

1. 雅可比迭代法

取x(0)=(0,0,0)T

迭代得

x(1)=(-2.4,5,0.3)T

x(2)=(-4.46,4.25,2.28)T

x(3)=(-4.556,2.745,2.467)T

x(4)=(-3.9914,2.6275,2.0347)T

x(5)=(-3.85794,2.9848,1.88653)T

x(6)=(-3.97123,3.09225,1.96703)T

x(7)=(-4.03031,3.02368,2.02192)T

x(8)=(-4.01386,2.98146,2.01316)T

x(9)=(-3.99522,2.98995,1.99721)T

x(10)=(-3.99542,3.00259,1.99603)T

x(11)=(-4.00024,3.00313,1.99986)T

x(12)=(-4.00122,3.00001,2.00099)T

x(13)=(-4.0002,2.9992,2.00025)T

x(14)=(-3.99973,2.99983,1.9998)T

x(15)=(-3.99989,3.00017,1.99989)T

x(16)=(-4.00005,3.00008,2.00003)T

x(17)=(-4.00004,2.99997,2.00003)T

x(18)=(-4,2.99997,2)T

此时满足 ||x(k+1)-x(k)||∞<10-4

1. 高斯-赛德尔迭代法

取x(0)=(0,0,0)T

迭代得

x(1)=(-2.4,4.4,2.1)T

x(2)=(-4.58,2.805,2.0575)T

x(3)=(-3.9335,2.98788,1.98306)T

x(4)=(-3.99176,3.01053,2.00151)T

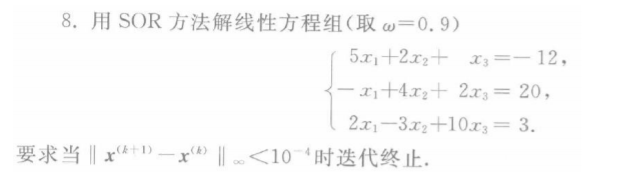
x(5)=(-4.00451,2.99812,2.00034)T

x(6)=(-3.99931,3,1.99986)T

x(7)=(-3.99997,3.00007,2.00002)T

x(8)=(-4.00003,2.99998,2)T

此时满足 ||x(k+1)-x(k)||∞<10-4



取，x(0)=(0,0,0)T

迭代得

x(1)=(-2.16,4.014,1.74258)T

x(2)=(-4.1347,3.18693,2.04898)T

x(3)=(-4.08958,2.9765,2.01468)T

x(4)=(-4.00314,2.99034,1.99942)T

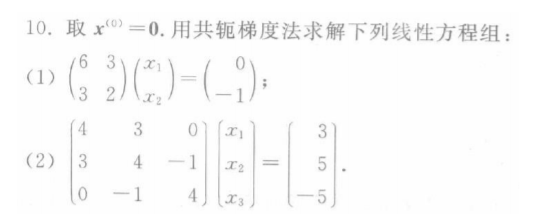
x(5)=(-3.99673,3.00003,1.99936)T

x(6)=(-3.99957,3.00039,1.99996)T

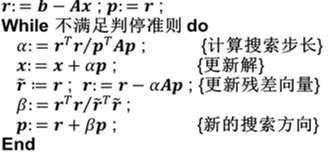
x(7)=(-4.00009,3.00004,2.00002)T

x(8)=(-4.00003,2.99999,2)T

此时满足 ||x(k+1)-x(k)||∞<10-4

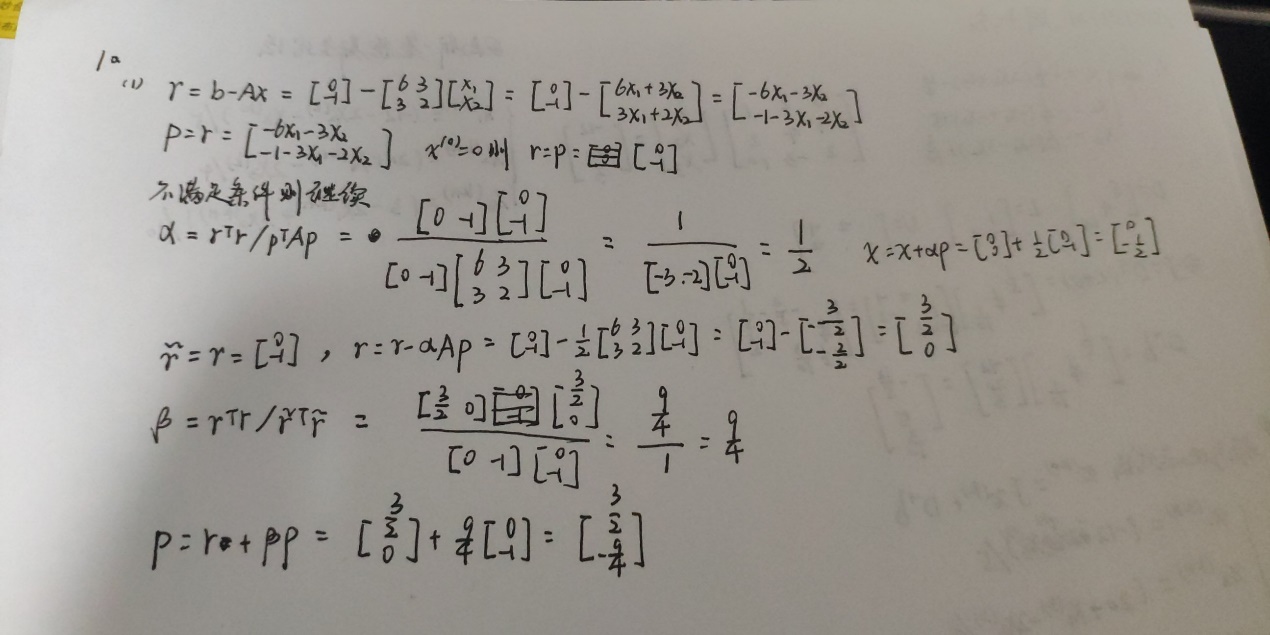


迭代过程如图：



（1）

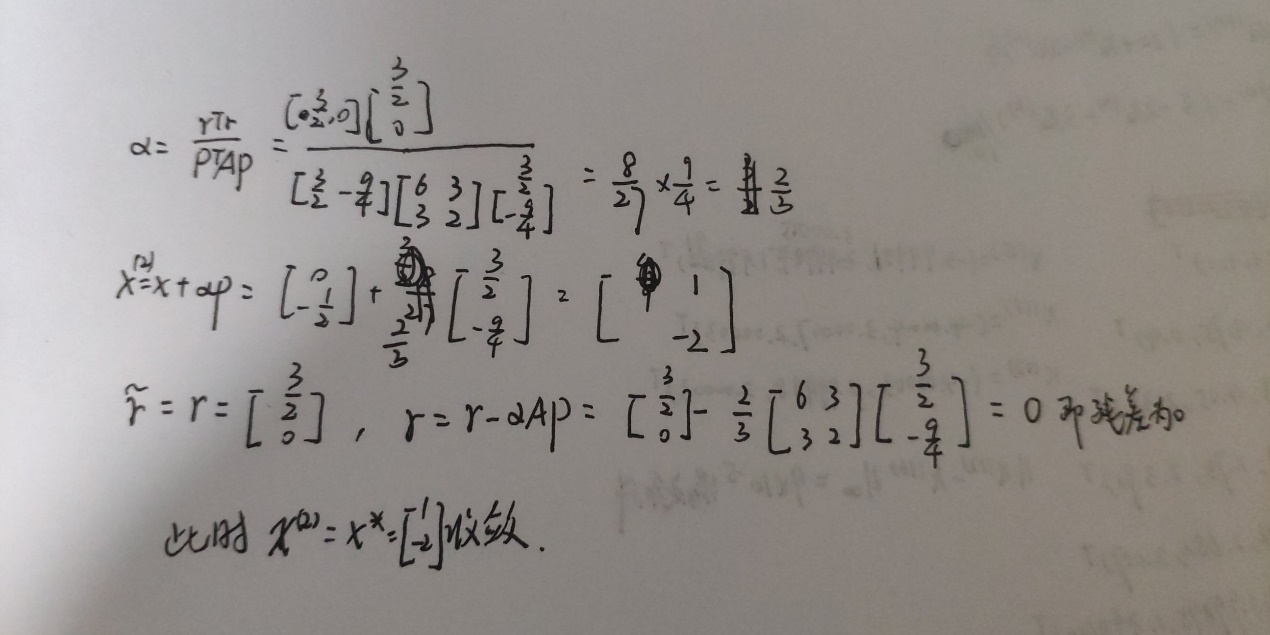
计算x(1):



得到x(1)=(0,-0.5)T

此时残差向量不为零，则继续

计算x(2):



得到x(2)=（1，-2）T

此时残差向量为0，x(2)为所求解。

（2）

x(0)=(0,0,0)T

第1次迭代结果：

Alpha=0.156915

x(1)=(0.470745,0.784574,-0.784574)T

r~=(3,5,-5)T

r=(-1.2367,-0.335106,-1.07713)T

Beta=0.0474904

p=(-1.09423,-0.0976545,-1.31458)T

第2次迭代结果：

Alpha=0.231099

x(2)=(0.217869,0.762007,-1.08837)T

r~=(-1.2367,-0.335106,-1.07713)T

r=(-0.157496,0.209994,0.115497)T

Beta=0.0293519

p=(-0.189613,0.207128,0.0769114)T

第3次迭代结果：

Alpha=1.14902

x(3)=(-5.55112e-016,1,-1)T

r~=(-0.157496,0.209994,0.115497)T

r=(1.16573e-015,1.27676e-015,-8.32667e-016)T

Beta=4.47749e-029

p=(1.16573e-015,1.27676e-015,-8.32667e-016)T

附程序：

1. #include <iostream>
2. #include <iomanip>
3. #include <cmath>
4. **using** **namespace** std;
5. **int** main()
6. {
7. **const** **int** a[9]={4,3,0,3,4,-1,0,-1,4};
8. **const** **int** b[3]={3,5,-5};
9. **double** x[3]={0,0,0};
11. **double** r[3]={3,5,-5};
12. **double** p[3]={3,5,-5};
13. **double** alpha=0;
14. **double** r2[3]={0};
15. **double** beta=0;
17. **int** n=0;
19. **while**(fabs(r[0])>=0.0001||fabs(r[1])>=0.0001||fabs(r[2])>=0.0001){
21. //计算alpha （计算搜索步长）
22. alpha=1.0\*(r[0]\*r[0]+r[1]\*r[1]+r[2]\*r[2])/
23. ((p[0]\*a[0]+p[1]\*a[3]+p[2]\*a[6])\*p[0]+
24. (p[0]\*a[1]+p[1]\*a[4]+p[2]\*a[7])\*p[1]+
25. (p[0]\*a[2]+p[1]\*a[5]+p[2]\*a[8])\*p[2]);
27. //计算x (更新解)
28. **for**(**int** i=0;i<3;i++)
29. x[i]=x[i]+alpha\*p[i];
31. //计算r2
32. **for**(**int** i=0;i<3;i++)
33. r2[i]=r[i];
35. //计算r（更新残差向量）
36. r[0]=r[0]-alpha\*(a[0]\*p[0]+a[1]\*p[1]+a[2]\*p[2]);
37. r[1]=r[1]-alpha\*(a[3]\*p[0]+a[4]\*p[1]+a[5]\*p[2]);
38. r[2]=r[2]-alpha\*(a[6]\*p[0]+a[7]\*p[1]+a[8]\*p[2]);
40. //计算beta
41. beta=1.0\*(r[0]\*r[0]+r[1]\*r[1]+r[2]\*r[2])/
42. (r2[0]\*r2[0]+r2[1]\*r2[1]+r2[2]\*r2[2]);
44. //计算p（新的搜索方向）
45. **for**(**int** i=0;i<3;i++)
46. p[i]=r[i]+beta\*p[i];
48. n++;
50. //输出
51. cout<<"第"<<n<<"次迭代结果："<<endl;
52. cout<<"Alpha="<<alpha<<endl;
53. cout<<"x("<<n<<")=("<<x[0]<<","<<x[1]<<","<<x[2]<<")T"<<endl;
54. cout<<"r~=("<<r2[0]<<","<<r2[1]<<","<<r2[2]<<")T"<<endl;
55. cout<<"r=("<<r[0]<<","<<r[1]<<","<<r[2]<<")T"<<endl;
56. cout<<"Beta="<<beta<<endl;
57. cout<<"p=("<<p[0]<<","<<p[1]<<","<<p[2]<<")T"<<endl;
58. cout<<endl;
59. }
60. }