SHANGHAI JIAO TONG UNIVERSITY
SCHOOL OF MATHEMATICAL SCIENCES

Report for Matrix Theory Computer Project (2021-Spring)

HAN Yuxuan, LIU Heng and TAN Zheng

June 13, 2021

Part I

Content

Problem 1 - Calculating Jordan Canonical Form

Using Eigen Library(C++):

- Compute eigenvalues using EigenSolver.
- Compute vector d_i and j_i associated with each eigenvalue λ_i.
- Construct the Jordan block for Jordan Canonical Form.

Problem 1 - Details of Our Codes

- Cauculate every eigenvalue of M and store them in vector $eigenvalue$, remove the repeat eigenvalues.

C++ Codes

```cpp
std::vector<double> eigenvalue;
Eigen::EigenSolver<Eigen::MatrixXd> es(mat_in);
eigenvalue.push_back(es.eigenvalues()[0].real());
for (int eig = 1; eig < es.eigenvalues().size(); eig++){
    if(abs(es.eigenvalues()[eig].real() - eigenvalue[eigenvalue.
        size() - 1]) > 0.01){
        eigenvalue.push_back(es.eigenvalues()[eig].real()); }
    else{}
}
```

Problem 1 - Details of Our Codes

- Cauculate $dim(ker(M_\lambda^k))$, stop adding k until $rank(M_\lambda^k)$ stop changing.

C++ Codes

```cpp
Eigen::FullPivLU<Eigen::MatrixXd> lu_decomp(mat_lambda_power);
int dimker = dim - lu_decomp.rank();
if(DimKer.size() == 0){
    DimKer.push_back(dimker);
    last_dimker = dimker;}
else{
    if(last_dimker != dimker){
        DimKer.push_back(dimker);
        last_dimker = dimker;}
    else{con_pow = 0;}}
```

Problem 1 - Details of Our Codes

- Cauculate d_i according to $dim(ker(M_\lambda^k))$.
- Cauculate j_i according to d_i.

C++ Codes

```cpp
for (int m = 0; m < DimKer.size(); m++) {
    if (m == 0) { d.push_back(DimKer[m]); }
    else { d.push_back(DimKer[m] - DimKer[m - 1]);}}
D.push_back(d);
for (int n = 0; n < DimKer.size(); n++) {
    if (n == 0) { j.push_back(d[DimKer.size() - 1]); }
    else { j.push_back(d[DimKer.size() - n - 1] - d[DimKer.size() -
        n]); }}
J.push_back(j);
```

Problem 1 - Details of Our Codes

- Show Jordan Canonical Form according to each λ_i and its associated d_i, j_i.

C++ Codes

```cpp
for (int itr = 0; itr < eigenvalue.size(); itr++){
    for (int sub_j=0; sub_j<J[itr].size();sub_j++){
        for (int num_sub_j=0;num_sub_j<J[itr][sub_j];num_sub_j++){
            int end_row = start_row + J[itr].size() - sub_j;
            for (int sub_row=start_row;sub_row<end_row;sub_row++){
                for (int sub_col=start_row;sub_col<end_row;sub_col++){
                    if(sub_row==sub_col){mat_Jordan(sub_row,sub_col) =
                        eigenvalue[itr];}
                    else if(sub_col == sub_row+1){mat_Jordan(sub_row,
                        sub_col) = 1;}}}
                start_row = end_row;}}}
```

```
PS C:\Users\lenovo\source\repos\mtcp_1\Debug> .\mtcp_1.exe
chooose problem number(1 or 2):
1
intput command to run a demo.
command 'std' means input in std stream.
command 'rand' means run a random demo
command 'quit' means quit
>> std
please type the dimension of your linear operator here:
4
please type your linear operator here:
3 -4 0 2
4 -5 -2 4
0 0 3 -2
0 0 2 -1
The matrix you input:
 3 -4  0  2
 4 -5 -2  4
 0  0  3 -2
 0  0  2 -1
The eigenvalues of the input matrix are:
(-1,0)
(-1,0)
 (1,0)
 (1,0)
```

```
Size of eigenvalues:
4
Dim: 4
The eigenvalues are: -1 1
Length of eigenvalue:2
-1
vector d=[  1  1  ]
vector j=[  1  0  ]
1
vector d=[  1  1  ]
vector j=[  1  0  ]
D:2
J:2
After process
Eigenvalue: -1
Start_row: 0
Eigenvalue: 1
Start_row: 2
The Jordan Matrix should be:
-1  1  0  0
 0 -1  0  0
 0  0  1  1
 0  0  0  1
>> 
```

Figure: demo: input from user-defined matrix

Problem 2 -

Using Eigen Library(C++):

- Compute eigenvalues using SelfAjointEigenSolver(es in the codes below).
- Push back the diagonal eigenvalues and orthonormal eigenvectors in $mats\_out$.

C++ Codes

```
void uSimilar::process()
{
    es.compute(mat_in);
    mats_out.push_back(es.eigenvalues().asDiagonal());
    mats_out.push_back(es.eigenvectors());
}
```

```
PS C:\Users\lenovo\source\repos\mtcp_1> .\Debug\mtcp_1.exe
chooose problem number(1 or 2):
2
intput command to run a demo.
command 'std' means input in std stream.
command 'rand' means run a random demo
command 'quit' means quit
>> rand
Here is a matrix:
  -1.99499    0.297189  0.0322886    1.03848
  0.297189 -0.0805078   0.193793    0.818995
 0.0322886    0.193793   -1.30357   0.325877
   1.03848    0.818995   0.325877   -1.94006
After process
```

```
eigenbasis matrix V:
-0.663372 -0.609898   0.32813   -0.28336
-0.126677   0.29025 -0.441557 -0.839485
-0.108252  0.610492  0.764849 -0.174888
 0.729501 -0.413617  0.335207 -0.429401
V*V^dagger:
           1  2.77556e-17 -4.16334e-17  4.16334e-16
 2.77556e-17            1  1.38778e-16 -2.22045e-16
-4.16334e-17  1.38778e-16            1 -1.38778e-16
 4.16334e-16 -2.22045e-16 -1.38778e-16            1
diagonal matrix D:
-3.07498        0        0        0
       0 -1.46447        0        0
       0        0 -1.25877        0
       0        0        0 0.479098
>>
```

Figure: demo: input from random symetric matrix

Problem 3(a) – Linear Fitting

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{1}$$

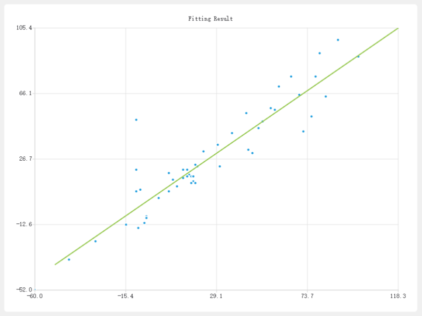
$$d\left(b, C\left(A\right)\right) = ||Ax - b|| \tag{2}$$

Problem 3(a) – Codes

C++ Codes

```cpp
//ATAx = ATb
F_result = (A.transpose() * A).ldlt().solve(A.transpose() * b);
cout << "The solution using normal equations is:\n"
    << F_result << endl;
cout << "The distance is : " << (A * F_result - b).norm() <<
    endl;
```

Problem 3(a) – Program Results



$$d(b, C(A)) = 84.89 \tag{3}$$

Problem 3(b) – Parabolic Fitting

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{4}$$

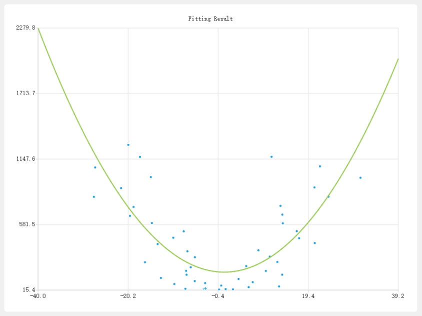
$$d\left(b, C\left(A\right)\right) = ||Ax - b|| \tag{5}$$

Problem 3(b) – Codes

C++ Codes

```cpp
for (int i = 0; i < N; i++)
{
    A(i, 0) = mat(i, 0) * mat(i, 0);
    A(i, 1) = mat(i, 0);
    A(i, 1) = 1;
    b(i) = mat(i, 1);
}
//ATAx = ATb
F_result = (A.transpose() * A).ldlt().solve(A.transpose() * b);
cout << "The distance is : " << (A * F_result - b).norm() <<
    endl;
```

Problem 3(b) – Program Results



$$d(b, C(A)) = 1789.58 \tag{6}$$

Thank You

HAN Yuxuan, LIU Heng and TAN Zheng · Report for Matrix Theory Computer Project
(2021-Spring)