NUMERICAL METHODS

EIGENVECTORS AND EIGENVALUES

Sergio Barbero, Mateusz Korusiewicz

Introduction to the problem

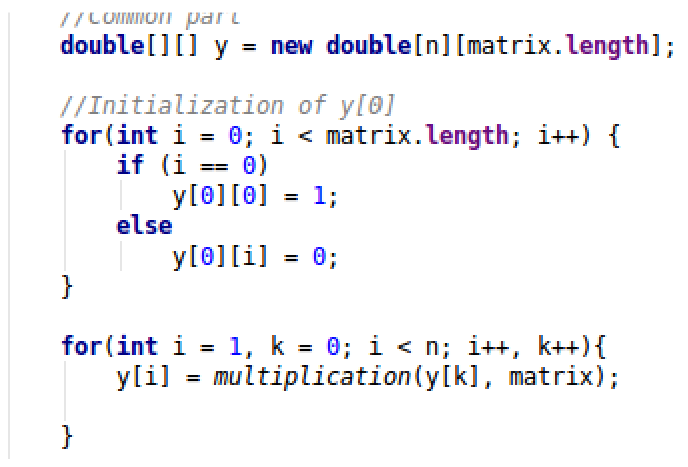
We want to study the eigenvalues and eigenvectors of a matrix and the accuracy of the eigenvalue returned by the “Power Method” compared with the analytical solution of our matrix, we also will get the eigenvalues retrieves by Krylov Method.

We worked with the library “Jama” for Java in order to get the analytical eigenvectors and eigenvalues.

The called “Power Method” retrieves the highest-modulus eigenvalue with highest modulus, we will compare this result with the analytical solution.

Implementation

For both methods we have a common part, we initialize the first iteration with 1 in the first row and 0s in the rest of them, after, we multiply the vector y[k] by the matrix, in order to get the new vector y[i], with i=k-1.



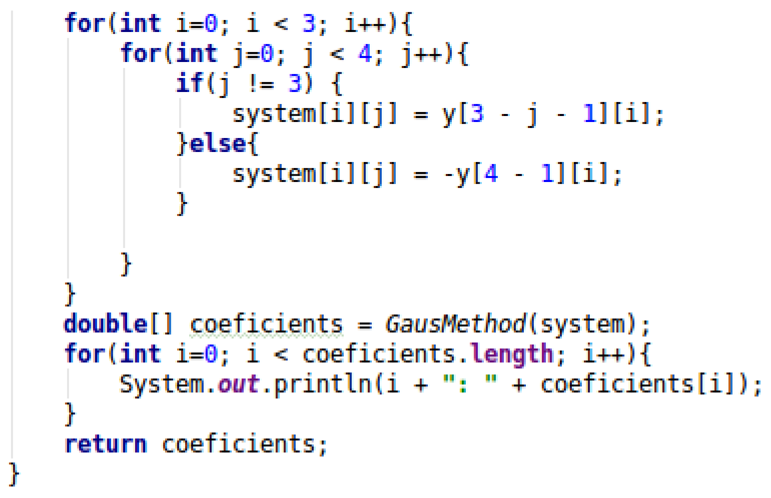
```java
//common part
double[][] y = new double[n][matrix.length];

//Initialization of y[0]
for(int i = 0; i < matrix.length; i++) {
    if (i == 0)
        y[0][0] = 1;
    else
        y[0][i] = 0;
}

for(int i = 1, k = 0; i < n; i++, k++){
    y[i] = multiplication(y[k], matrix);

}
```

For Krylov Method we perform a system of 3 equations, in order to get the coefficients p:



```java
    for(int i=0; i < 3; i++){
        for(int j=0; j < 4; j++){
            if(j != 3) {
                system[i][j] = y[3 - j - 1][i];
            }else{
                system[i][j] = -y[4 - 1][i];
            }

        }
    }
    double[] coeficients = GausMethod(system);
    for(int i=0; i < coeficients.length; i++){
        System.out.println(i + ": " + coeficients[i]);
    }
    return coeficients;
}
```

Once we have these coefficients we just need to solve the 3rd grade equation, in our case:

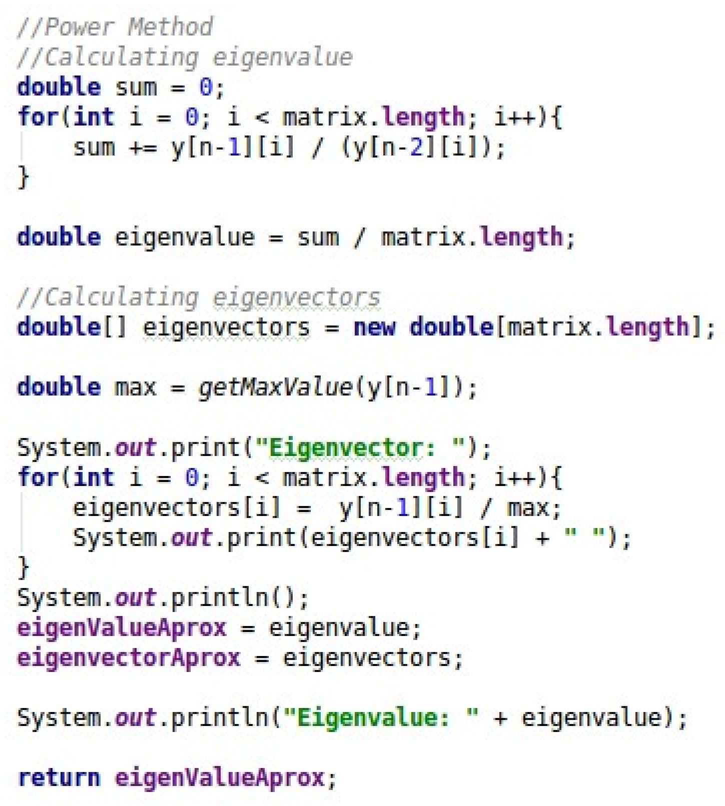
$\lambda^3 - 9\lambda^2 + 9 * \lambda - 1 = 0$

The results of lambda are:

$\lambda = 1 ; \lambda = 0.12702 ; \lambda = 7.8730$

The results are very perfect approximations of our analytic eigenvalues

In tthe case of the Power Method for calculating the eigenvalue, we have to accumulate the value of the division between the last iteration of y and the previous one in the differents rows, finally this accumulated value is divided by the lenght of the matrix and the result is the eigenvalue.



```
//Power Method
//Calculating eigenvalue
double sum = 0;
for(int i = 0; i < matrix.length; i++){
    sum += y[n-1][i] / (y[n-2][i]);
}

double eigenvalue = sum / matrix.length;

//Calculating eigenvectors
double[] eigenvectors = new double[matrix.length];

double max = getMaxValue(y[n-1]);

System.out.print("Eigenvector: ");
for(int i = 0; i < matrix.length; i++){
    eigenvectors[i] =  y[n-1][i] / max;
    System.out.print(eigenvectors[i] + " ");
}
System.out.println();
eigenValueAprox = eigenvalue;
eigenvectorAprox = eigenvectors;

System.out.println("Eigenvalue: " + eigenvalue);

return eigenValueAprox;
```

In the next part we calculate the eigenvector, we store the value of the last iteration of y in the column i between max (maximum value of the last iteration of y).

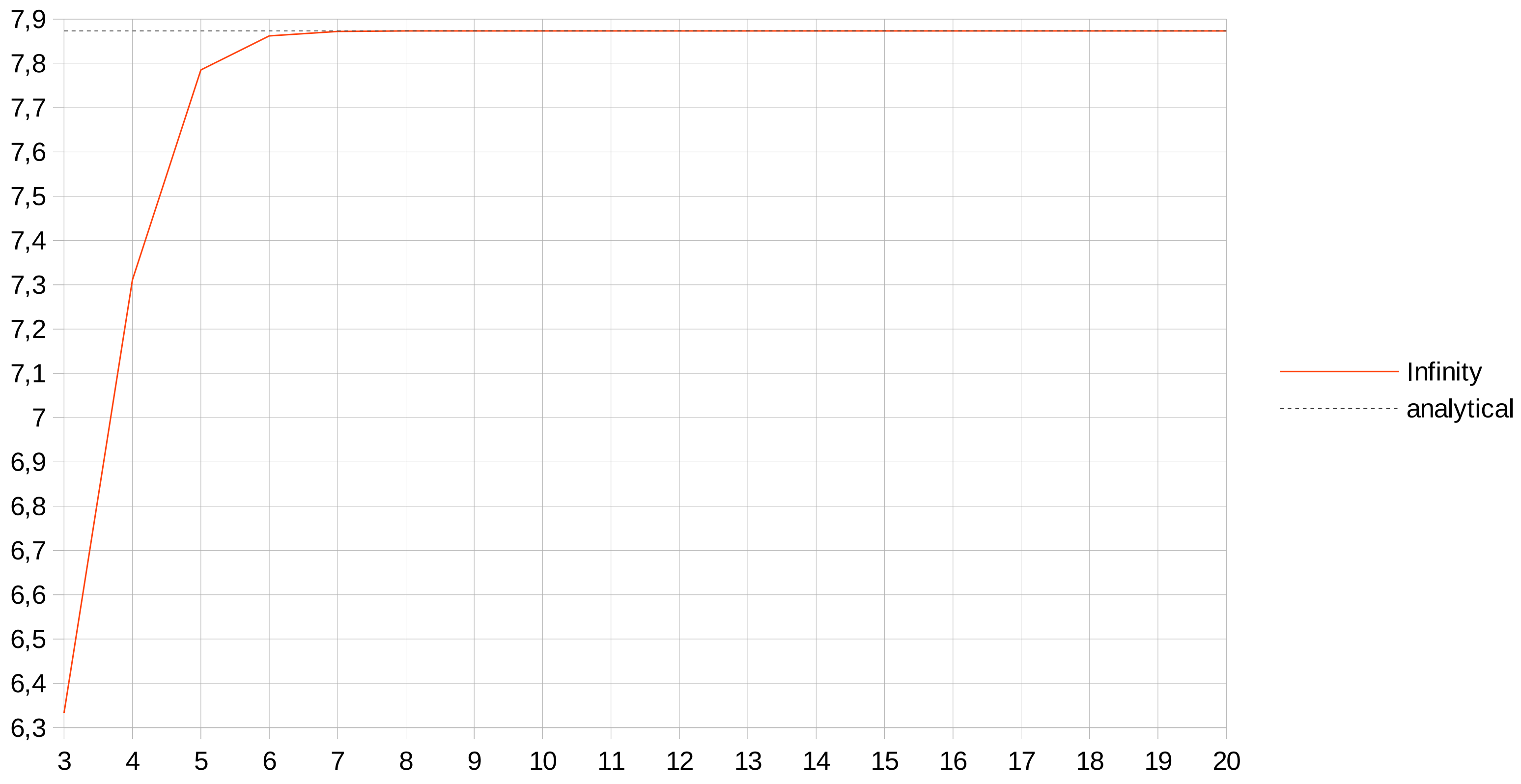
Results

In this graph we are showing the eigenvalues. As we can see, the larger number of iterations the more accurate with the analytical solution,

n	Aproximated	Analytical
2	Infinity	7,8729833462
3	6,3333333333	7,8729833462
4	7,3111111111	7,8729833462
5	7,7850119125	7,8729833462
6	7,8614730277	7,8729833462
7	7,8715157364	7,8729833462
8	7,8727968445	7,8729833462
9	7,8729596559	7,8729833462
10	7,8729803371	7,8729833462
11	7,872982964	7,8729833462
12	7,8729832977	7,8729833462
13	7,87298334	7,8729833462
14	7,8729833454	7,8729833462
15	7,8729833461	7,8729833462
16	7,8729833462	7,8729833462
17	7,8729833462	7,8729833462
18	7,8729833462	7,8729833462
19	7,8729833462	7,8729833462
20	7,8729833462	7,8729833462

This is the associated table

And here we can see the graph of our table



We can observe a great improvement from n=3 to 4.

This improvement is being reduced every step and from n=8 the approximated result is very accurate to the analytical solution, by getting closer little by little then.