【Experiment name】

<u>Conjugate gradient method to solve Rosen brock function</u>

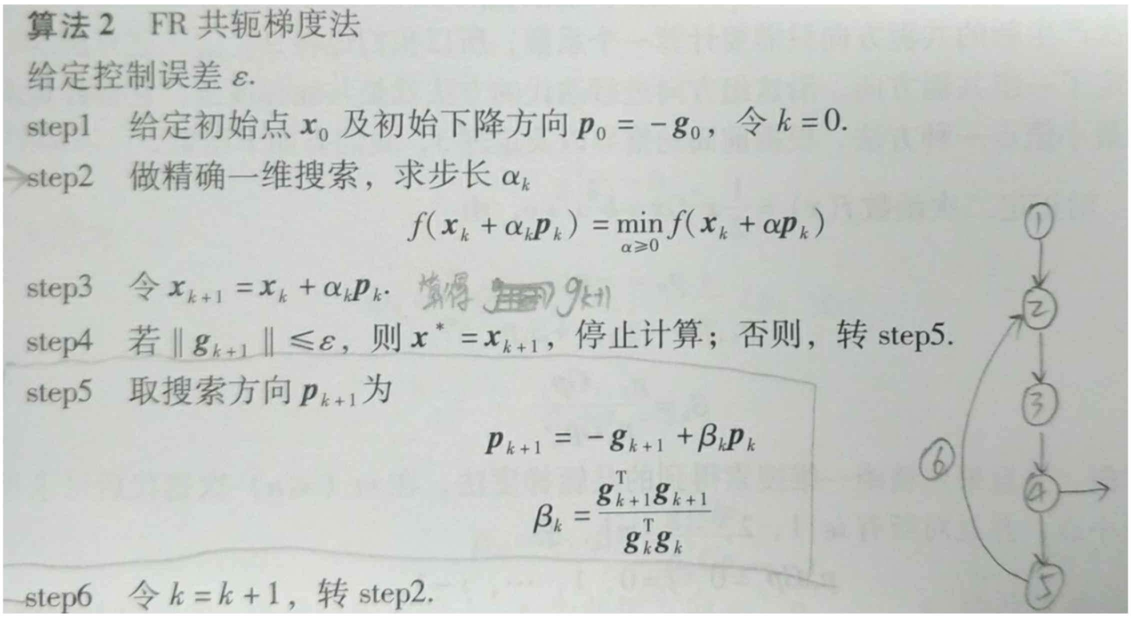
【Purpose】

1.Master the solution of Rosenbrock function by FR conjugate gradient method and PRP conjugate gradient method respectively.

2. Master and call the precise one - dimensional search golden section method

【Experimental principle】

F R Conjugate Gradient Algorithm



算法 2 FR 共轭梯度法

给定控制误差 ε.

step1 给定初始点 $\boldsymbol{x}_0$ 及初始下降方向 $\boldsymbol{p}_0 = -\boldsymbol{g}_0$，令 $k=0$.

step2 做精确一维搜索，求步长 α_k

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) = \min_{\alpha \geqslant 0} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)$$

step3 令 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$. 懒得 g_{k+1}

step4 若 $\|\boldsymbol{g}_{k+1}\| \leqslant \varepsilon$，则 $\boldsymbol{x}^* = \boldsymbol{x}_{k+1}$，停止计算；否则，转 step5.

step5 取搜索方向 $\boldsymbol{p}_{k+1}$ 为

$$\boldsymbol{p}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{p}_k$$

$$\beta_k = \frac{\boldsymbol{g}_{k+1}^{\mathrm{T}} \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^{\mathrm{T}} \boldsymbol{g}_k}$$

step6 令 $k = k+1$，转 step2.

PRP Conjugate Gradient Method Algorithm

算法 3　PRP 共轭梯度法

在算法 2 的 step 5 中，用 PRP 公式

$$\beta_k = \frac{g_{k+1}^{T}(g_{k+1} - g_k)}{g_k^{T} g_k}$$

代替 FR 公式，就得到 PRP 共轭梯度法．

对于正定的二次函数，FR 共轭梯度法与 PRP 共轭梯度法等价．但是对于一般函数，二者是不同的，因为对于目标函数的 Hesse 阵不是常数矩阵，所以迭代过程中所产生的方向不再是共轭方向了．不过若采用精确一维搜索，两个算法所产生的搜索方向都满足

$$p_k^{T} g_k = (-g_k + \beta_{k-1} p_{k-1})^{T} g_k = -g_k^{T} g_k < 0$$

故两者都是下降算法．

PRP 算法与 FR 算法都是常用的共轭梯度法．从一些实际计算的结果发现，PRP 算法一般优于 FR 算法．

Example 1:

例 3　用 FR 共轭梯度法求解 Rosenbrock 函数

$$\min f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

问题有唯一极小点，精确解为 $x^* = (1,\ 1)^T$，$f\ (x^*) = 0$.

Example 2:

例 5　利用 PRP 共轭梯度法求解 Rosenbrock 函数

$$\min f(x) = 100\ (x_2 - x_1^2)^2 + (1 - x_1)^2$$

问题有唯一极小点，精确解为 $x^* = (1,\ 1)^T$，$f(x^*) = 0$.

【Experimental content】

I use the matlab software to write the experimental code, and successfully demonstrate the algorithm process of the two conjugate gradient methods and print them.

FR method experimental results: (epsilon takes 1 e-6)

The golden section takes the interval [0,5]				
k	x1	x2	f(x1,x2)	||gk||^2
1	0.999932	1.000000	0.000002	0.061098
2	0.999986	0.999973	0.000000	0.000121
3	0.999987	0.999973	0.000000	0.000012
4	0.999987	0.999973	0.000000	0.000052
5	0.999998	0.999995	0.000000	0.000249
6	1.000001	1.000001	0.000000	0.000010
7	1.000001	1.000001	0.000000	0.000001

The golden section takes the interval [0,0.1]				
k	x1	x2	f(x1,x2)	||gk||^2
1	0.999932	1.000000	0.000002	0.061098
2	0.999936	0.999872	0.000000	0.000139
3	0.999936	0.999873	0.000000	0.000082
4	0.999941	0.999883	0.000000	0.000197
5	0.999943	0.999885	0.000000	0.000264
6	0.999944	0.999887	0.000000	0.000529
7	1.000001	1.000001	0.000000	0.000341
8	1.000000	1.000000	0.000000	0.000004
9	1.000000	1.000000	0.000000	0.000004
10	1.000000	1.000000	0.000000	0.000000

P RP experiment results: (epsilon takes 1 e-6)

The golden section takes the interval [0,5]				
k	x1	x2	f(x1,x2)	||gk||^2
1	0.999932	1.000000	0.000002	0.061098
2	0.999936	0.999872	0.000000	0.000150
3	0.999936	0.999872	0.000000	0.000057
4	0.999990	0.999979	0.000000	0.001059
5	1.000000	1.000000	0.000000	0.000006
6	1.000000	1.000000	0.000000	0.000006
7	1.000000	1.000000	0.000000	0.000000

The golden section takes the interval [0,0.1]				
k	x1	x2	f(x1,x2)	||gk||^2
1	0.999932	1.000000	0.000002	0.061098
2	0.999936	0.999872	0.000000	0.000139
3	0.999936	0.999873	0.000000	0.000082
4	0.999941	0.999883	0.000000	0.000197
5	0.999943	0.999885	0.000000	0.000264
6	0.999944	0.999887	0.000000	0.000529
7	1.000001	1.000001	0.000000	0.000341
8	1.000000	1.000000	0.000000	0.000004
9	1.000000	1.000000	0.000000	0.000004
10	1.000000	1.000000	0.000000	0.000000

Other function test cases (eg: f=(1/2)*x1^2+(9/2)*x2^2; x0=[9 1];)

FR method: (epsilon takes 1 e-4)

The golden section takes the interval [0,0.1]				
k	x1	x2	f(x1,x2)	||gk||^2
1	7.199976	-0.800024	28.800000	10.182475

2	5.904438	-0.656071	19.368123	8.350277
3	4.442956	0.030986	9.874249	4.451699
4	3.583445	0.198382	6.597640	4.003607
5	2.530044	0.155297	3.309089	2.890434
6	1.728076	-0.006876	1.493336	1.729184
7	1.268311	-0.058731	0.819828	1.374047
8	0.851220	-0.038635	0.369004	0.919500
9	0.579349	0.005123	0.167941	0.581181
10	0.412822	0.017995	0.086668	0.443455
11	0.274602	0.009300	0.038092	0.287074
12	0.189228	-0.002711	0.017937	0.190794
13	0.132601	-0.005577	0.008931	0.141784
14	0.088074	-0.002142	0.003899	0.090160
15	0.061265	0.001174	0.001883	0.062169
16	0.042394	0.001695	0.000912	0.045054
17	0.028245	0.000443	0.000400	0.028526
18	0.019750	-0.000457	0.000196	0.020174
19	0.013527	-0.000496	0.000093	0.014245
20	0.009072	-0.000069	0.000041	0.009093
21	0.006349	0.000167	0.000020	0.006525
22	0.004313	0.000138	0.000009	0.004489
23	0.002918	0.000000	0.000004	0.002918
24	0.002037	-0.000058	0.000002	0.002103
25	0.001375	-0.000036	0.000001	0.001414
26	0.000939	0.000006	0.000000	0.000941
27	0.000652	0.000019	0.000000	0.000675
28	0.000439	0.000009	0.000000	0.000446
29	0.000302	-0.000004	0.000000	0.000304
30	0.000208	-0.000006	0.000000	0.000216
31	0.000140	-0.000002	0.000000	0.000141
32	0.000097	0.000002	0.000000	0.000098

PRP method: (epsilon takes 1 e-4)

The golden section takes the interval [0,0.3]				
k	x1	x2	f(x1,x2)	||gk||^2
1	7.199976	-0.800024	28.800000	10.182475
2	3.312444	-0.368102	6.095891	4.684841
3	3.300493	0.011568	5.447229	3.302135
4	2.310082	-0.006886	2.668452	2.310913
5	1.824031	0.015558	1.664633	1.829397
6	1.355016	-0.030050	0.922097	1.381741
7	1.022075	0.058221	0.537572	1.148565

8	0.702244	-0.095441	0.287564	1.109490
9	0.258053	0.050006	0.044548	0.518784
10	0.047801	-0.016743	0.002404	0.158088
11	-0.007284	-0.003326	0.000076	0.030810
12	0.000002	-0.000000	0.000000	0.000004

Attachment: Experimental code (4 files in total)

test.m

```
% Conjugate gradient method test area

% define several functions
syms x1 x2;
x={x1,x2};
% function f
f = 100 * (x2 - x1^2)^2 + (1 - x1)^2;
% initial point x0
x0=[-1 1];
% set epsilon
e=1e-6;

% function call
[result_x] = FR(f, x, x0, e); % FR method test
%[result_x] = PRP(f, x, x0, e); % PRP method test

% other test data
%f=(3/2)*x1^2+(1/2)*x2^2-x1*x2-2*x1;
%x0=[0 0];
%f=(1/2)*x1^2+(9/2)*x2^2;
%x0=[9 1];
```

gold.m

```
function [result] = gold(f, a, b)
% golden section method
%Input f: function a, b: [a, b] operation interval

% Default epsilon=1e-4 (can be adjusted below)
syms x x2 f1 f2 xx ah;
flag=1; ratio=0.618; e=1e-4;%epsilon
for i = 1:100
%step1:
if flag == 1
x1 = a + (1-ratio) * (b - a);
f1 = subs(f, ah, x1);
x2 = a + ratio * (b - a);
```

```
f2 = subs(f, ah, x2);

flag = 2;
end
%step2
if flag == 2
if (abs(b - a) <= e)
xx = (a + b) / 2;
break;
else
flag = 3;
end
end

%step3
if flag == 3
if f1 < f2
b = x2;
x2 = x1;
f2 = f1;
x1 = a + 0.382 * (b - a);
f1 = subs(f, ah, x1);

flag = 2; % flag update

elseif f1 == f2
a = x1;
b = x2;

flag = 1; % flag update

elseif f1 > f2
a = x1;
x1 = x2;
f1 = f2;
x2 = a + 0.618 * (b - a);
f2 = subs(f, ah, x2);

flag = 2; % flag update
end
end
end
result = xx;
end
```

F _

```
function [result_x] = FR(f,x,x0,e)
% Using the "FR conjugate method" of the exact one-dimensional search golden section method
% Input f: complete function x: independent variable set x0: initial point e: epsilon
% output result_x: exact solution approximation result x*

% Define the alpha used in the step size; k is used for counting
syms ah;
k=0;
% Create an n-dimensional cell array (only x1, x2 in the example, ie n=2)
n = length(x);
gx = cell(1, n);
for i = 1 : n
gx{i} = diff(f, x{i});
end
% data initialization
g = subs(gx, x, x0); % partial derivative solution g(x)
g_pre = g;
xx = x0; % Substitute xx for x(i+1) and x_pre for x(i) in the later stage
x_pre=x0;
p = - g; % p0 reverse direction
% first iteration
xx = xx+ah*p;
phi = subs(f, x, xx);
nphi = double(gold(phi, 0, 2));
aa = nphi;
xx = double(x_pre+aa*p);
g = subs(gx, x, xx);
% Data output format [k x1 x2 f(x1,x2) ||gk||^2]
fprintf('k\t\tx1\t\t\tx2\t\t\tf(x1,x2)\t||gk||^2\n');

while (norm(g) > e)
% FR core algorithm step5
beta=(sumsqr(g)/sumsqr(g_pre));
p= - g+beta*p;

% Temporary data update
g_pre=g;
x_pre=xx;
k=k+1;
% data output
ff=double(subs(f, x, xx)); % f(x1,x2)
```

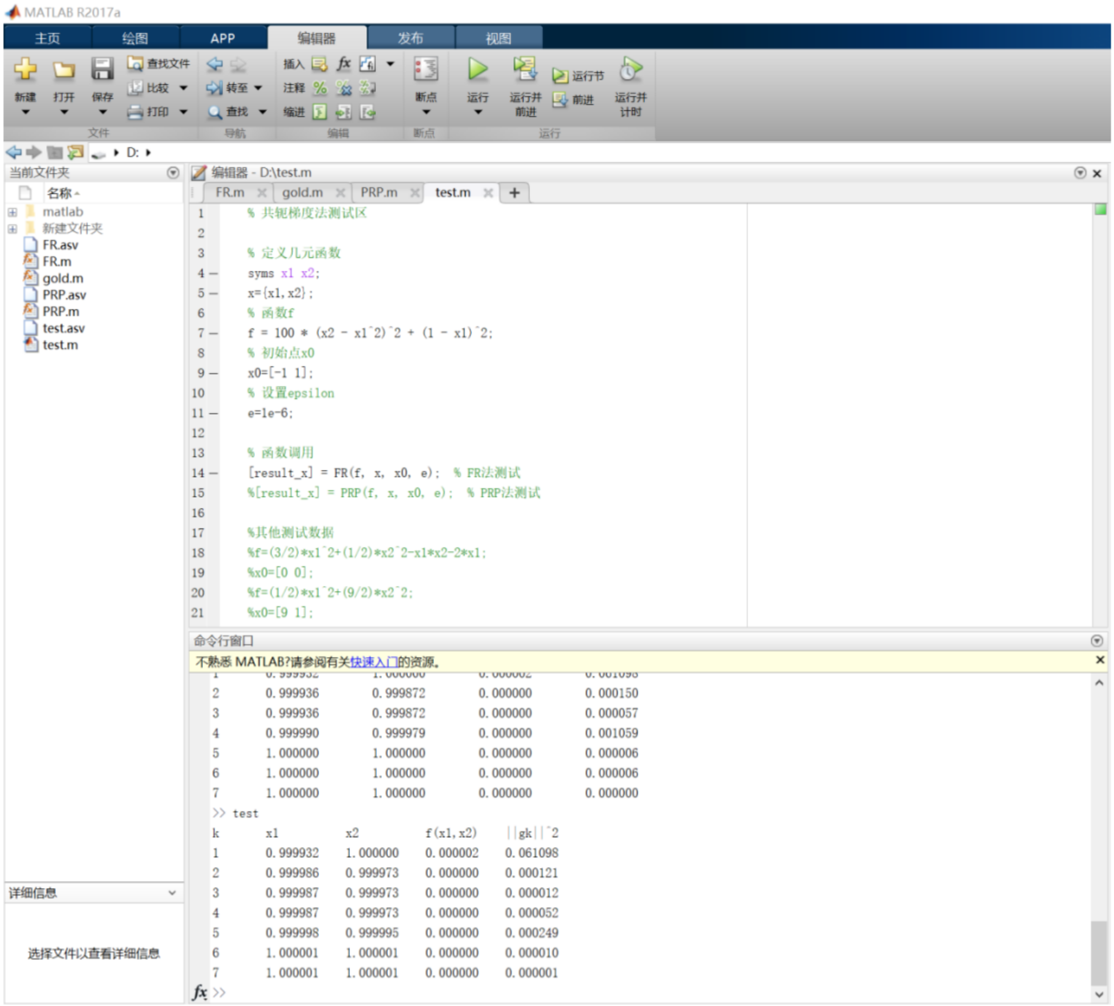
```
g2=double(norm(g)); % ||gk||^2
fprintf('%d\t\t',k);fprintf('%f\t',xx);fprintf('%f\t%f\n',ff, g2);

% (k-1)th iteration
xx = xx+ah*p;
phi = subs(f, x, xx);
nphi = double(gold(phi, 0, 5)); % golden section search 5
%nphi = double(gold(phi, 0, 0.1)); % Golden section search 0.1
aa = nphi;
xx = double(x_pre+aa*p);
g = subs(gx, x, xx);

end

% Last line of data output
ff=double(subs(f, x, xx));
g2=double(norm(g));
fprintf('%d\t\t',k+1);fprintf('%f\t',xx);fprintf('%f\t%f\n',ff, g2);
%The function returns the solution result x*
result_x=xx;
end
```

P RP.m

```
function [result_x] = PRP(f,x,x0,e)
% Using the "FR conjugate method" of the exact one-dimensional search golden section method
% Input f: complete function x: independent variable set x0: initial point e: epsilon
% output result_x: exact solution approximation result x*

% Define the alpha used in the step size; k is used for counting
syms ah;
k=0;
% Create an n-dimensional cell array (only x1, x2 in the example, ie n=2)
n = length(x);
gx = cell(1, n);
for i = 1 : n
gx{i} = diff(f, x{i});
end
% data initialization
g = subs(gx, x, x0); % partial derivative solution g(x)
g_pre = g;
xx = x0; % Substitute xx for x(i+1) and x_pre for x(i) in the later stage
x_pre=x0;
p = - g; % p0 reverse direction
```

```
% first iteration
xx = xx+ah*p;
phi = subs(f, x, xx);
nphi = double(gold(phi, 0, 2));
aa = nphi;
xx = double(x_pre+aa*p);
g = subs(gx, x, xx);
% Data output format [k x1 x2 f(x1,x2) ||gk||^2]
fprintf('k\t\tx1\t\t\tx2\t\t\tf(x1,x2)\t||gk||^2\n');

while (norm(g) > e)
% PRP core algorithm step5
g1=double((g-g_pre)*(g'));
g2=double(g_pre*(g_pre'));
beta=(g1/g2);
p= - g+beta*p;

% Temporary data update
g_pre=g;
x_pre=xx;
k=k+1;
% data output
ff=double(subs(f, x, xx)); % f(x1,x2)
g2=double(norm(g)); % ||gk||^2
fprintf('%d\t\t',k);fprintf('%f\t',xx);fprintf('%f\t%f\n',ff, g2);

% (k-1)th iteration
xx = xx+ah*p;
phi = subs(f, x, xx);
nphi = double(gold(phi, 0, 5)); % golden section search
%nphi = double(gold(phi, 0, 0.1)); % golden section search
aa = nphi;
xx = double(x_pre+aa*p);
g = subs(gx, x, xx);

end

% Last line of data output
ff=double(subs(f, x, xx));
g2=double(norm(g));
fprintf('%d\t\t',k+1);fprintf('%f\t',xx);fprintf('%f\t%f\n',ff, g2);
%The function returns the solution result x*
result_x=xx;
end
```

M a tlab experiment interface;



【Experiment Summary】

Through this experiment, I have a better understanding of the use of the FR conjugate gradient method and the PRP conjugate gradient method in the unconstrained optimization numerical algorithm. I have clarified the algorithm idea by writing the algorithm code, and I have observed and understood the numerical optimization through specific experiments. The process, and the golden section one-dimensional search algorithm is well applied to it, which deepens the learning and understanding of our "Optimization Method" course.