PHYS 905 - Project 2

Terri Poxon-Pearson

February 27, 2017

INSERT ABSTRACT HERE

Contents

1 Introduction

In this project we will be solving Schroedinger's equation for the case of electrons in a 3D harmonic oscillator. First, we will first solve the simpler, non-interacting case with just one electron in the harmonic oscillator potential using Jacobi's method. Next, we will add a second electron with interacts via a repulsive, Coulomb potential. This will involve a change into center of mass coordinates. Both problems will be solved for l=0 case. Additionally, we will explore our calculation's sensitivity to parameters such as the number of mesh points and maximum integration radius.

This report will begin with a brief introduction to the physical system we are studying in this project, as well as a description of how these equations are discretized. Next I will describe the method used to solve this eigenvalue problem. I will discuss the implementation of this algorithm, including a discussion of sensitivity to various integration variables and tests used to check the code's validity. Finally, I will present the results of my calculations, followed by some conclusions and perspectives for future calculations.

2 Methods

2.1 Physics of Single Electron in 3D Oscillator

2.2 Discretization of Schroedinger's Equation

2.3 Jacobi's Rotation Algorithm

2.4 Physics of Interacting Electrons in 3D Oscillator

3 Code and Implementation

All of the programs, results, and benchmarks for this work can be found in my GIT repository (https://github.com/poxonpea/PHYS905). All codes for this project were written in FORTRAN.

3.1 Implementing Single Electron Case

3.2 Implementing Interacting Electron Case

3.3 Tests of Code

4 Results and Discussion

4.1 Exploring Dependence of Integration Parameters

4.2 Convergence

4.3 Computational Speeds

5 Conclusions

6 Appendices

6.1 Appendix A

Consider a basis of orthogonal basis vectors $\mathbf{v}_i$,

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \cdots \\ \cdots \\ v_{in} \end{bmatrix}$$

Orthogonality requires that

$$\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}.$$

We can apply an orthogonal or unitary transformation such that

$$\mathbf{w}_i = \mathbf{U}\mathbf{v}_i.$$

Unitarity enforces that the product of a matrix with its conjugate transpose is the identity matrix. Orthogonal matrices are a subset of real, unitary matrices. This condition implies that the product of a matrix with its transpose is the identity matrix. These two conditions can be expressed as

$$\mathbf{U}^*\mathbf{U} = \mathbf{U}\mathbf{U}^* = \mathbb{I}$$

$$\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbb{I}.$$

If we now look at the product of our transformed matrix with its transpose, we find

$$\mathbf{w}_j^T \mathbf{w}_i = (\mathbf{U}\mathbf{v}_i)^T(\mathbf{U}\mathbf{v}_j) = \mathbf{v}_i^T \mathbf{U}^T \mathbf{U}\mathbf{v}_j = \mathbf{v}_i^T \mathbf{U}\mathbf{U}^T \mathbf{v}_j = \mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}.$$

Therefore, the dot product is preserved.

References

[1] Hjorth-Jensen, Mortehn. Computational Physics, Lecture Notes Fall 2015. August 2015.

[2] Chiarandini, Marco. LU Factorization, Linear and Integer Programming. http://www.imada.sdu.dk/ marco/DM554/Slides/dm554-lu.pdf.

[3] Hjorth-Jensen, Mortehn. Introduction to Programming. Computational Physics. https://compphysics.github.io/ComputationalPhysicsMSU/doc/pub/languages/pdf/languages-minted.pdf.