Practical Optimization Method: Homework 2

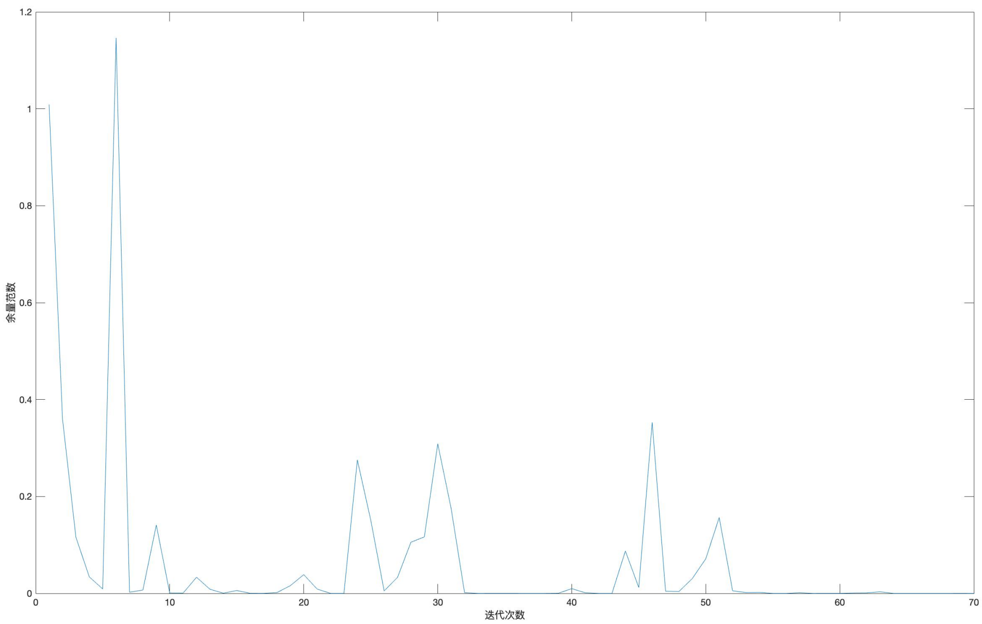
钟琦 3210103612

Question 1

利用 CG 法课件 P25 所给出的算法，设置 $A(i,j)=\frac{1}{i+j-1}$，初始值 $x_0=0$，$r_0=Ax_0-b$，$p_0=-r_0$，迭代次数 $k=0$，通过共轭梯度法迭代使 r_k 下降到 10^{-6} 以下，运行结果如下。

n	迭代次数	迭代结束时的余量
5	6	$7.25*10^{-8}$
8	19	$8.63*10^{-9}$
12	39	$7.02*10^{-7}$
20	71	$8.95*10^{-7}$

下图以 $n=20$ 为例，可以发现在 CG 算法运算过程中，随着迭代次数的增加，余量会出现较大幅度的波动，但最终趋于平稳。



Question 2&3

2. 将 $\hat{x}=Cx$ 代入 ϕ 得 $\hat{\phi}(\hat{x})=\frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})^{-1}\hat{x}-(C^{-T}b)^T\hat{x}$

即解 $(C^{-T}AC^{-1})\hat{x}=C^{-T}b$

∴原算法变为:

Given $\hat{x}_0$;

Set $\hat{r}_0 \leftarrow C^{-T}AC^{-1}\hat{x}_0-C^{-T}b$, $\hat{p}_0 \leftarrow -\hat{r}_0$, $k\leftarrow 0$

while $\hat{r}_k \neq 0$ do

$\hat{\alpha}_k \leftarrow -\frac{\hat{r}_k^T p_k}{\hat{p}_k^T C^{-T}AC^{-1}p_k}$;

$\hat{x}_{k+1} \leftarrow \hat{x}_k+\hat{\alpha}_k\hat{p}_k$;

$\hat{r}_{k+1} \leftarrow \hat{r}_k+\hat{\alpha}_k C^{-T}AC^{-1}\hat{p}_k$;

$\hat{\beta}_{k+1} \leftarrow \frac{\hat{r}_{k+1}^T\hat{r}_{k+1}}{\hat{r}_k^T\hat{r}_k}$;

$\hat{p}_{k+1} \leftarrow -\hat{r}_{k+1}+\hat{\beta}_{k+1}\hat{p}_k$;

$k\leftarrow k+1$;

End (while)

其中, $\hat{r}_0=C^{-T}AC^{-1}\hat{x}_0-C^{-T}b=C^{-T}(A(C^{-1}\hat{x}_0)-b)=C^{-T}(Ax_0-b)=C^{-T}r_0$

$\hat{p}_0=-\hat{r}_0=-C^{-T}r_0$

令 $My_0=r_0$ $\quad \hat{\alpha}_0=-\frac{(C^{-T}r_0)^T(-C^{-T}r_0)}{(C^{-1}\hat{p}_0)^T A(C^{-1}\hat{p}_0)}$

$$=\frac{r_0^T C^{-1}C^{-T}r_0}{(C^{-1}\hat{p}_0)^T A(C^{-1}\hat{p}_0)}=-\frac{r_0^T y_0}{p_0^T A p_0}$$

∵ $C^{-1}C^{-T}r_0=y_0=M^{-1}r_0$ $\Rightarrow M=C^TC$.

$C^{-1}\hat{p}_0=-C^{-1}C^{-T}r_0=p_0=-M^{-1}r_0$

且 $\hat{p}_k=Cp_k$ ∴ $\hat{\alpha}_k=-\frac{\hat{r}_k^T p_k}{\hat{p}_k^T C^{-T}AC^{-1}\hat{p}_k}=\frac{r_k^T C^{-1}C^{-T}r_k}{(C^{-1}\hat{p}_k)^T A(C^{-1}\hat{p}_k)}=\frac{r_k^T y_k}{p_k^T A p_k}$

$x_{k+1}=C^{-1}\hat{x}_{k+1}=C^{-1}(Cx_k+\hat{\alpha}_k Cp_k)=x_k+\hat{\alpha}_k p_k$

$r_{k+1}=C^T\hat{r}_{k+1}=C^T(\hat{r}_k+\hat{\alpha}_k C^{-T}AC^{-1}\hat{p}_k)=r_k+\hat{\alpha}_k Ap_k$

$\hat{\beta}_{k+1}=\frac{(C^{-T}r_{k+1})^T(C^{-T}r_{k+1})}{(C^{-T}r_k)^T(C^{-T}r_k)}=\frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$

$p_{k+1}=C^{-1}\hat{p}_k=C^{-1}(-C^{-T}r_{k+1}+\hat{\beta}_{k+1}Cp_k)=-y_{k+1}+\hat{\beta}_{k+1}p_k$

∴ 经简化后的算法为:

Given x_0, preconditioner M $(M=C^TC)$

Set $r_0 \leftarrow Ax_0 - b$

Solve $My_0 = r_0$, $p_0 \leftarrow -y_0$, $k \leftarrow 0$

while $r_k \neq 0$, do

$$\alpha_k \leftarrow -\frac{r_k^T y_k}{p_k^T A p_k};$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k;$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k;$$

Solve $My_{k+1} = r_{k+1}$;

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k};$$

$$p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k;$$

$$k \leftarrow k+1;$$

End (while)

3. 证明: $\det(B_{k+1}) = \det\left(B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi_k (s_k^T B_k s_k) v_k v_k^T\right)$

$$= \det\left(B_{k+1}^{(BFGS)} + \phi_k (s_k^T B_k s_k) v_k v_k^T\right)$$

$$= \det\left(B_{k+1}^{(BFGS)}\right) \det\left(I + \phi_k (s_k^T B_k s_k) H_{k+1}^{(BFGS)} v_k v_k^T\right)$$

下证形如 Avv^T 的矩阵为秩一矩阵 $(A, v \neq 0)$

$$vv^T = \begin{pmatrix} v_1^2 & v_1v_2 & \cdots & v_1v_n \\ v_1v_2 & v_2^2 & \cdots & v_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1v_n & v_2v_n & \cdots & v_n^2 \end{pmatrix}$$

$$Avv^T = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} v_1^2 & v_1v_2 & \cdots & v_1v_n \\ v_1v_2 & v_2^2 & \cdots & v_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1v_n & v_2v_n & \cdots & v_n^2 \end{pmatrix}$$

$$(Avv^T)_{i,j} = v_j(a_{i1}v_1 + a_{i2}v_2 + \cdots + a_{in}v_n)$$

∴ Avv^T 秩为1

∴ $H_{k+1}^{(BFGS)} v_k v_k^T$ 秩为1

$$\therefore \det\left(I + \phi_k (s_k^T B_k s_k) H_{k+1}^{(BFGS)} v_k v_k^T\right)$$
$$= 1 + \phi_k (s_k^T B_k s_k)\, v_k^T (H_{k+1}^{(BFGS)})^T v_k$$

$$\phi_k (s_k^T B_k s_k)\, v_k^T (H_{k+1}^{(BFGS)})^T v_k$$
$$= \frac{(s_k^T y_k)^2}{(s_k^T y_k)^2 - (s_k^T B_k s_k)(y_k^T H_k y_k)} (s_k^T B_k s_k)\, v_k^T \left[\left(I - \frac{s_k y_k^T}{s_k^T y_k}\right) H_k \left(I - \frac{y_k s_k^T}{s_k^T y_k}\right) + \frac{s_k s_k^T}{s_k^T y_k} \right] v_k$$

$$s_k^T v_k = s_k^T \left(\frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k} \right) = \frac{s_k^T y_k}{y_k^T s_k} - \frac{s_k^T B_k s_k}{s_k^T B_k s_k} = 1 - 1 = 0$$
$$\therefore v_k^T s_k = s_k^T v_k = 0$$

$$\therefore \phi_k (s_k^T B_k s_k)\, v_k^T (H_{k+1}^{(BFGS)})^T v_k$$
$$= \frac{(s_k^T y_k)^2}{(s_k^T y_k)^2 - (s_k^T B_k s_k)(y_k^T H_k y_k)} (s_k^T B_k s_k)\, v_k^T H_k v_k$$
$$= \frac{(s_k^T y_k)^2}{(s_k^T y_k)^2 - (s_k^T B_k s_k)(y_k^T H_k y_k)} (s_k^T B_k s_k) \left(\frac{y_k^T}{y_k^T s_k} - \frac{s_k^T B_k}{s_k^T B_k s_k} \right) H_k \left(\frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k} \right)$$
$$= \frac{(s_k^T y_k)^2}{(s_k^T y_k)^2 - (s_k^T B_k s_k)(y_k^T H_k y_k)} (s_k^T B_k s_k) \cdot \frac{(s_k^T B_k s_k)(y_k^T H_k y_k) - (y_k^T s_k)^2}{(y_k^T s_k)^2 (s_k^T B_k s_k)}$$
$$= -1$$

$$\therefore \det\left(I + \phi_k (s_k^T B_k s_k) H_{k+1}^{(BFGS)} v_k v_k^T\right) = 1 - 1 = 0$$
$$\therefore \det(B_{k+1}) = 0$$

$\therefore B_{k+1}$ 是奇异矩阵.

Question 4

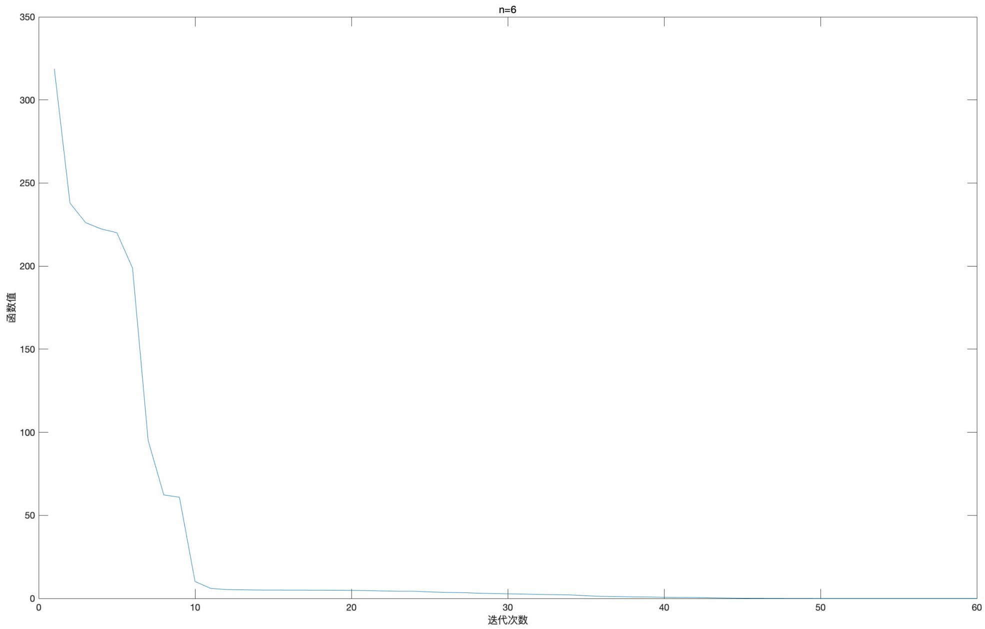
利用拟牛顿法课件 P23 所给出的算法，依次求解 Rosenbrock 函数和 Powell singular 函数求最小值，两者除了给定函数和初值不同外，其余算法流程均相同。

算法细节方面，对于 α 的选取，本代码中采用了满足 Wolfe 条件的不精确求解，设置 $c_1 = 10^{-4}$，$c_2 = 0.9$，在不满足 $f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k (\nabla f(x_k))^T p_k$ 时进行 $\alpha = \alpha * 0.5$ 的操作，若已满足，在 $(\nabla f(x_k + \alpha_k p_k))^T p_k \ge c_2 (\nabla f(x_k))^T p_k$ 时采用对前一取值和后一取值取平均的方式得到迭代的 α_k，再次进入上述判断，直至满足 Wolfe 条件，运行结果如下，可以看到，在拟牛顿法 BFGS 迭代过程中，函数值持续下降，且存在下降速率突然增加的拐点。具体代码见于附录。

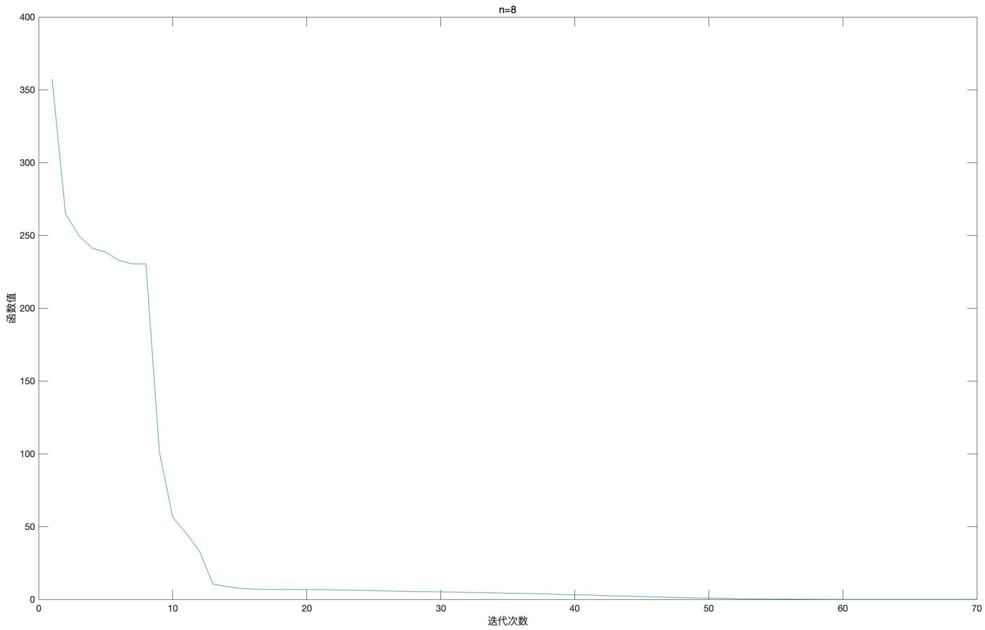
4.1 Rosenbrock 函数

n	迭代次数	x^*	$f(x^*)$
6	60	$(1,1,1,1,1,1)^T$	$2.48 * 10^{-17}$
8	70	$(1,1,1,1,1,1,1,1)^T$	$6.96 * 10^{-15}$
10	81	$(1,1,1,1,1,1,1,1,1,1)^T$	$2.48 * 10^{-15}$

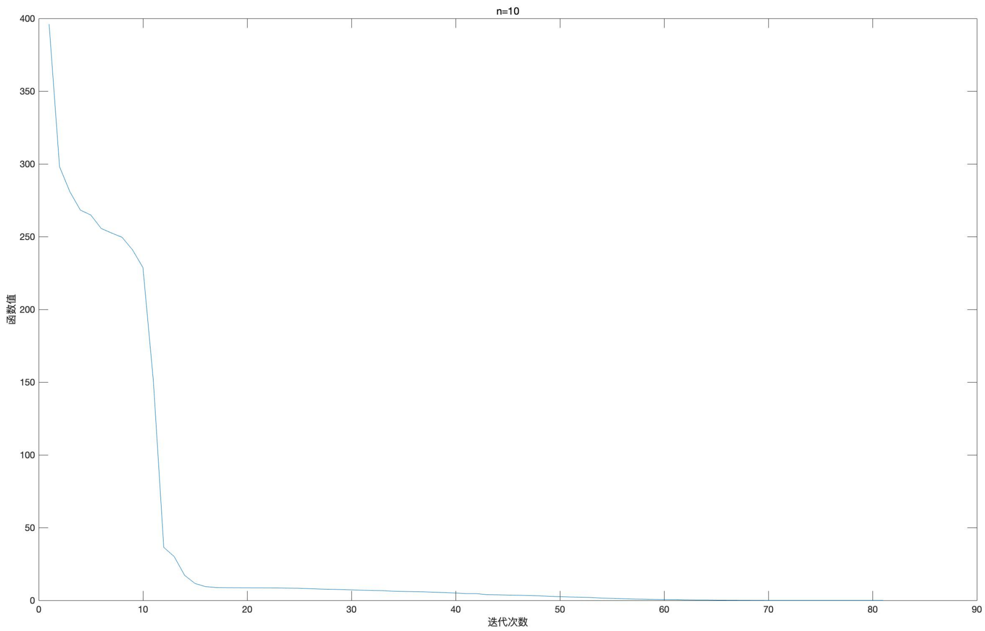
4.1.1 n=6



4.1.2 n=8



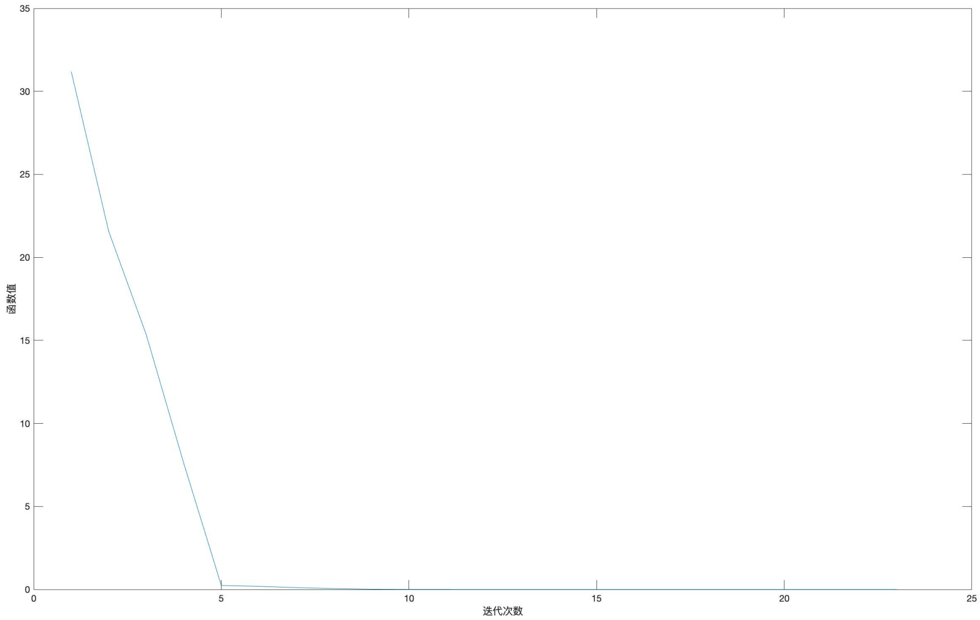
4.1.3 n=10



4.2 Powell singular 函数

改变函数值，其余结构均不变，运行结果如下：

迭代次数	x^*	$f(x^*)$
23	$(0.0021, -0.0002, 0.0034, 0.0034)^T$	$2.48 * 10^{-9}$



A 附录

A.1 Question 1 CG 法代码

```
#include <iostream>
#include <Eigen/Dense>

using namespace std;
using namespace Eigen;

void CG(MatrixXd A, VectorXd b, VectorXd& x, double e){
    VectorXd r=A*x-b,r_pre;
    VectorXd p=-1.0*r;
    int iter=0;
    while(iter<=100000&&r.norm()>=e){
        double alpha=-1.0*(r.dot(p))/(p.dot(A*p));
        x=x+alpha*p;
        r_pre=r;
        r+=alpha*A*p;
        cout<<iter<<":"<<r.norm()<<endl;
```

```
        double beta=r.dot(r)/r_pre.dot(r_pre);
        p=-1*r+beta*p;
        iter++;
    }
    cout<<"total iteration times:"<<iter<<endl;
}

int main() {
    int size=20;
    double e=1e-6;
    MatrixXd A(size,size);
    VectorXd b(size);
    for(int i=0;i<size;i++){
        for(int j=0;j<size;j++){
            A(i,j)=1.0/(i+j+1);
        }
        b(i)=1;
    }

    VectorXd x = VectorXd::Zero(size);

    CG(A,b,x,e);
    std::cout << "Solution x:\n" << x << std::endl;
    return 0;
}
```