&= I + \frac{A^{-1}ab^Tb^TA^{-1}a - A^{-1}a(b^TA^{-1}a)b^T}{1+b^TA^{-1}a} \\
&= I + \frac{A^{-1}ab^Tb^TA^{-1}a - A^{-1}ab^T(b^TA^{-1}a)}{1+b^TA^{-1}a} \\
&= I
\end{aligned}$$

要注意的是，上式的变换中，因为 $b^TA^{-1}a$ 为实数，因此我们可以在式中交换其位置。
因此有 $(A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1+b^TA^{-1}a})\overline{A} = I$，故 $\overline{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1+b^TA^{-1}a}$。
(b) Using the above formula to show that

$$B_{k+1} = B_k + \frac{(y_k - B_ks_k)(y_k - B_ks_k)^T}{(y_k - B_ks_k)^Ts_k}$$

is the inverse of

$$H_{k+1} = H_k + \frac{(s_k - H_ky_k)(s_k - H_ky_k)^T}{(s_k - H_ky_k)^Ty_k}$$

when $H_k^{-1} = B_k$ is symmetric, $s_k = x_{k+1} - x_k$ and $y_{k+1} = \nabla f(x_{k+1}) - \nabla f(x_k)$.
证明： 由 (a) 知，取 $A = H^k$，$a = s_k - H_ky_k$，$b^T = \frac{(s_k - H_ky_k)^T}{(s_k - H_ky_k)^Ty_k}$，则 $H_{k+1} = A + ab^T = \overline{A}$。

同时由题意，$H_k^{-1}=B_k$，$H_k=H_k^T$，$B_k=B_k^T$，我们有

$$\begin{aligned}
B_{k+1}=H_{k+1}^{-1}=H_k^{-1}-\frac{H_k^{-1}\frac{(s_k-H_ky_k)(s_k-H_ky_k)^T}{(s_k-H_ky_k)^Ty_k}H_k^{-1}}{1+\frac{(s_k-H_ky_k)^TH_k^{-1}(s_k-H_ky_k)}{(s_k-H_ky_k)^Ty_k}}
&=B_k-\frac{B_k(s_k-H_ky_k)(s_k-H_ky_k)^TB_k}{(s_k-H_ky_k)^Ty_k+(s_k-H_ky_k)^TB_k(s_k-H_ky_k)}\\
&=B_k-\frac{(B_ks_k-y_k)(B_ks_k-y_k)^T}{(s_k-H_ky_k)^T(y_k+B_ks_k-y_k)}\\
&=B_k-\frac{(y_k-B_ks_k)(y_k-B_ks_k)^T}{(s_k-H_ky_k)^TB_k^Ts_k}\\
&=B_k-\frac{(y_k-B_ks_k)(y_k-B_ks_k)^T}{(B_ks_k-y_k)^Ts_k}\\
&=B_k+\frac{(y_k-B_ks_k)(y_k-B_ks_k)^T}{(y_k-B_ks_k)^Ts_k}
\end{aligned}$$

Problem 3

3. Minimize the Rosenbrock function $f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$ and Beale function $f(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$ by the steepest descent method and Newtons method respectively, where $x^{(0)}=(-1.2,1)^T$.

解：对于本题，我们通过编写程序求解，分别用最速下降法和牛顿法求解得到极小值点和极小值，并通过比较两种方法的迭代次数来衡量收敛速度。

首先令 $f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$,$g(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$。

```
function y=f(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
end
```

```
function y=g(x)
y=(1.5-x(1)+x(1)*x(2))^2+(2.25-x(1)+x(1)*x(2)^2)^2
+(2.625-x(1)+x(1)*x(2)^3)^2;
end
```

下面写出两个函数的梯度：

```
function y=Gradf(x)
y=zeros(2,1);
y(1)=200*(x(1)^2-x(2))*2*x(1)+2*(x(1)-1);
y(2)=200*(x(2)-x(1)^2);
end
```

```
function y=Gradg(x)
y=zeros(2,1);
y(1)=2*(1.5-x(1)+x(1)*x(2))*(x(2)-1)+2*(2.25-x(1)+x(1)*x(2)^2)
*(x(2)^2-1)+2*(2.625-x(1)+x(1)*x(2)^3)*(x(2)^3-1);
```

```
y(2)=2*(1.5-x(1)+x(1)*x(2))*x(1)+2*(2.25-x(1)+x(1)*x(2)^2)*2
*x(1)*x(2)+2*(2.625-x(1)+x(1)*x(2)^3)*3*x(1)*x(2)^2;
end
```

写出两个函数的 *Hesse* 矩阵

```
function y=Hessef(x)
y=zeros(2,2);
y(1,1)=1200*x(1)^2-400*x(2)+2;
y(1,2)=-400*x(1);
y(2,1)=-400*x(1);
y(2,2)=200;
end
```

```
function y=Hesseg(x)
y=zeros(2,2);
y(1,1)=2*(x(2)-1)^2+2*(x(2)^2-1)^2+2*(x(2)^3-1)^2;
y(1,2)=2*(1.5-x(1)+x(1)*x(2))+2*x(1)*(x(2)-1)
+2*(2.25-x(1)+x(1)*x(2)^2)*2*x(2)+2*2*x(1)*x(2)*(x(2)^2-1)
+2*(2.625- x(1)+x(1)*x(2)^3)*3*x(2)^2+2*3*x(1)*x(2)^2*(x(2)^3-1);
y(2,1)=2*(1.5-x(1)+x(1)*x(2))+2*x(1)*(x(2)-1)
+2*(2.25-x(1)+x(1)*x(2)^2)*2*x(2)+2*2*x(1)*x(2)*(x(2)^2-1)
+2*(2.625- x(1)+x(1)*x(2)^3)*3*x(2)^2+2*3*x(1)*x(2)^2*(x(2)^3-1);
y(2,2)=2*x(1)^2+2*(2*x(1)*x(2))^2+2*(2.25-x(1)
+x(1)*x(2)^2)*2*x(1)+2*(3*x(1)*x(2)^2)^2
+2*(2.625-x(1)+x(1)*x(2)^3)*6*x(1)*x(2);
end
```

下面我们首先根据书本 57 至 58 页的算法编写了利用进退法进行一维搜索确定区间的函数:

```
function [a,b]=Range(f,a0,h0,x,p)
t=2;k=0;ak=a0;yk=f(x+ak*p);h=h0;ak1=0;yk1=0;alpha=0;
while(0<=k)
        ak1=ak+h;
        if(ak1<0)
                ak1=0;
                break;
        end
        yk1=f(x+ak1*p);
        if(yk1<=yk)
                h=t*h; alpha=ak; ak=ak1; yk=yk1; k=k+1;
        else
                if k==0
                        h=-h; ak=ak1; yk=f(x+ak*p);
                else
                        break;
                end
```

```
        end
end
if(alpha<ak1)
        a=alpha;
        b=ak1;
else
        a=ak1;
        b=alpha;
end
end
```

下面利用 0.618 法进行精确一维搜索：

```
function y = Minimum(f,a0,b0,x,p,epsilon)
a=a0;b=b0;
lambda=a+0.382*(b-a);mu=a+0.618*(b-a);
y1=f(x+lambda*p);y2=f(x+mu*p);
k=1;y=0;
while(k<=10000)
        if(y1>y2)
                if(b-lambda<=epsilon)
                        y=mu;
                        break;
                end
                a=lambda; lambda=mu;
                y1=f(x+lambda*p);
                mu=a+0.618*(b-a);
                y2=f(x+mu*p);
        else
                if(mu-a<=epsilon)
                        y=lambda;
                        break;
                end
                b=mu;mu=lambda;
                y2=f(x+mu*p);
                lambda=a+0.382*(b-a);
                y1=f(x+lambda*p);
        end
        k=k+1;
end
end
```

接下来我们写最速下降法和牛顿法的函数，首先写最速下降法，返回值是极小值点和迭代次数：

```
function [x,k] = SteepestDescent(f,Gradf,x0,epsilon1,epsilon2)
while(0<=k)
        p=-Gradf(x);
```

```
        if(sqrt(p'*p)<=epsilon1)
                break;
        end
        a0=2;h0=0.5;
        [a,b]=Range(f,a0,h0,x,p);
        x=x+Minimum(f,a,b,x,p,epsilon2)*p;
        k=k+1;
end
end
```

下面是牛顿法，返回值和最速下降法相同：

```
function [x,k] = Newtons(f,Gradf,Hessef,x0,epsilon)
k=0;x=x0;
while(0<=k)
        p=-Gradf(x);
        if(sqrt(p'*p)<=epsilon)
                break;
        end
        x=x+inv(Hessef(x))*p;
        k=k+1;
end
end
```

下面我们取初值为 $x0 = [-1.2; 1]^T$，利用最速下降法 $SteepestDescent(f, Gradf, x_0, epsilon_1, epsilon_2)$ 得到 Rosenbrock 函数和 Beale 函数的极小值点和极小值：

函数	$epsilon_1$	$epsilon_2$	极小值点	极小值	迭代次数
Rosenbrock	10^{-10}	10^{-12}	$(1.000, 1.000)^T$	9.9748×10^{-21}	29239
Beale	5×10^{-4}	10^{-9}	$(-55.1, 1.0)^T$	0.48	139165

上面表格中 $epsilon$ 的值在经过多次尝试后取了某合适的值进行计算，由于 Beale 函数以 $(-1.2, 1)^T$ 为初值的情况下并不收敛，因此取的值较大，得到近似解。
我们可以发现，在忽略误差的情况下，我们利用最速下降法得到了 Rosenbrock 函数的极小值点为 $(1, 1)^T$，极小值为 0，符合预期，但得到的关于 Beale 函数的结果却不符合极小值点为 $(3, 0.5)^T$ 和极小值为 0 的预期，说明此时并未收敛向最小值点。
下面我们进行牛顿法 $Newtons(f, Gradf, Hessef, x_0, epsilon)$ 得到 Rosenbrock 函数和 Beale 函数的极小值点和极小值：

函数	$epsilon$	极小值点	极小值	迭代次数
Rosenbrock	10^{-15}	$(1, 1)^T$	0	7
Beale	10^{-15}	$(0, 1)^T$	1.42×10^1	1

我们发现，利用牛顿法我们成功得到了关于 Rosenbrock 函数的最小值点，但关于 Beale 函数仅仅得到了局部最小值点，并未得到全局最小值。

Problem 4

4. Let $f(x) = \frac{1}{2}x^T x + \frac{1}{4}\sigma(x^T Ax)^2$**, where**

$$\begin{bmatrix} 5 & 1 & 0 & \frac{1}{2} \\ 1 & 4 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 3 & 0 \\ \frac{1}{2} & 0 & 0 & 2 \end{bmatrix}$$

Let (1) $x^{(0)} = (\cos 70°, \sin 70°, \cos 70°, \sin 70°)^T$**;**
(2) $x^{(0)} = (\cos 50°, \sin 50°, \cos 50°, \sin 50°)^T$
In the case of $\sigma = 1$ **and** $\sigma = 10^4$**, discuss the numerical results and behavior of convergence rate of pure Newtons method and Newtons method with line search respectively.**
解：首先我们仍然定义函数及其梯度函数和 *Hesse* 矩阵函数，这里为方便起见，我们仍然定义 f 和 g 两个函数，但是其定义中只有 *sigma* 的值发生了改变，因此我们下面只以 f 相关的函数为例进行说明：

```
function y=f(x)
sigma=1;
A=[5,1,0,0.5;1,4,0.5,0;0,0.5,3,0;0.5,0,0,2];
y=0.5*x'*x+0.25*sigma*(x'*A*x)^2;
end
```

```
function y=Gradf(x)
sigma=1;
A=[5,1,0,0.5;1,4,0.5,0;0,0.5,3,0;0.5,0,0,2];
y=x+sigma*(x'*A*x)*A*x;
end
```

```
function y=Hessef(x)
sigma=1;
A=[5,1,0,0.5;1,4,0.5,0;0,0.5,3,0;0.5,0,0,2];
y=eye(4)+2*sigma*A*x*x'*A'+sigma*sigma*(x'*A*x)*A;
end
```

Range 函数、*Minimum* 函数、*Newtons* 函数在本题仍然需要用到，但在上题中已经进行过阐述，这里不再赘述，下面是带线搜索牛顿法的函数：

```
function [x,k]=NewtonsLineSearch(f,Gradf,Hessef,x0,epsilon1,epsilon2)
k=0;x=x0;
while(k>=0)
        if(sqrt((Gradf(x))'*Gradf(x))<=epsilon1)
                break;
        end
        p=-inv(Hessef(x))*Gradf(x);
        a0=2;h0=0.3;
        [a,b] = Range(f,a0,h0,x,p);
        x=x+Minimum(f,a,b,x,p,epsilon2)*p;
```

```
        k=k+1;
    end
end
```

(1) $x^{(0)} = (\cos 70°, \sin 70°, \cos 70°, \sin 70°)^T$**;**
下面我们利用牛顿法 $Newtons(f, Gradf, Hessef, x_0, epsilon)$ 得到如下结果：

σ	$epsilon$	极小值点	极小值	迭代次数
1	10^{-12}	$(9.8\times10^{-19}, 7.0\times10^{-19}, 1.6\times10^{-19}, 1.3\times10^{-19})^T$	7.5×10^{-39}	9
10^4	10^{-12}	$(0,0,0,0)^T$	0	63766

利用带线搜索牛顿法 $NewtonsLineSearch(f, Gradf, Hessef, x_0, epsilon_1, epsilon_2)$ 得到如下结果：

σ	$epsilon_1$	$epsilon_2$	极小值点	极小值	迭代次数
1	10^{-18}	10^{-9}	$(5.4\times10^{-20}, 2.0\times10^{-20}, 7.6\times10^{-21}, 5.1\times10^{-20})^T$	3.0×10^{-39}	5
10^4	10^{-18}	10^{-9}	$(1.2\times10^{-20}, 1.0\times10^{-20}, 1.4\times10^{-21}, 1.1\times10^{-20})^T$	1.9×10^{-40}	4

我们可以发现，在合理的误差范围内，两种方法求得的极小值点和极小值相同，分别为 $(0,0,0,0)^T$ 和 0，而后者的收敛速度明显快于前者，尤其是当 σ 较大时，差异非常明显。
(2) $x^{(0)} = (\cos 50°, \sin 50°, \cos 50°, \sin 50°)^T$
下面我们利用牛顿法 $Newtons(f, Gradf, Hessef, x_0, epsilon)$ 得到如下结果：

σ	$epsilon$	极小值点	极小值	迭代次数
1	10^{-18}	$(0,0,0,0)^T$	0	10
10^4	10^{-18}	$(1.4\times10^{-25}, 9.1\times10^{-26}, 1.7\times10^{-26}, 1.9\times10^{-26})^T$	1.5×10^{-50}	64998

利用带线搜索牛顿法 $NewtonsLineSearch(f, Gradf, Hessef, x_0, epsilon_1, epsilon_2)$ 得到如下结果：

σ	$epsilon_1$	$epsilon_2$	极小值点	极小值	迭代次数
1	10^{-20}	10^{-10}	$(2.9\times10^{-21}, 1.1\times10^{-21}, 2.7\times10^{-22}, -3.6\times10^{-21})^T$	1.1×10^{-41}	5
10^4	10^{-20}	10^{-10}	$(1.3\times10^{-21}, 7.4\times10^{-22}, -2.9\times10^{-22}, 6.2\times10^{-22})^T$	1.3×10^{-42}	4

我们可以发现，当改变初值时，两种算法得到的结果仍然没有较大差异，结论与上一种初值相比时也没有发生改变，两种方法均收敛达到了最小值点，并且在较快时间内完成了解答。