**EE407002 Numerical Analysis**

**Homework 6. Matrix Condition Numbers**

102061110 YUNG-SHAN SU

1. **Introduction**

The power method and inverse power method are good ways to find the extreme values of matrix spectrum. However, when we need to find all eigenvalues, these methods are not very appropriate. As a result, in this assignment, I will implement QR iteration and shifted QR iteration to find all eigenvalues. Additionally, to discuss the property of this algorithm, I will discuss the computation complexity and its convergence rate.

1. **Algorithm**
2. **QR iteration**

Algorithm-1 QR iteration

int EVqr(MAT &A,double tol,int maxiter){

MAT r(A.dim()); //Declare r

MAT q(A.dim()); //Declare q

double err=1+tol;

r=0;

q=0;

int count=0;

while(count<maxiter&&err>tol){

//QR shifting Algorithm

A=A.tpose();

r[0][0]=sqrt(A[0]\*A[0]); //r11=sqrt(A1\*A1)

q[0]=A[0]/r[0][0]; //q1=A1/r11

for(int j=1;j<A.dim();j++){

q[j]=A[j];

for(int i=0;i<j;i++){

r[j][i]=q[i]\*A[j]; //rij=qi\*Aj

q[j]=q[j]-r[j][i]\*q[i];

}

r[j][j]=sqrt(q[j]\*q[j]); //rjj=Aj-sigma(rij\*qi)

q[j]=(q[j])/r[j][j]; //qj=qj/rjj

}

A=(r.tpose())\*(q.tpose());

//Calculate error

err=A[1][0];

for(int i=2;i<A.dim();i++)

if(A[i][i-1]>err)

err=A[i][i-1];

count++;

}

return count;

}

1. **Shifted QR iteration**

Algorithm-2 Shifted QR iteration

int EVqrShifted(MAT &A,double mu,double tol,int maxiter){

MAT r(A.dim()); //Declare r

MAT q(A.dim()); //Declare q

MAT In(A.dim());

double err=1+tol;

VEC reg(A.dim());

In=0;

r=0;

q=0;

for(int i=0;i<A.dim();i++)

In[i][i]=1;

int count=0;

//Shifted QR iteration Algorithm

while(count<maxiter&&err>tol){

A=A-(mu\*In); //A=A-mu\*In

A=A.tpose();

r[0][0]=sqrt(A[0]\*A[0]); //r11=sqrt(A1\*A1)

q[0]=A[0]/r[0][0]; //q1=A1/r11

for(int j=1;j<A.dim();j++){

q[j]=A[j];

reg=0;

for(int i=0;i<j;i++){

r[j][i]=q[i]\*A[j];

reg+=r[j][i]\*q[i];

}

q[j]=q[j]-reg;

r[j][j]=sqrt(q[j]\*q[j]);

q[j]=(q[j])/r[j][j]; //qj=qj/rjj

}

A=(r.tpose())\*(q.tpose())+(mu\*In);//A=A+mu\*In

//Calculate error

err=fabs(A[1][0]);

for(int i=2;i<A.dim();i++)

if(fabs(A[i][i-1])>err)

err=fabs(A[i][i-1]);

count++;

}

return count;

}

1. **Discussion**

Table-1 QR Iteration and Shifted QR Iteration CPU Time

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| QR Iteration | | | | Shifted QR Iteration | | | |
| Dim | Iteration number | CPU  Time(sec) | Average  Iteration  Time(sec) | Dim | Iteration number | CPU time(sec) | Average  Iteration  Time(sec) |
| 3 | 20 | 0.001184 | 0.0000592 | 3 | 17 | 0.000281 | 1.65294E-05 |
| 10 | 249 | 0.037274 | 0.000149695 | 10 | 35 | 0.00591 | 0.000168857 |
| 20 | 909 | 0.848516 | 0.000933461 | 20 | 67 | 0.048706 | 0.000726955 |
| 30 | 1939 | 5.97325 | 0.003080583 | 30 | 106 | 0.169524 | 0.001599283 |
| 40 | 3262 | 19.475382 | 0.005970381 | 40 | 136 | 0.380352 | 0.002796706 |
| 50 | 4974 | 52.900662 | 0.010635437 | 50 | 176 | 0.932258 | 0.00529692 |

Table-2 Eigenvalues

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dim |  |  |  |  |  |  |
| 3 | 6.372281 | 2 | 0.627719 | 6.372281 | 2 | 0.627719 |
| 10 | 67.840399 | 20.431729 | 4.455992 | 0.629808 | 0.55164 | 0.512543 |
| 20 | 270.495189 | 81.223819 | 17.235222 | 0.528819 | 0.512479 | 0.503097 |
| 30 | 608.253606 | 182.544889 | 38.53868 | 0.512543 | 0.505511 | 0.501373 |
| 40 | 1081.115447 | 324.394506 | 68.364136 | 0.507004 | 0.503093 | 0.500772 |
| 50 | 1689.080688 | 506.772618 | 106.711318 | 0.504441 | 0.502 | 0.500486 |

1. **Computation complexity**
2. **CPU Time**

Graph-1 CPU Time

Observing QR Iteration and Shifted QR iteration CPU time to n^4 and n^3 , we can find the line(QR iteration) is horizontal to line(n^4), and the line(Shifted QR iteration) is horizontal to line(n^3). As a result, we speculate:

Computational complexity of QR iteration is O()

Computational complexity of Shifted QR iteration is O()

1. **Each Iteration Time**

Graph-2 Average Iteration Time

From Algorithm-1 and Algorithm-2, we can find in each loop of these algorithms, it include matrix multiplication. As a result, the computation complexity is O(). Also, from Graph-2, we can find these lines are horizontal to the line n^3. This can also prove they are in direct proportion.

1. **Convergence rate**
2. **Reference**

Lec42.pdf written by Professor Chang, Mi-Chang