Software assignment

EE24BTECH11050-Pothuri Rahul

1 Eigenvalues

Let A be a square matrix of order n and λ be a scalar such that, for any non-zero matrix X,

$$AX = \lambda X$$

Those values of λ are defined as Eigenvalues.
Eigenvalues are scalars associated with a linear system of equations, and are used to transform eigenvectors.
Their are many ways to compute the eigenvalues. One of those is by doing QR decomposition.

2 QR Decomposition

This is the method of factorizing the matrix into two matrices,namely Q and R. Where the matrix Q is orthogonal and matrix R is Upper triangular matrix.
Let A be the input matrix. We get Q matrix by Gram-Schmidt orthogonalization process.
The Gram-Schmidt orthogonalization process is a method used to convert a set of linearly independent vectors into an orthogonal set.
To convert a set of linearly independent vectors into an orthogonal set using Gram-Schmidt method:
Mathematically, we compute the columns of Q and the entries of R as:

- The k-th column of Q is computed as:

  $$Q[:,k] = \frac{A[:,k]}{\|A[:,k]\|}$$

  where $\|A[:,k]\|$ is the norm of the k-th column of A.

- The R matrix entries are computed as:

  $$R[k,j] = Q^T A[:,j]$$

  for $j \geq k$.

- The matrix A is then updated by subtracting the projection onto the k-th column of Q, i.e.,

  $$A[:,j] = A[:,j] - Q[:,k]R[k,j]$$

  for $j > k$.

3 How To Find Eigenvalues Using QR Decomposition

Let A_0=A, where A is the input matrix.

$$\begin{gathered} A_0 = Q_0 R_0 \\ \text{Form } A_1 = R_0 Q_0 \\ \text{then } A_1 \text{ can be factorised into } A_1 = Q_1 R_1 \\ \text{Form } A_2 = R_1 Q_1 \\ \text{then } A_2 \text{ can be factorised into } A_2 = Q_2 R_2 \end{gathered}$$

Iterate until convergence.That is,till the matrix formed A_k becomes diagonal.
And the diagonal elements in the A_k are the required Eigenvalues of the input matrix A.

4 Explaination Of code

4.1 Libraries used:

```
#include <stdio.h>
#include <math.h>
```

stdio.h is used for input and output.
math.h is used for mathematical operations.

4.2 Function used for Multiplication of Matrices:

```
void Multiply(int n,float A[n][n],float B[n][n],float AB[n][n]){
    for(int i = 0 ; i < n ; i++){
        for(int j = 0 ; j < n ; j++){
            AB[i][j] = 0 ;
        }
    }
    for(int i = 0 ; i < n ; i++){
        for(int j = 0 ; j < n ; j++){
            for(int k = 0 ; k < n ; k++){
            AB[i][j] = AB[i][j] + ( A[i][k] * B[k][j] );
            }
        }
    }
}
```

4.3 Function for tarnspose a Matrix:

```
void transpose(int r , int c , float A[r][c] , float B[c][r]){
    for(int i = 0 ; i < r ; i++){
        for(int j = 0 ; j < c ; j++){
            B[j][i] = A[i][j] ;
```

```
        }
    }
}
```

4.4 Function for QR Decomposition:

```
void qrDecomposition(int n , float A[n][n] , float Q[n][n] , float R[n][n]){

    for(int i = 0 ; i < n ; i++){
        for(int j = 0 ; j < n ; j++){
            Q[i][j] = 0 ;
            R[i][j] = 0 ;
        }
    }

    for(int k = 0 ; k < n ; k++){
        double normSqr = 0;
        for(int i = 0 ; i < n ; i++){
            normSqr = normSqr + pow(A[i][k] , 2);
        }
        R[k][k] = sqrt(normSqr);
        for(int i = 0 ; i < n ; i++){
            Q[i][k] = A[i][k]/R[k][k] ;
        }
        for (int j = k + 1 ; j < n; j++) {
            R[k][j] = 0;
            for (int i = 0 ; i < n ; i++) {
                R[k][j] = R[k][j] + ( Q[i][k] * A[i][j] );
            }
            for (int i = 0 ; i < n ; i++) {
                A[i][j] = A[i][j] - ( Q[i][k] * R[k][j] );
            }
        }
    }
}
```

1. Intialise all elements of Q and R to 0.
2. R[k][k] terms equals to the norm of k^{th} column of A.
3. The orthogonal vector(Q) is the normalized version of the k-th column of A, so each entry $A[i][k]$ is divided by $R[k][k]$ to ensure that the column in Q has unit length.
4. For each column j of A, starting from the $(k+1)$-th column, the code computes the projection of the j-th column of A onto the k-th orthogonal vector $Q[:,k]$. This projection is used to determine the coefficients for the matrix R:
5. Subtracting the Projection from A to Make Columns Orthogonal.

4.5 Function to find eigenvalues by QR Decomposition

```
void Algorithm(int n , float A[n][n] , float result[n]){
    float Q[n][n] , R[n][n];

    for(int x = 0 ; x < 1000 ; x++){
        float A1[n][n];
        for(int i = 0 ; i < n ; i++){
            for(int j = 0 ; j < n ; j++){
                A1[i][j] = A[i][j];
            }
        }
        qrDecomposition(n , A1 , Q , R);
        Multiply(n , R , Q , A);
        double nonDiagNorm = 0;
        for (int i = 0 ; i < n ; i++) {
            for (int j = 0 ; j < n ; j++) {
                if (i != j) {
                    nonDiagNorm = nonDiagNorm + pow(A[i][j] , 2 )
                        ;
                }
            }
        }
        for (int i = 0 ; i < n ; i++) {
        result[i] = A[i][i];

        }
        if (sqrt(nonDiagNorm) < pow(10,-100)) {
            break;
        }
    }


}
```

This algorithm is used to find the eigenvalues by QR decomposition. The whole process happens here is explained in Section 3.

4.6 Main function:

```
int main(){
    int n;
    printf("Enter the size of the matrix: ");
    scanf("%d",&n);
    float A[n][n];
    float result[n];
    for(int i = 0 ; i < n ; i++){
        for(int j = 0 ; j < n ; j++){
            scanf("%f", &A[i][j]);
        }
    }
```

```c
    Algorithm( n , A , result);
    printf("Eigenvalues of the given Mastrix are : \n");
    for(int i = 0 ; i < n ; i++){
        printf("%f ",result[i]);
    }

    return 0;
}
```

Main function is used to take the inputs and give out the output.

4.6.1 Input:

The order of the Matrix (n) and the elements (n^2 elements) are taken as the input of the matrix.

4.6.2 Output:

The n-eigenvalues of the input matrix print as output.

5 Verification

5.1 Input:

5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25.

5.2 output

Eigenvalues of the given Matrix are :
68.642082 -3.642080 -0.000000 -0.000000 0.000000

6 Time Complexity

In general, Time complexity of

1. QR Decomposition is $O(n^3)$
2. Matrix Multiplication is $O(n^3)$
3. Norm calculation and Convergence check is $O(n^2)$

Hence, Total time complexity is $O(n^3)$
Let m be no.of itrations until convergence, Then Total time complexity is $O(mn^3)$.

7 Comparision Of Algorithms

As mentioned earlier their are lot many Algorithms to find the eigenvalues of a given matrix.

1. Power Itration :For m itrations,Time complexity is $O(mn^2)$, but in this method we can compute only the largest eigenvalue among all eigenvalues.Highly accurate for the largest eigenvalue but fails for others. Sensitive to conditioning.

2. Jacobi Method: Time complexity of this method is $O(n^3)$, but this method is used only for the symmetric matrices.Exact for symmetric matrices, numerically stable. Slow convergence for large matrices.

3. Inverse Power Iteration: Requires matrix inversion which is expensive and finds only one eigenvalue when we execute the process.

Among all these, Finding eigenvalues using QR Decomposition(using Gram-Schmidt) is better as this algorithm works for any matrix.Time complexity of this process is comparable with all other processes.