**Project 2**

Computational Physics I FYS3150/FYS4150

Hyejin Yun, Yisha Chen

Oct 3, 2017

**Abstract**

We solved the eigenvalue problem of a Schrodinger’s equation of one and two particles in a harmonic oscillator. Jacobi rotation algorithm is used as a method in order to diagonlize a given matrix, which is a symmetrical diagonal matrix rewritten by a Schrodinger’s equation of one and two particles in a harmonic oscillator, and extract the diagonal to find the eigenvalues for one particle harmonic oscillator system. Increasing the range of *ρ*, where , and , decreases the eigen values. The comparison with armadillo functions and unit tests assures the reliability of the solution. Afterwards, we modify the code of the diagonal element to apply it for a two particle harmonic oscillator system which not considers Coulomb interaction between the two particles. When varies, the eigen value increases proportionally to the range of *ρ*.

## Introduction

The aim of this project is to develop a program that uses Jacobi’s method for finding eigenvalues. The Jacobi’s method is implemented to diagonalize a given matrix, which is a symmetrical diagonal matrix rewritten by a Schrodinger’s equation of one and two particles in a harmonic oscillator, while not violating the value of dot product and orthogonality of the original matrix. Armadillo library is used to manage matrix and vectors, but dynamic allocation is mainly used. By comparing the eigenvalues with the solution attained by numerical calculation, we can see how many mesh points are needed at least to receive accurate eigenvalues. The modified eigenvalue solver for two particle harmonic oscillator system shows that we can apply a general eigenvalue solver program to other systems by controlling the diagonal elements. Through varying the frequency of the harmonic oscillator it is shown that the eigenvlaues increase in a certain proportion depending on the range of distance(*ρ*). Implemented unit tests check the functionality of functions used in the program. Detail methods and algorithms are described in the following section.

## 2. Methods

**2.1. Unitary Transformation**

## 3. Results and Discussion

(c)

We went through unit tests using ‘catch for C++’. A link of the catch header file used in this program is mentioned in the appendix.

**3.1. Test case 1: max\_nondiag function**

To test the ‘max\_nondiag’ function, which returns the biggest element of a symmetric matrix among the non-diagonal elements; we set up a matrix **mat** as,

, (1)

and therefore the function must return the value 11. The macro ‘REQUIRE’ produced by catch compares the result given by the tested function and the known answer. In our case we received a result that the ‘max\_nondiag’ function returned the correct value and passed the test, which is shown in Figure 1 below.

**3.2. Test case 2: jacobi\_method function**

To test the ‘jacobi\_method’ function, which transforms a given symmetric matrix into a diagonal matrix, we set up a symmetric matrix **mat** as,

, (2)

and therefore, after the function ends, the matrix should be,

. (3)

In (3) the non-diagonal elements are calculated smaller than a specific tolerance, which is in our case,

, (4)

are approximated as 0. Using the macro ‘REQUIRE’ produced by catch, the diagonal elements generated by the function are compared with the known eigenvalues(3, 8, 6) and non-diagonal elements generated by the function are compared with the tolerance(). Therefore we use the ‘REQUIRE’ macro 9 times for each element. It is shown in Figure 1 that every assertion in each test have passed successfully.

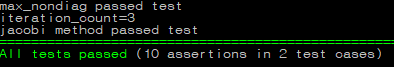


Figure 1 Output of catch unit test, both unit tests successfully passed

(d)

Figure 2 the eigenvalues of different harmonic oscillator frequency when ρmax = 8.0 and n = 200

Figure 2 is a chart of the eigenvalues of a system of two electrons in a harmonic oscillator without considering repulsive Coulomb interaction when *ρmax* = 8.0 at ground state with different : 0.01, 0.5, 1.0, and 5.0. While the eigenvalues are stable when is 0.01, 0.5, and 1.0 and are similar to the one electron eigenvalues. However, as increases to 5.0, the eigenvalues increase.

Figure 3 the eigenvalues of different harmonic oscillator frequency when ρmax = 1.0 and n = 200

In Figure 3, we decreased the size of *ρmax* to 1.0. The result shows us that at any (0.01, 0.5, 1.0, and 5.0), it is stable. We can see that the eigenvalues are approximately same to the one particle system result which means the results are more stable when *ρmax* is smaller.

Since the difference between the results of Figure 2 and Figure 3 is the size of *ρmax*, we can conclude that the eigenvalues, which are the states of possible energy, becomes larger when the range is smaller because compared to the range the strength of the oscillator is big. However, when the range is big enough, in our case when *ρmax* = 8.0, the eigenvlaues, when the strength of the harmonic oscillator is low(0.01, 0.5, 1.0,) remains stable and similar to the one particle system, while when is increased to 5.0, the eigenvalues increase largely. We assume it is because the range of the system is not small enough to calculate the eigenvalues correctly compared to the strength of the harmonic oscillator. Therefore, to solve a eigenvalue problem accurately, we should use adequate size and range.

## 4. Conclusion and Perspectives

## Appendix with extra material

Github address for full code :

Github address for Catch :

<https://github.com/philsquared/Catch/blob/master/single_include/catch.hpp>

## Bibliography

David Potter, *Computational Physics*, Imperial College, London, John Wiley & Sons*,* 1973, 82-87

W.H. Press, S.A. Teukolsky, W.T Verrerling, B.P. Flannery, *Numerical Recipies in C++: The Art of Scientific Computing*, Cambridge University Press, 1992, 463-469