,

FYS4150
Project 2, deadline October 2.

Authors: Robin D. Kifle, Sander W. Losnedahl and Vemund S. Thorkildsen

University of Oslo, Autumn 2017

Abstract

We looked at three cases involving electrons in a harmonic oscillator well. The first case is for a single electron moving in the well, the second is for two electrons moving in a well without interaction. The third is for two electrons moving in the harmonic oscillator with a repulsive coulomb interaction. The Schroedinger equation for these three cases was solved by making a Jacobi eigenvalue solver. This algorithm is extremely inefficient. It uses typically between $12n^3 - 20n^3$ flops. The algorithm is a pure eigenvalue solver, and as a cause of this it can be used in different cases with smaller modifications. The case with one electron and two non-interacting electrons have a similar solution. By looking at different values of ρ_{max}, we found that values smaller than 5 affected the solution drastically. In these cases the wave function is pushed into a to small area, altering the solution. For the interacting case we varied the oscillator potential, and found that a strong potential narrowed the probability and shifted it towards ρ_0.

Introduction

The problem we will deal with in this project is of quantum mechanical nature. As none of the three authors have had any quantum mechanics courses we will focus on the mathematical and numerical side of this problem. In this project we are going to develop our own eigenvalue-solver by using Jacobi's method. We will study two different cases, the first is for one electron moving in a harmonic oscillator. The second case is for two electrons moving in a harmonic oscillator with and without repulsive coulomb interaction. The code, and benchmarks can be found at `https://github.com/VemundStenbekkThorkildsen/Assigment2`

The article will start by tackling the mathematics and programming, i.e how the code was made and the mathematics behind it. We will then move on to the results, and discuss these to the best of our ability.

Method

To create our eigenvalue solver we first have to take a look at the matrix at hand, to get an understanding of the problem.

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2 u(\rho) = \lambda u(\rho).$$

This equation will be solved numerically, and has given eigenvalues $\lambda_0 = 3$, $\lambda_1 = 7$ and $\lambda_2 = 11$. We can use the expression for the second derivative to rewrite this. This expression is in our case given by:

$$u'' = \frac{u(\rho + h) - 2u(\rho) + u(\rho - h)}{h^2} + O(h^2)$$

The last part of this expression, namely $O(h^2$, is the truncation error and will not be used further. By using the first part, our expression now looks like this:

$$\frac{-u(\rho_i + h) + 2u(\rho_i) - u(\rho_i - h)}{h^2} + \rho_i^2 u(\rho_i) = \lambda u(\rho_i)$$

$$\frac{-u_{i+1} + 2u_i - u_{i-1}}{h^2} + \rho_i^2 u_i = \lambda u_i$$

Our h is given by $h = \frac{\rho_n - \rho_0}{n}$, as we want our ρ_i to vary with step length h. By using this h, ρ_i will take the form: $\rho_i = \rho_o + ih$. The oscillator potential is given by $(\rho_i)^2$ and will be denoted as V_i in the rest of this article. Now we have everything we need to rearrange this problem as a a matrix eigenvalue problem.

From our expression it is easy to see that the matrix we are looking for takes the negative of element $i + 1$ and $i - 1$, divided by the step length squared. It also need to take two times the positive of element i divided by h^2 plus element i multiplied by the oscillator potential V_i. This means that we are once again faced by a problem that involves a tridiagonal matrix.

$$\text{Main diagonal} = \frac{2}{h^2} + V_i, \text{ first diagonal above and below} = -\frac{1}{h^2}$$

On matrix form this looks like:

$$\begin{bmatrix} \frac{2}{h^2}+V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2}+V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2}+V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & \frac{2}{h^2}+V_{N-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \dots \\ \dots \\ u_n \end{bmatrix} = \lambda \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \dots \\ \dots \\ u_n \end{bmatrix}$$

We are going to solve this by using a Jacobi rotation algorithm. This uses a lot of similarity transformations. As long as:

$$v_j^T v_i = \delta_{ij}$$

$$U^T U = I$$

The transformations can be shown to preserve the dot product and orthogonality by multiplying with the transpose of the matrix:

$$w = Uv$$

$$w^T w = (Uv)^T Uv = v^T U^T Uv = v^T v = \delta_{ij}$$

As stated earlier in this article we are going to solve this problem by using the Jacobi eigenvalue algorithm. Consider a symmetric matrix A and Givens rotation matrix R:

$$A' = RAR^T$$

A' is symmetric and similar. A' has entries:

$$A'_{ii} = c^2 A_{ii} - 2scA_{ij} + s^2 Ajj$$

$$A'_{jj} = s^2 A_{ii} + 2scA_{ij} + c^2 A_{jj}$$

$$A'_{ij} = A'_{ji} = (c^2 - s^2)A_{ij} + sc(A_{ii} - A_{jj})$$

$$A'_{ik} = A'_{ki} = cA_ik - sA_{jk} \qquad k \neq i, j$$

$$A'_{jk} = A'_{kj} = sA_{ik} + cA_{jk} \qquad k \neq i, j$$

$$A'_{kl} = S_{kl} \qquad k, l \neq i, j$$

where $s = sin(\theta)$ and $c = cos(\theta)$ We choose θ such that $A'_{ij} = 0$ Rewrite the 3rd equation:

$$A'_{ij} = c(2\theta)A_{ij} + \frac{1}{2}s(2\theta)(A_{ii} - A_{jj})$$

And set this equal to zero:

$$tan(2\theta) = \frac{2A_{ij}}{A_{jj} - A_{ii}}$$

if $A_{jj} = A_{ii}$, then $\theta = \frac{\pi}{4}$ This is optimized by using the off-diagonal element with the largest absolute value as A_{ij} The Jacobi eigenvalue method repeats this rotation until the matrix becomes close to diagonal. In reality it checks if all the non-diagonal elements are smaller than a given tolerance. The elements on the diagonal are then approximations of the real eigenvalues of A.

The Jacobi eigenvalue method repeats this rotation until the matrix becomes close to diagonal. The elements on the diagonal are then approximations of the real eigenvalues of A.

The algorithm was developed by first creating a matrix U as shown above, letting the matrix size n run from 40 to 400 in 10 steps so that the three lowest eigenvalues produced by a later algorithm would approach 3, 7 and 11. The first algorithm/function ($maxoffdiag.cpp$) finds the maximum values of the upper triangular matrix of U and then located the k and l coordinates. The max value of the upper triangular matrix is returned.

The Jacobi rotation is defined as a function in our script and is later called upon in the *jacobi.cpp*. In this script, the matrix is rotated by and angle θ until the maximum value of the upper, and also the lower diagonal, has a lower value

than a set tolerance ϵ of 10^{-8}. When all the elements of besides the diagonal is less than ϵ, the rotation stops and the number of iterations is printed for the user to see.

The last script *lowesteigen.cpp* finds the three lowest eigenvalues with the corresponding eigenvectors. The eigenvalues and eigenvectors were saved as a txt file and implemented into a MatLab script to plot.

Two tests were implemented using user input. The first one is testing if the rotation matrix R is orthogonal so that $R^T R = I$.

Two tests were implemented using user input. The first one is testing if the rotation matrix R is orthogonal so that $R^T R = I$. This was calculated by approximately comparing $R^T R$ to I. The second test was comparing the numerically calculated eigenvalues of a matrix S to known eigenvalues. If these are approximately the same, then the test passes.

Results

We have run our algorithm for different cases and focused on different parameters. The different cases have been for two electrons with and without repulsive coulomb interaction. In the first four figures we have focused on the eigenvalues created by the eigenvalue solver. This has been plotted against the number of iterations n for 4 different values of ρ_{max}. This is compared to the known values of the three lowest eigenvalues, $\lambda_1 = 3$, $\lambda_2 = 7$, $\lambda_3 = 11$, to find a sensible value of ρ_{max}. In Figure 1 through 7, the potential is set to ρ^2

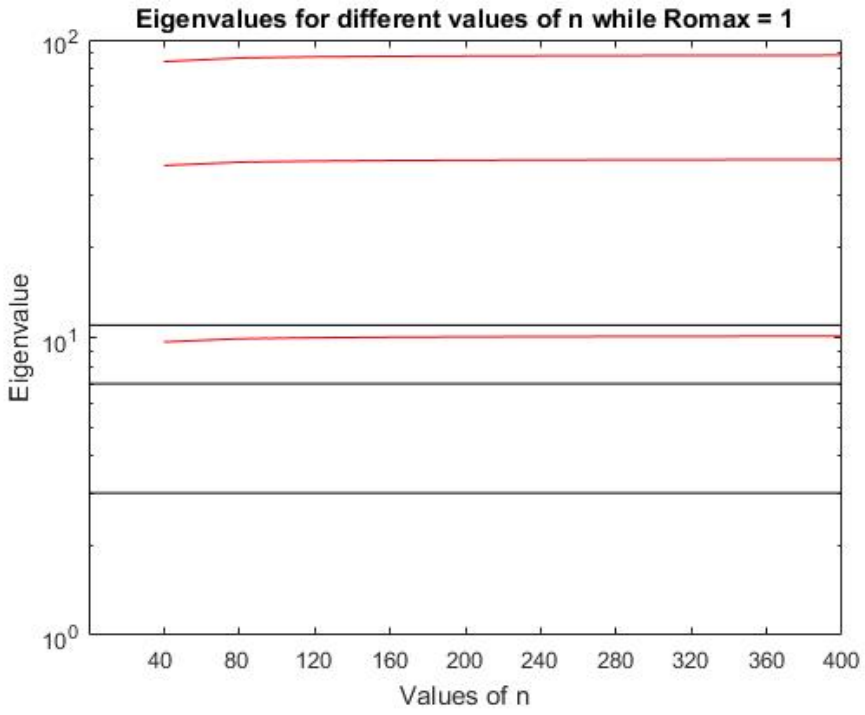
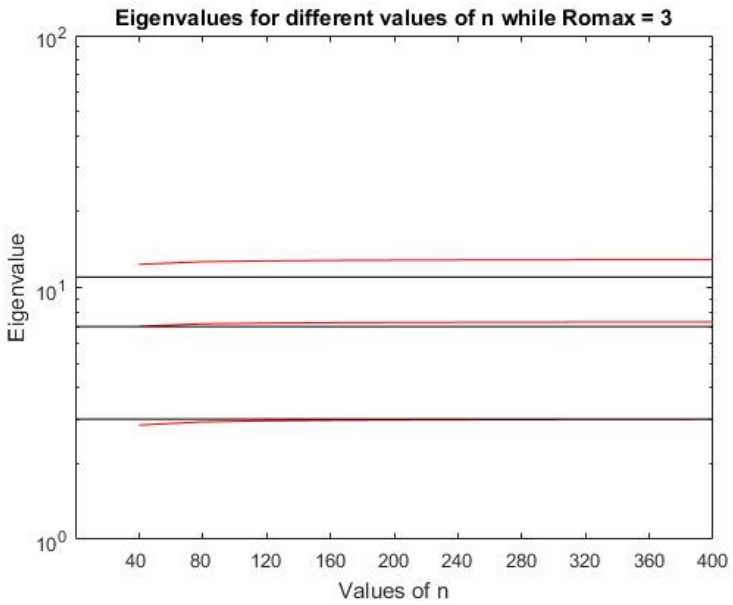
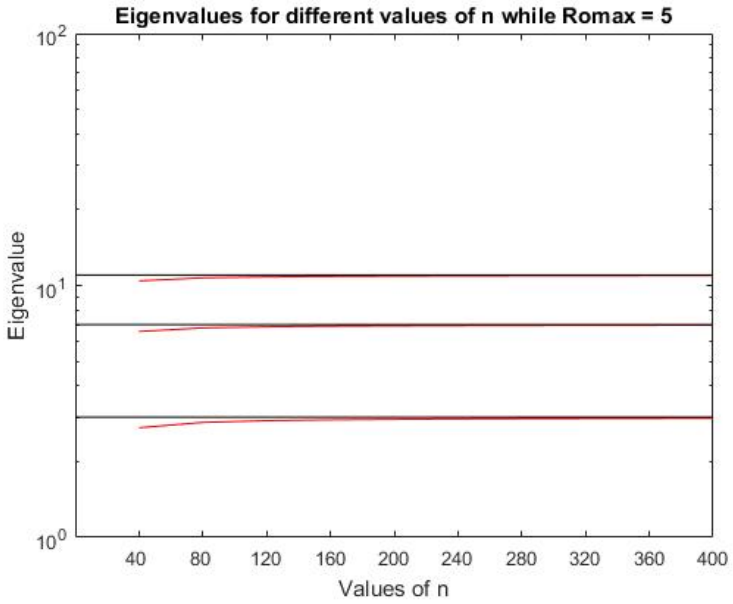


Figure − 1: Here we have chosen ρ_{max} to be 1. As the three lowest eigenvalues are known, this is obviously not a good fit.



$Figure - 2$: Here we have chosen ρ_{max} to be 3. The eigenvalues are much closer than in figure 1, but they are still not correct.



$Figure - 3$: Here we have chosen ρ_{max} to be 5. In this case all the eigenvalues converge to the correct values.

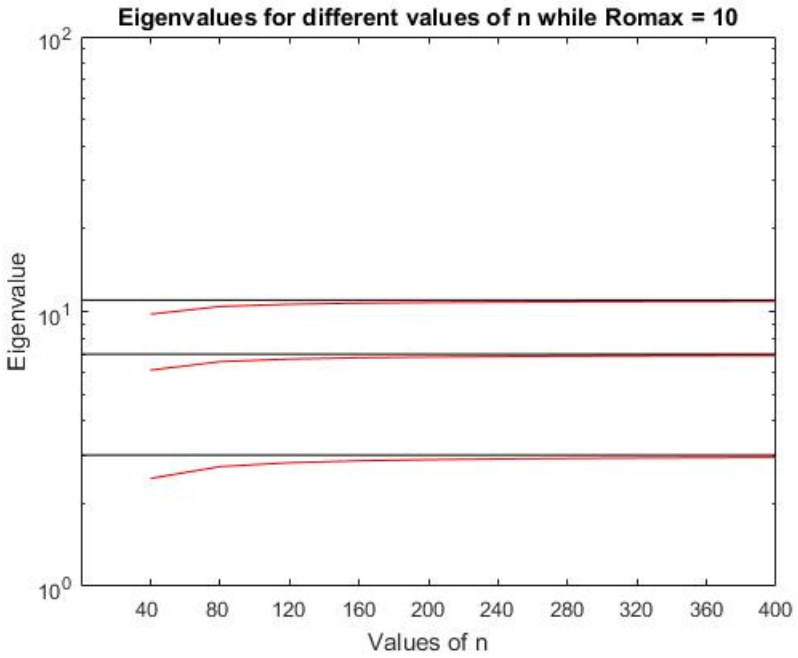
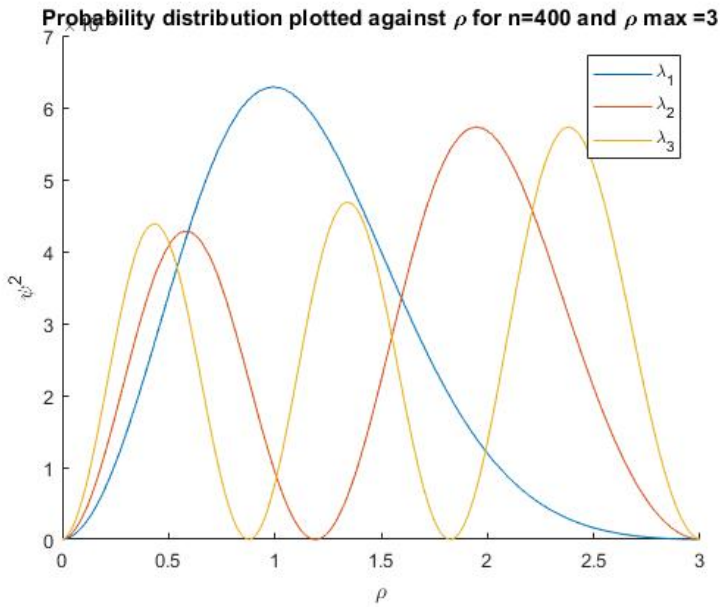
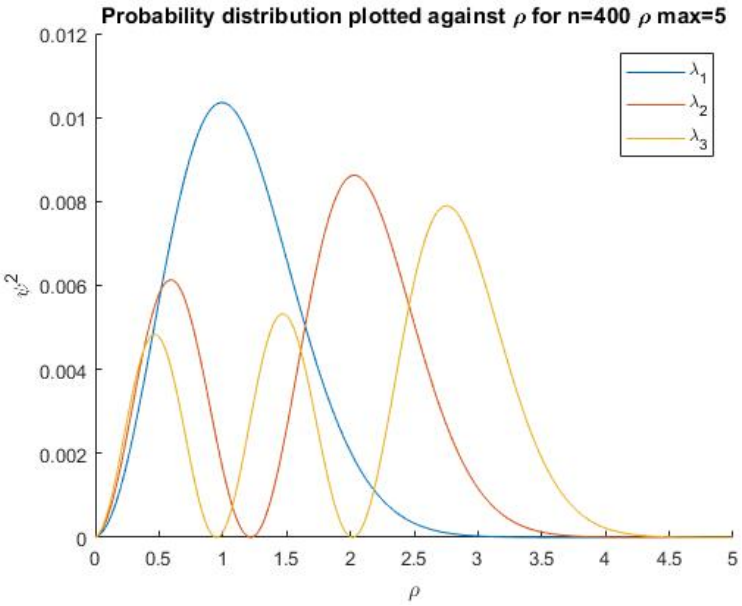


Figure − 4: Here we have chosen ρ_{max} to be 10. In this case all the eigenvalues converge to the correct values, but slower than in the case with $\rho_{max} = 5$

From these plots it is easy to see that $\rho_{max} = 1$ and $\rho_{max} = 3$ is to low(Figure 1, Figure 2), while the two larger values seem to produce the correct results (Figure 3, Figure 4). The reason for this will be discussed later in the article. Now we can plot the wave functions produced by the same runs, i.e the case with 2 non-interacting electrons.



$Figure - 5$: Here we have chosen ρ_{max} to be 3. The probability distribution is pushed into the boundaries of ρ



$Figure - 6$: Here we have chosen ρ_{max} to be 5. The probability dies out before we reach ρ_{max}. We also see that λ_1 only has one top, λ_2 has two tops and λ_3 has three tops.

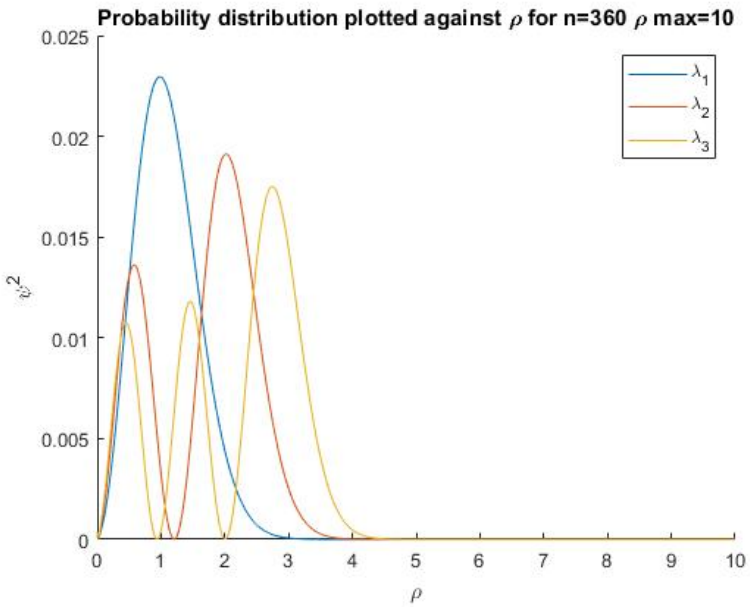


Figure − 7: Here we have chosen ρ_{max} to be 10. The wave function is of similar shape, and dies out at the approximately the same place as for $\rho_{max} = 5$.

As the wave function dies out at approximately the same place in both Figure 6 and Figure 7, the most sensible ρ_{max} will be 5.

One of the problems with the jacobi eigenvalue solver is that it uses a lot of flops, which can take a lot of time. Figure 8 demonstrates that the total number of operations grows exponentially.

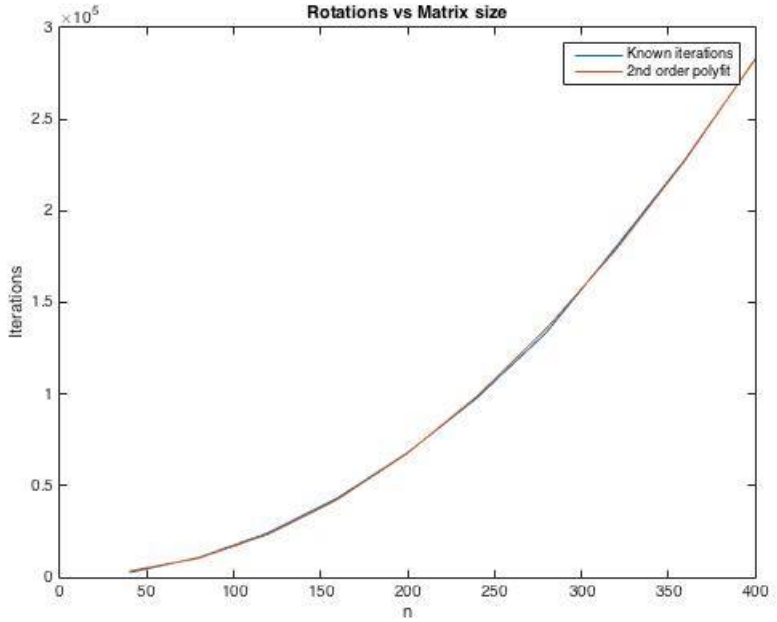


Figure − 8: This figure is showing the number of iterations plotted against n. The growth was calculated to be the second order polynomial

$$1.9x^2 - 46.5x + 2219.3$$

Figure 8 is showing an exponential growth of iterations for growing n. This means that the time needed to compute is getting extremely high for high n.

In the following figures, the potential is no longer determined by ρ^2, it is instead set to ${\omega_r}^2\rho^2 + \frac{1}{\rho}$. We will hold ρ_{max} constant equal to 5 and vary

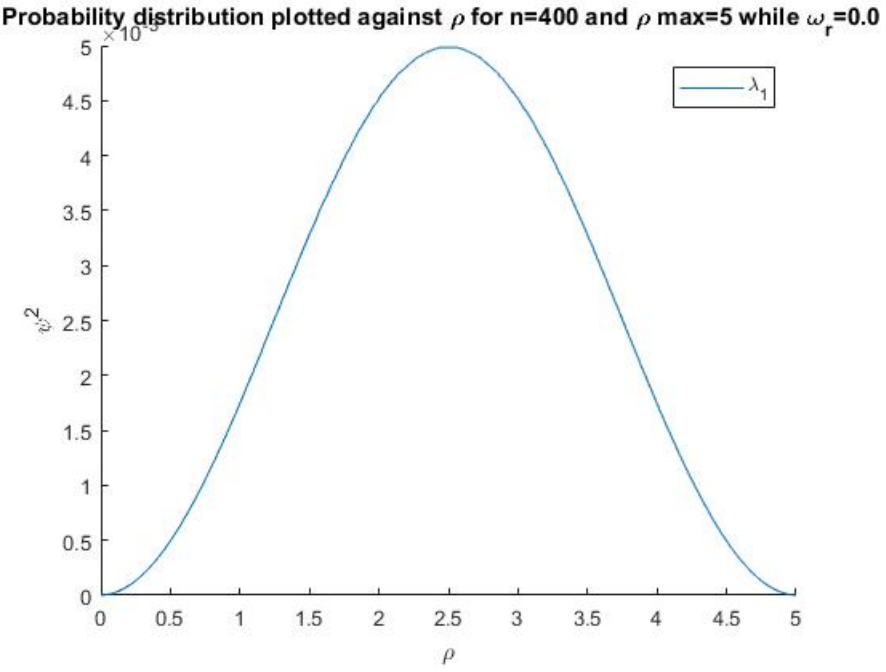


Figure – 9: The probability distribution for the ground state with $\omega_r = 0.01$ is elongated because of the weak potential.

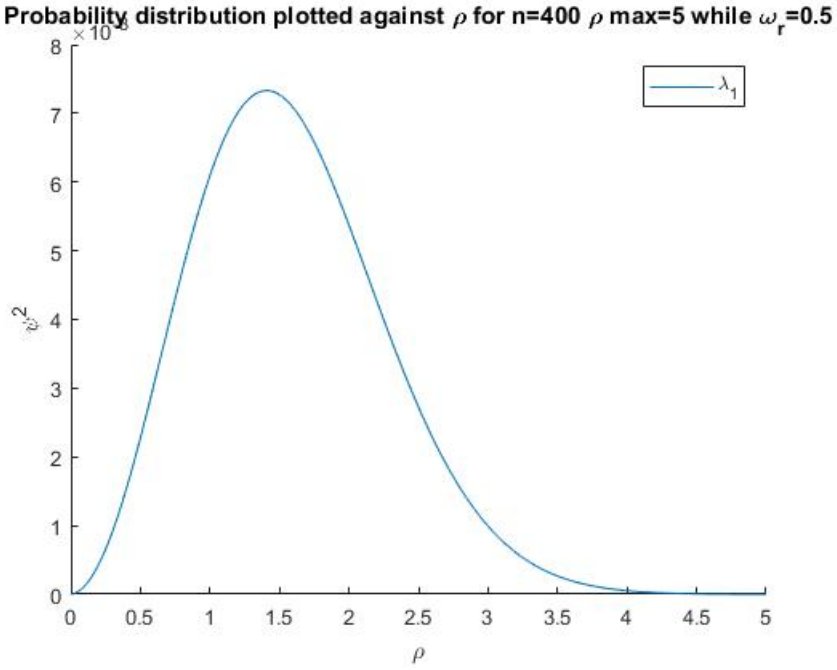
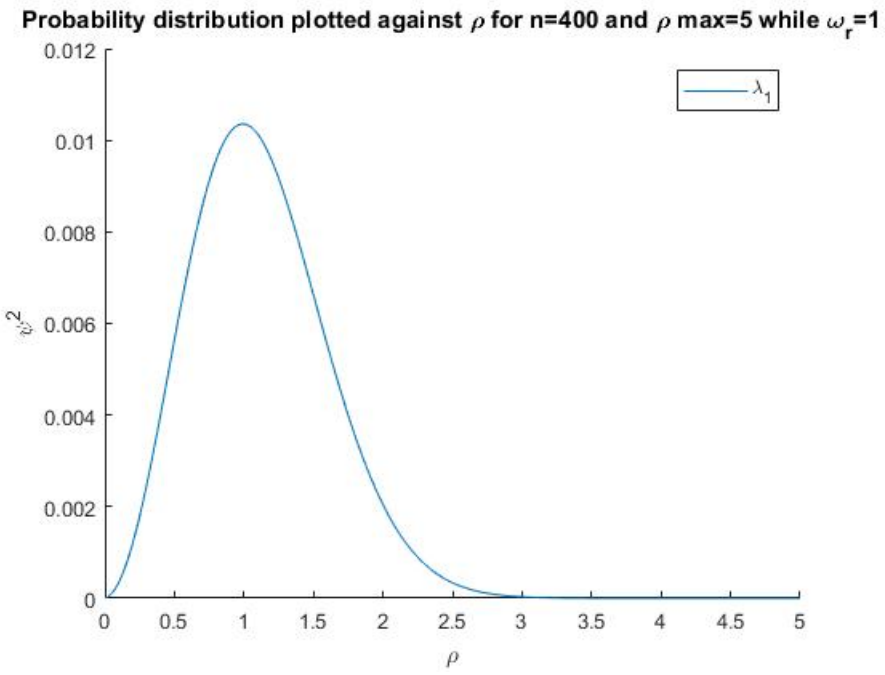
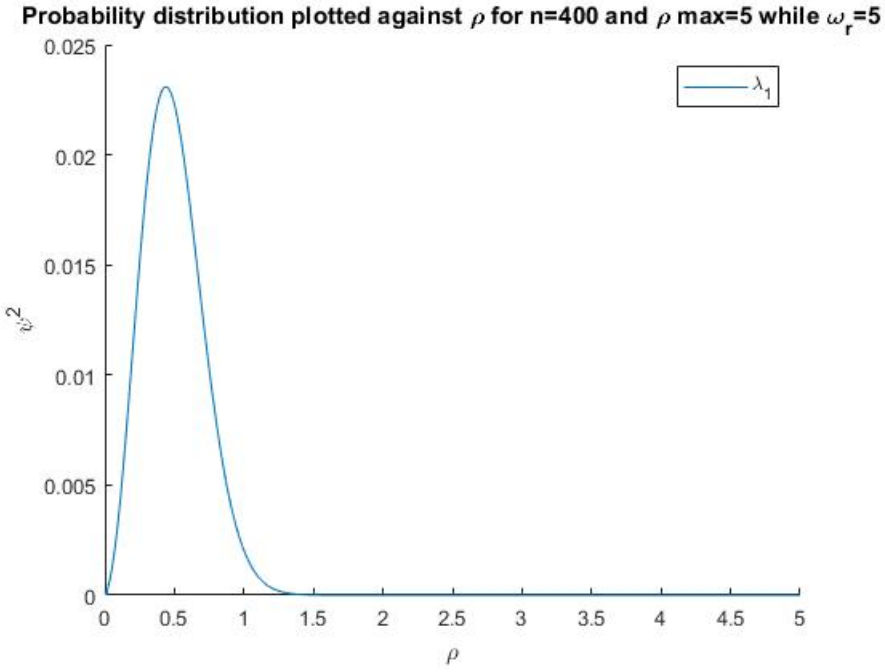


Figure – 10: The probability distribution for the ground state with $\omega_r = 0.5$



$Figure - 11$: The probability distribution for the ground state with $\omega_r = 1$



$Figure - 12$: The probability distribution for the ground state with $\omega_r = 5$

For bigger values of ω_r the probability distribution is shifted to the left

Discussion

In the first case we established a sensible value of ρ_{max}. We were given the analytic solution for the three lowest eigenvalues. By looking at Figure 1,2,3 and 4, it soon becomes clear that using $\rho_{max} < 5$ does not give the correct eigenvalues. Figure 5 shows this problem in another context. We are effectively trying to push our probability distribution into the boundaries of ρ. In figure 6 and 7 we see that the probability dies out before reaching ρ_{max}. The probability is getting lower for larger ρ but never goes completely down to zero. On these figures we are looking at three different energy levels, corresponding to the three eigenvectors made by the three lowest eigenvalues. It is also possible to see that the probability distributions corresponding to λ_2 and λ_3 naturally dies out at respectively $\rho \approx 3.6$ and $\rho \approx 4.5$. If we choose a ρ_{max} smaller than this we are, as stated earlier, pushing our probability distribution in to boundaries that are that are too small. This leads to the wrong eigenvalues shown in Figure 1 and 2.

As stated earlier in this article, the Jacobi eigenvalue solver is extremely inefficient. Even though we have not timed the algorithm, Figure 8 describes the inefficiency-problem. There is an exponential growth of iterations, and this also effects the execution time. Running the program for $n = 400$ can easily take 20 minutes. Running for $n = 900$ took us 5 hours. The reason for this extremely long calculation time is that the algorithm checks every non-diagonal to check if it is lower than the given tolerance. In our case $\epsilon = 1+^{-8}$

For the second case with two electrons, we ran the program for different values of ω_r. ω_r reflects the strength of the oscillator potential. Remember that the potential in this case is ${\omega_r}^2\rho^2 + \frac{1}{\rho}$. For weak potential the probability is elongated, that means that the probability of finding the interacting electrons is spread out over a bigger area (Figure 9). By looking at Figure 10 through 12 with increasing potential, it becomes clear that a stronger potential increases the probability of finding the electrons in a narrow window.

Concluding remarks

The Jacobi eigenvalue solver we have developed is extremely inefficient. It uses long time to calculate the eigenvalues, because it checks if every non-diagonal element is smaller than the tolerance. By studying the case with two non-interacting electrons, and varying ρ_{max}, we found that the eigenvalues were severely altered if $\rho_{max} < 5$. With small values of ρ_{max} we are effectively confining the wave function (ψ) and probability distribution(ψ^2) into a too small area. This is altering the wave function and probability distribution. In the case with two electrons interacting, we studied the effect of varying potential strength. The algorithm is the same in both cases, but the potential is different. For cases with weak potential the probability is widespread. With stronger potential, the probability is shifted towards $\rho_0 = 0$ and narrows.

References