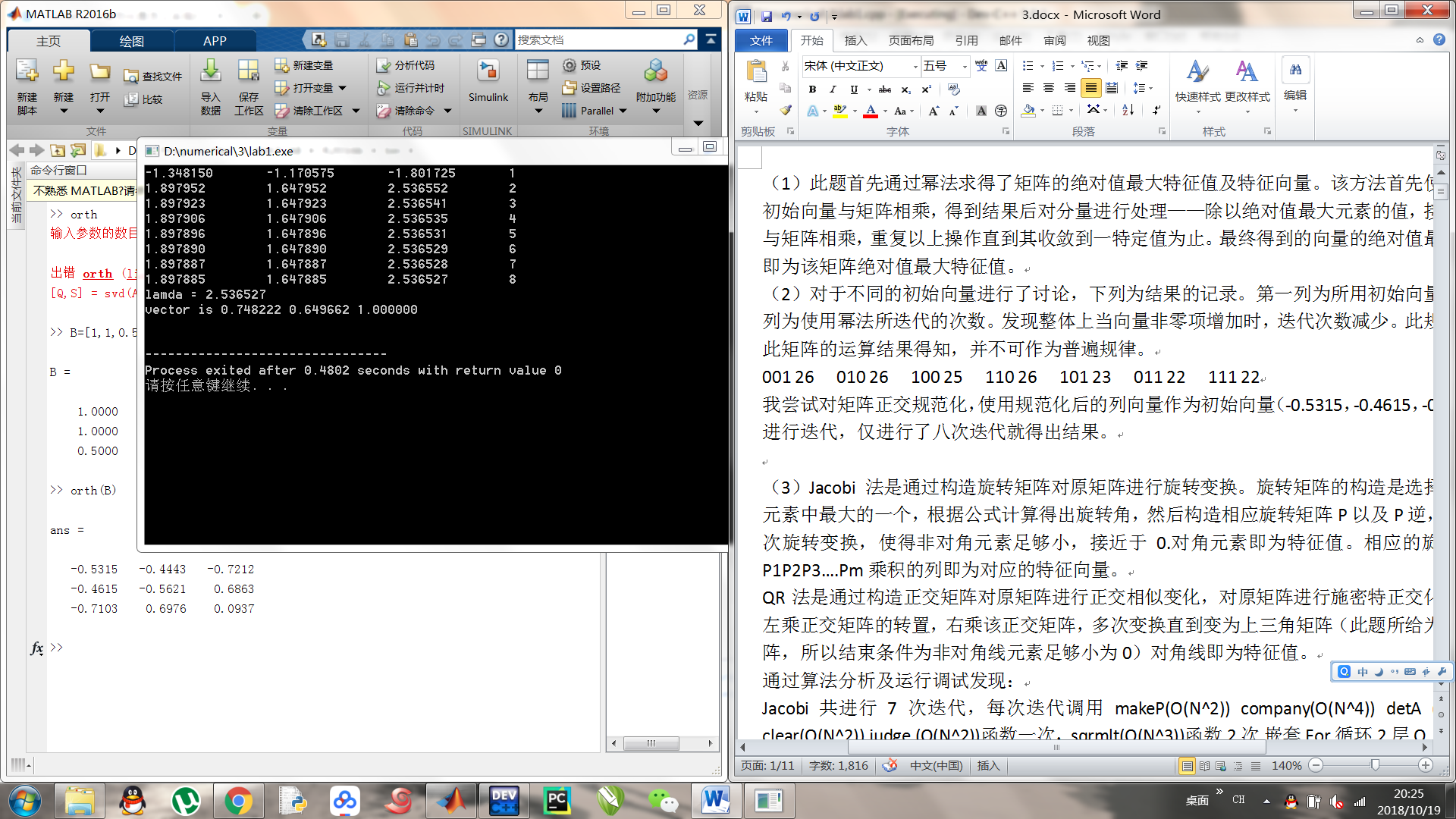
（1）此题首先通过幂法求得了矩阵的绝对值最大特征值及特征向量。该方法首先使用一个初始向量与矩阵相乘，得到结果后对分量进行处理——除以绝对值最大元素的值，接着继续与矩阵相乘，重复以上操作直到其收敛到一特定值为止。最终得到的向量的绝对值最大元素即为该矩阵绝对值最大特征值。

（2）对于不同的初始向量进行了讨论，下列为结果的记录。第一列为所用初始向量，第二列为使用幂法所迭代的次数。发现整体上当向量非零项增加时，迭代次数减少。此规律仅从此矩阵的运算结果得知，并不可作为普遍规律。

001 26 010 26 100 25 110 26 101 23 011 22 111 22

我尝试对矩阵正交规范化，使用规范化后的列向量作为初始向量（-0.5315，-0.4615，-0.7103）进行迭代，仅进行了八次迭代就得出结果。如图。所以，在条件允许的情况下可先将矩阵正交规范化化，选取适当列向量即可迭代更快。



（3）Jacobi法是通过构造旋转矩阵对原矩阵进行旋转变换。旋转矩阵的构造是选择非对角元素中最大的一个，根据公式计算得出旋转角，然后构造相应旋转矩阵P以及P逆，通过多次旋转变换，使得非对角元素足够小，接近于0.对角元素即为特征值。相应的旋转矩阵P1P2P3….Pm乘积的列即为对应的特征向量。

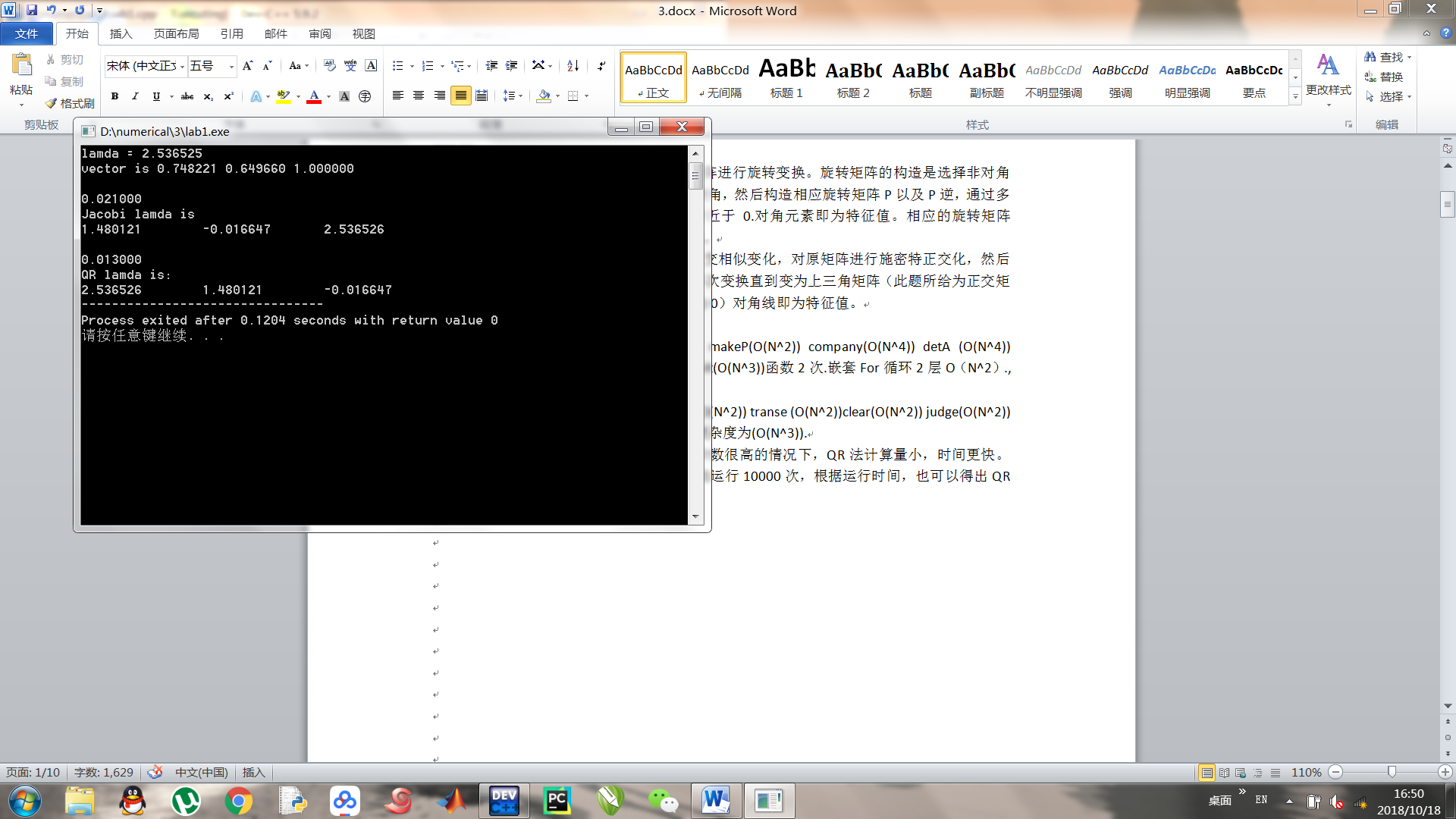
QR法是通过构造正交矩阵对原矩阵进行正交相似变化，对原矩阵进行施密特正交化，然后左乘正交矩阵的转置，右乘该正交矩阵，多次变换直到变为上三角矩阵（此题所给为正交矩阵，所以结束条件为非对角线元素足够小为0）对角线即为特征值。

通过算法分析及运行调试发现：

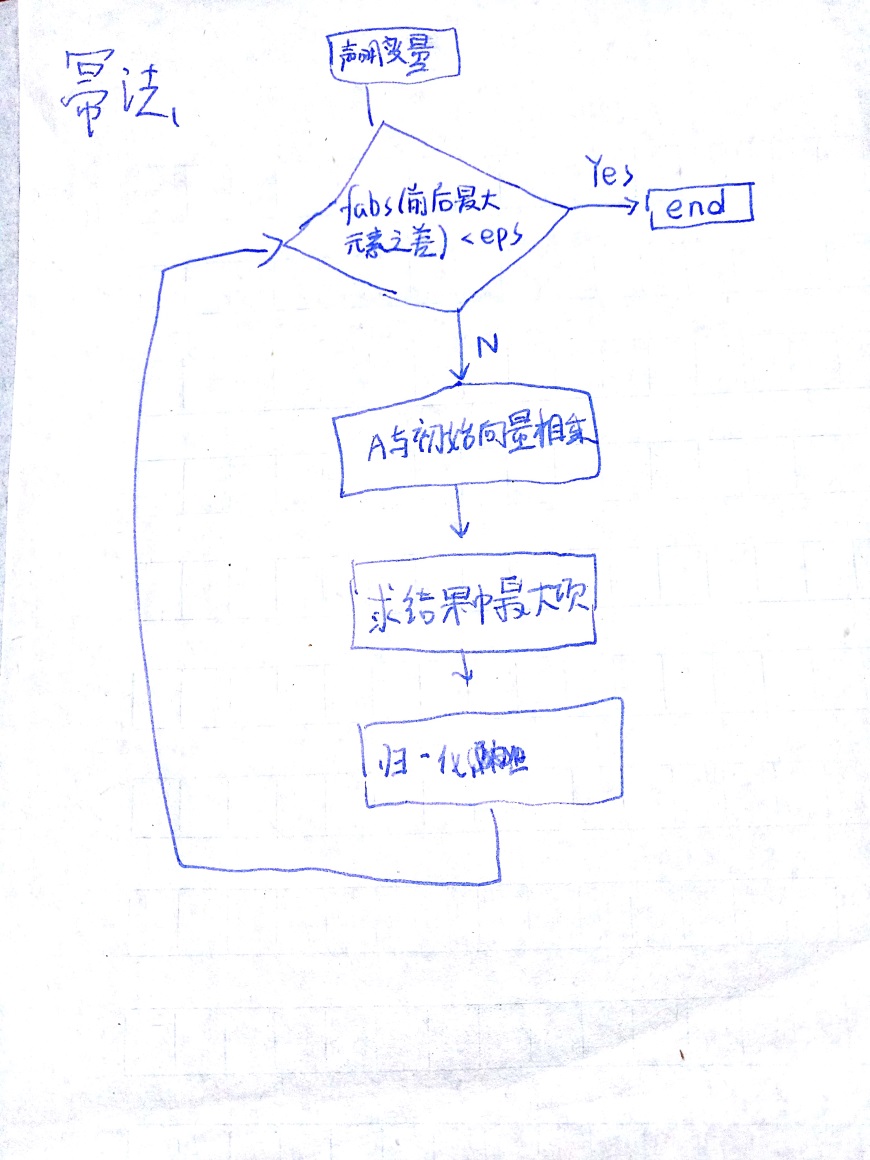
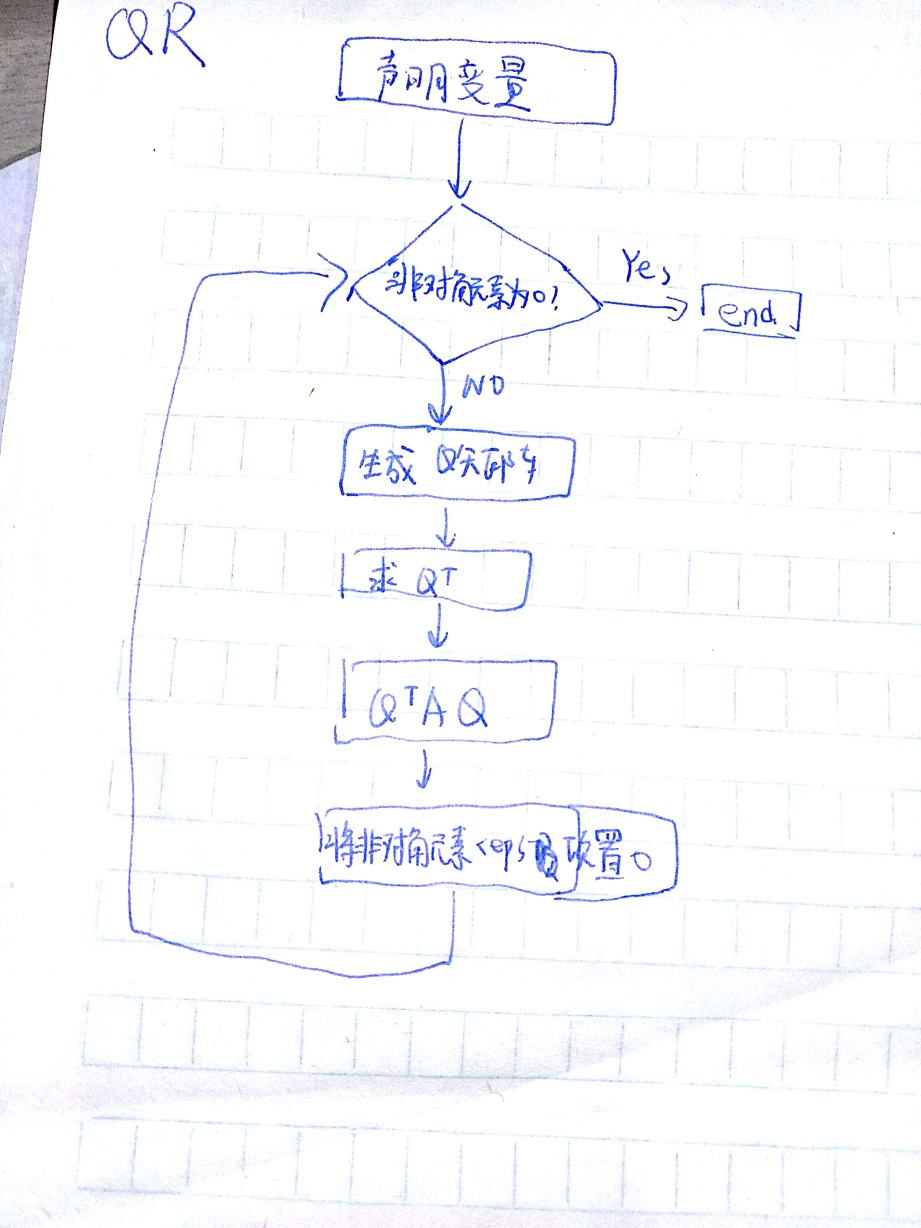
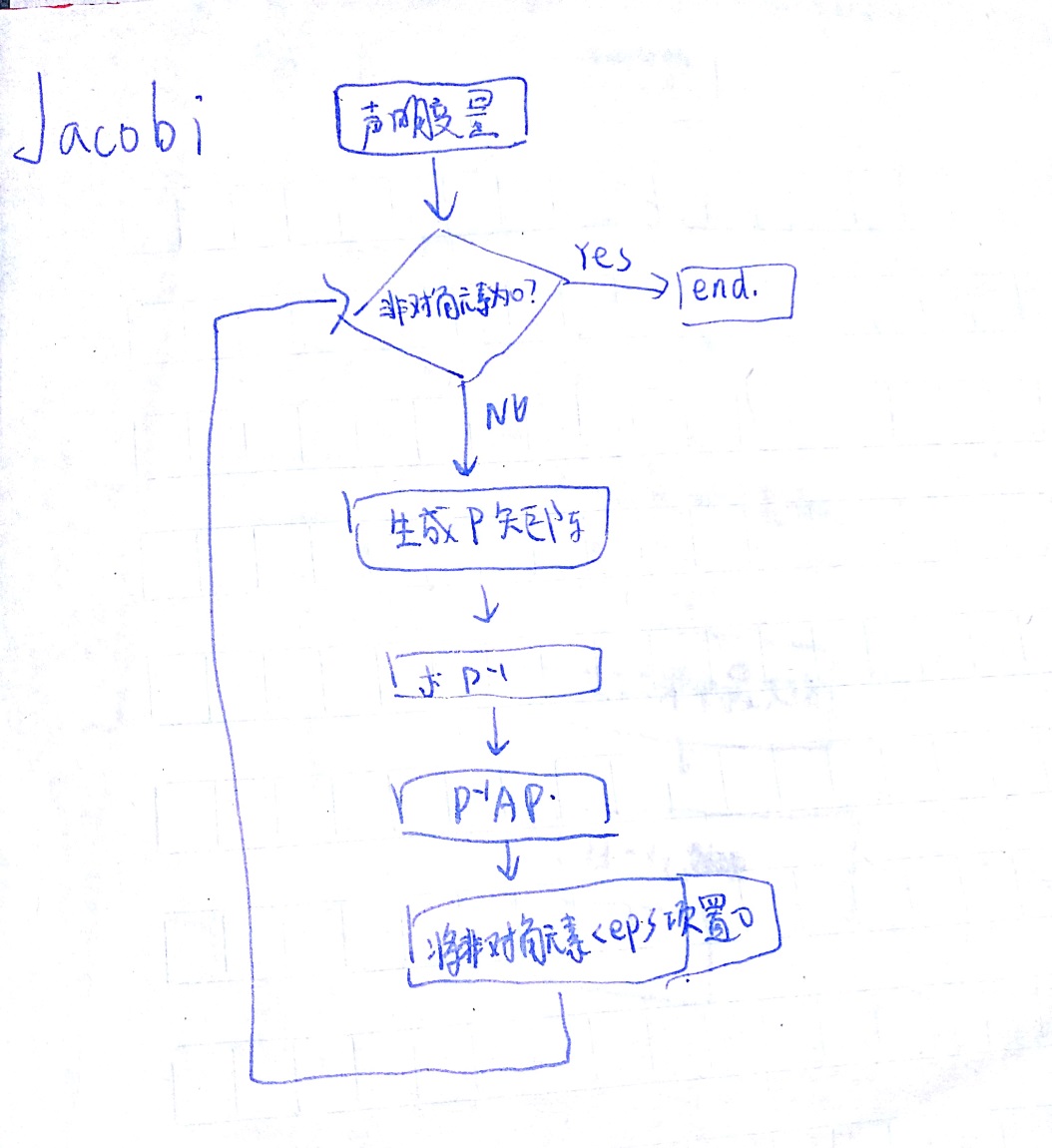
Jacobi 共进行7次迭代，每次迭代调用makeP(O(N^2)) company(O(N^4)) detA (O(N^4)) clear(O(N^2)) judge (O(N^2))函数一次，sqrmlt(O(N^3))函数2次.嵌套For循环2层O（N^2）.,合计时间复杂度为(O(N^4)).

QR共进行26次迭代,每次迭代调用makeQ(O(N^2)) transe (O(N^2))clear(O(N^2)) judge(O(N^2)) 一次，sqrmlt(O(N^3))函数2次, 合计时间复杂度为(O(N^3)).

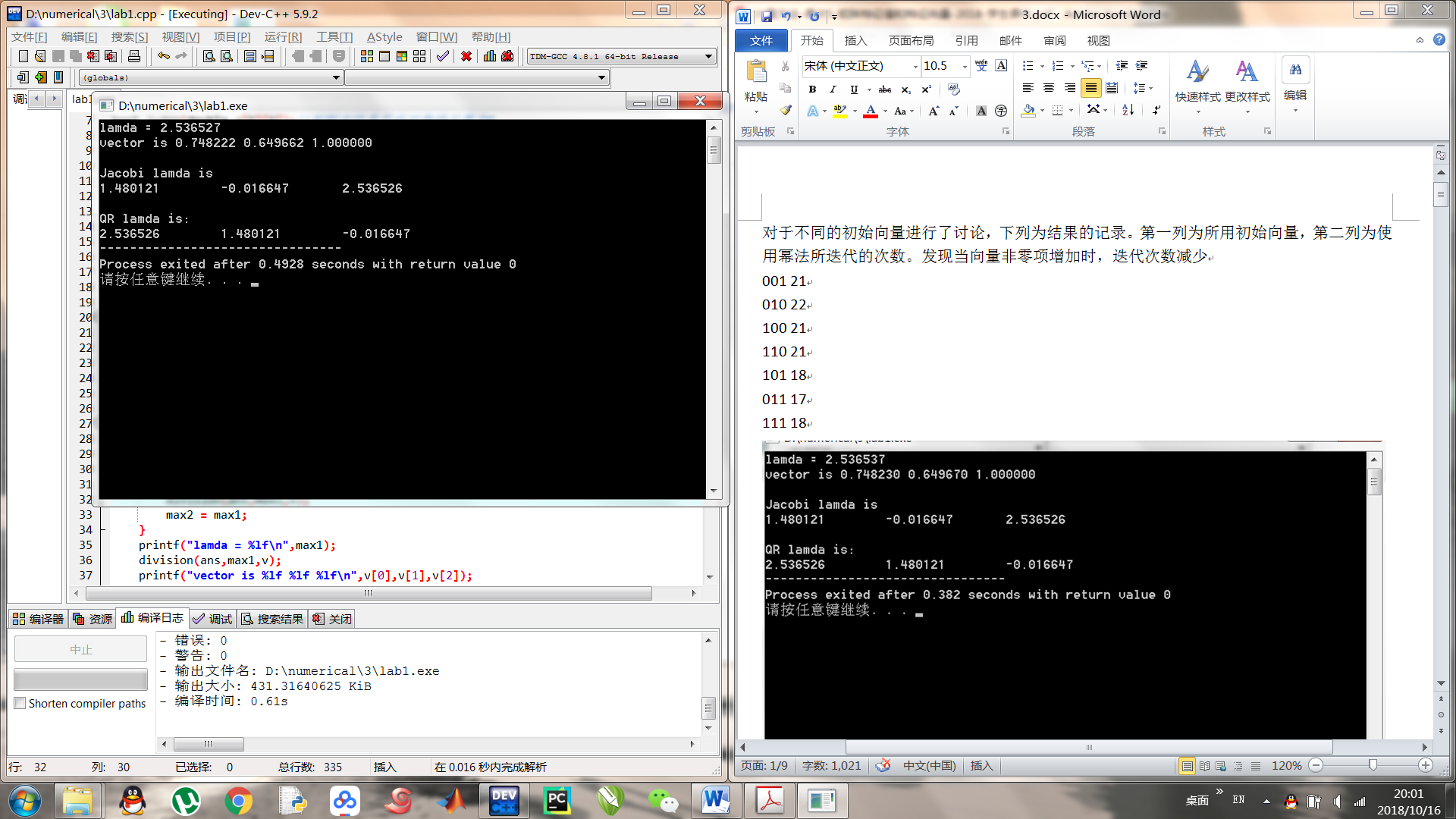
所以理论上对于求特征值来说， QR法计算量小，时间更快。通过修改运行程序添加计时函数，各自方法运行10000次，根据运行时间，也可以得出QR法运算更快的结论。结果如图，jacobi 0.021s QR 0.013s



流程图：



原始程序运行结果：



源代码：（注释复制过来变成了乱码，建议从附件lab1.cpp查看）

#include<stdio.h>

#include<math.h>

#define eps 1e-6

#define PI 3.14159265

#define N 3

double A[3][3] = {1,1,0.5,1,1,0.25,0.5,0.25,2};

bool judge(double a[N][N]);//ÅÐ¶Ï·¡è½Õó§?Ê§oÇ·¡èñ·¡èÇ¶Ô½ÇÏßÔªËØÎª0

double max(double a[]);//·¡èµ¨¬»ØÏò§¡ãÁ§¡é¿ÖÐ×¨¢î´?ó§?ÔªËØµ¨¬ÄÖµ¨¬

void makeP(double a[3][3],double b[3][3]);//¹¹Ôì§?Jacobi·¡è¨¡ìÖÐµ¨¬ÄP

double detA(double arcs[N][N],int n);//Çó§?·¡è½Õó§?µ¨¬ÄÐÐÁ§¡éÐÊ§o½Öµ¨¬

void mtrmlt(double a[N][N],double b[],double c[]);//·¡è½Õó§?³ËÏò§¡ãÁ§¡é¿ £¨º¬·¡èµ¨¬»ØÖÁ§¡éc

double vecmlt(double \*a,double \*b,int n);//Ïò§¡ãÁ§¡é¿ÄÚ§2»ý

void division(double a[],double n,double result[]);//ÃÝ·¡è¨¡ìÖÐ¶ÔÏò§¡ãÁ§¡é¿½øÐÐ±¨¤ê§o×¨¢¼»¯£¤ £¨º¬·¡èµ¨¬»ØÖÁ§¡éresult

void company(double arcs[N][N],int n,double ans[N][N]);//°?é§|Ëæ¾ØÕó§? ,·¡èµ¨¬»ØÖÁ§¡éans

void sqrmlt(double a[N][N],double b[N][N],double c[N][N]);//½«³Ë»ý·¡èÅÈ§¡§ëC¾ØÕó§?

void clear(double a[N][N]);//½«£¨º¼epsµ¨¬ÄÏîÖÃÁ§¡éã

void makeQ(double a[N][N],double q[N][N]);//¹¹Ôì§?Q¾ØÕó§?,·¡èµ¨¬»ØÖÁ§¡éq

void transe(double a[N][N],double b[N][N]);//×¨¢ªÖÃ

int main(){

/\*ÃÝ·¡è¨¡ì\*/

double v[3] = {-0.5315,-0.4615,-0.7103};

double ans[3] = {0};

double P[3][3] = {0},Pni[3][3] = {0},detP = 0,inter[N][N] = {0},end[N][N] = {0};

double max1 = 1,max2 = 0;

int i,j,k,m,n;

while(1){

mtrmlt(A,v,ans);

max1 = max(ans);

if(fabs(max1-max2) < eps)

break;

division(ans,max1,v);

max2 = max1;

}

printf("lamda = %lf\n",max1);

division(ans,max1,v);

printf("vector is %lf %lf %lf\n",v[0],v[1],v[2]);

printf("\n");

/\*Jacobi\*/

double multiply = 1;

while(1)

{

makeP(A,P);

company(P,N,Pni);

detP = detA(P,N);

for(i = 0;i < N;i ++)

{

for(j = 0;j < N;j++)

{

Pni[i][j] = Pni[i][j] / detP;

}

}

sqrmlt(Pni,A,inter);

sqrmlt(inter,P,A);

clear(A);

if(judge(A))

{

break;

}

}

printf("Jacobi lamda is\n");

for(i = 0;i < N;i++)

{

printf("%lf\t",A[i][i]);

}

printf("%\n");

/\*QR\*/

double A[3][3] = {1,1,0.5,1,1,0.25,0.5,0.25,2};

double Q[N][N],R[N][N],Qt[N][N],AA[N][N];

makeQ(A,Q);

transe(Q,Qt);

sqrmlt(Qt,A,R);

sqrmlt(R,Q,AA);

while(1)

{

makeQ(AA,Q);

transe(Q,Qt);

sqrmlt(Qt,AA,R);

sqrmlt(R,Q,AA);

clear(AA);

if(judge(AA))

{

break;

}

}

printf("\n");

printf("QR lamda is:\n");

for(i = 0;i < N;i++)

{

printf("%lf\t",AA[i][i]);

}

getchar();

}

void transe(double a[N][N],double b[N][N])

{

int i,j;

for(i = 0;i < N - 1;i++)

{

for(j = i + 1;j < N;j++)

{

b[i][j] = a[j][i];

b[j][i] = a[i][j];

}

}

for(i = 0;i < N;i++)

{

b[i][i] = a[i][i];

}

}

double vecmlt(double \*a,double \*b,int n)

{

int i = 0;

double result = 0;

for(;i < n;i++)

{

result += \*(a + i) \* \*(b + i);

}

return result;

}

void makeQ(double a[N][N],double q[N][N])

{

double a1[N],a2[N],a3[N];

int i;

for(i = 0;i < N;i++)

{

a1[i] = a[i][0];

a2[i] = a[i][1];

a3[i] = a[i][2];

}

double para2 = -vecmlt(a1,a2,N)/vecmlt(a1,a1,N);

for(i = 0;i < N;i++)

{

a2[i] = a[i][1] + para2 \* a1[i];

}

double para3 = -vecmlt(a1,a3,N)/vecmlt(a1,a1,N);

para2 = -vecmlt(a2,a3,N)/vecmlt(a2,a2,N);

for(i = 0;i < N;i++)

{

a3[i] = a[i][2] + para3 \* a1[i] + para2 \* a2[i];

}

for(i = 0;i < N;i++)

{

q[i][0] = a1[i]/sqrt(vecmlt(a1,a1,N));

q[i][1] = a2[i]/sqrt(vecmlt(a2,a2,N));

q[i][2] = a3[i]/sqrt(vecmlt(a3,a3,N));

}

}

bool judge(double a[N][N])

{

int i = 0,j = 0;

for(i = 0;i < N - 1;i++)

{

for(j = i + 1;j < N;j++)

{

if(fabs(a[i][j]) > eps)

{

return 0;

}

}

}

return 1;

}

void clear(double a[N][N])

{

int i,j;

for(i = 0;i < N - 1;i++)

{

for(j = i + 1;j < N;j++)

{

if(fabs(a[i][j]) <= eps)

{

a[i][j] = 0;

a[j][i] = 0;

}

}

}

}

void mtrmlt(double a[N][N],double b[],double c[])

{

int i = 0,j = 0;

for(j = 0;j<3;j++)

c[j] = 0;

for(j = 0;j<3;j++)

for(i = 0;i<3;i++)

c[j] += a[j][i] \* b[i];

}

double max(double a[])

{

int i = 1;

double max = a[0];

while(i < N){

if (max < fabs(a[i]))

max = a[i];

i++;

}

return max;

}

void division(double a[],double n,double result[])

{

int i = 0;

while(i < 3){

result[i] = a[i] / n;

i++;

}

}

void makeP(double a[N][N],double b[N][N])

{

int i = 0,j = 0;

int l = 0,c = 0;

double max = 0;

for(;i < N;i++)

for(j = 0;j < N;j++)

b[i][j] = 0;

for(i = 0;i < N;i++){

for(j = i + 1;j < N;j++){

if(fabs(a[i][j]) > max){

l = i;

c = j;

max = fabs(a[i][j]);

}

}

}

double theta = 0;

if(a[l][l] == a[c][c]){

theta = PI / 4;

}

else{

theta = atan(2 \* a[l][c] / (a[l][l] - a[c][c])) / 2;

}

b[l][l] = cos(theta);

b[c][c] = cos(theta);

b[l][c] = -sin(theta);

b[c][l] = sin(theta);

for(i = 0;i < N;i++)

{

if(b[i][i] == 0)

{

b[i][i] = 1;

}

}

}

double detA(double arcs[N][N],int n)

{

if(n==1)

{

return arcs[0][0];

}

double ans = 0;

double temp[N][N];

int i,j,k;

for(i=0;i<n;i++)

{

for(j=0;j<n-1;j++)

{

for(k=0;k<n-1;k++)

{

temp[j][k] = arcs[j+1][(k>=i)?k+1:k];

}

}

double t = detA(temp,n-1);

if(i%2==0)

{

ans += arcs[0][i]\*t;

}

else

{

ans -= arcs[0][i]\*t;

}

}

return ans;

}

void company(double arcs[N][N],int n,double ans[N][N])

{

if(n==1)

{

ans[0][0] = 1;

return;

}

int i,j,k,t;

double temp[N][N];

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

for(k=0;k<n-1;k++)

{

for(t=0;t<n-1;t++)

{

temp[k][t] = arcs[k>=i?k+1:k][t>=j?t+1:t];

}

}

ans[j][i] = detA(temp,n-1);

if((i+j)%2 == 1)

{

ans[j][i] = - ans[j][i];

}

}

}

}

void sqrmlt(double a[N][N],double b[N][N],double c[N][N])

{

int i,j,k;

for(i = 0;i < N;i++)

{

for(j = 0;j < N;j++)

{

c[i][j] = 0;

for(k = 0;k < N;k++)

{

c[i][j] += a[i][k] \* b[k][j];

}

}

}

}