Project 2, FYS3150

Markus Borud Pettersen

September 2019

Abstract

By implementing Jacobi's method for finding eigenvalues, and then applying it to problems from quantum mechanics, different aspects of numerical eigenvalue solvers were explored. The first of these problems consisted of a particle trapped in a harmonic oscillator potential, and the second consisted of two interacting particles, in the same potential. In all cases, the execution time of Jacobi's method went approximately as $0.4n^3$, with n being the number of calculation points. For the one-particle problem, the absolute error for the lowest lying eigenvalues was explored for different boundary conditions, and values of n. Finally, the eigenvalue of the ground state of the two-particle system, $\lambda = 1.25$, for a certain strength of the harmonic oscillator potential, was compared to findings in the literature, which showed ample agreement.

1 Introduction

Several important problems in physics can be formulated as eigenvalue problems, from the energy levels of the hydrogen atom, to mechanical stress in materials. Common to many of these real world problems, is that they often appear in complex systems, and in most cases do not have analytical solutions. As a consequence, numerical methods are needed in order to solve such problems. In this text, we will consider a popular algorithm for finding eigenvalues, namely Jacobi's method. In order to test this algorithm, we will apply it to both one- and two-particle quantum mechanical problems, and in the process study how eigenvectors and eigenvalues can give a rich description of a physical system.

To begin, we will consider so-called unitary transformations, that is linear transformations whose hermitian conjugate is equal to its inverse, and how this relates to Jacobi's method. Then we will lay out the general steps in an implementation of Jacobi's method, after which we will briefly discuss and set up the radial part of the Schrödinger equation, for the single- and two-particle cases. Finally, we will investigate the solutions of these equations, and try to quantify the numerical error which arises when solving them.

2 Theory & Algorithm

To begin, we will look into unitary transformations, and some of their important properties. One such property, is that they preserve the norm and orthogonality of the vectors they act on. To see this, we can consider a transformed vector $\vec{w}_i = U\vec{v}_i$, where U is some unitary transformation, and $\vec{v}_i$ is a normalized vector in an orthogonal set, such that $\vec{v}_i \cdot \vec{v}_j = \delta_{ij}$, for all i, j. In the Dirac notation, the inner product of two such transformed vectors is given by

$$\langle w_i | w_j \rangle = \langle v_i | U^\dagger U | v_j \rangle,$$

where $U^\dagger$ is the Hermitian conjugate of U. Thankfully, a unitary transformation can actually be defined by the fact that $U^\dagger = U^{-1}$, such that $U^\dagger U = I$, and

$$\langle w_i | w_j \rangle = \langle v_i | v_j \rangle,$$

which shows that both orthogonality and the norm of a vector is conserved when it is transformed using a unitary operator.

One very useful application for unitary transformations, is finding eigenvalues. To see how, we can consider the eigenvalue problem

$$P\vec{v} = \lambda\vec{v},$$

where P is some linear transformation whose eigenvector $\vec{v}$ has a corresponding eigenvalue λ. Now we may investigate what happens when we apply a unitary transformation U to this equation, that is

$$UP\vec{v} = \lambda U\vec{v},$$

which is not all that interesting. However, we know that for a unitary transformation, $U^\dagger U = I$, which we may freely insert into the expression above, and obtain

$$U^\dagger PU(U^\dagger\vec{v}) = \lambda(U^\dagger\vec{v}),$$

such that $U^\dagger\vec{v}$ is an eigenvector of the new transformation $U^\dagger PU$, with *the same eigenvalue* λ. Using the same reasoning, it is easy to see that we may apply a chain of such operations, and still arrive at the same eigenvalue λ.

Motivated by this, we can attempt to apply a transformation of the kind

$$U = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & 0 \\ \vdots & \dots & \cos\theta & 0 & \dots & \sin\theta \\ \vdots & \dots & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & -\sin\theta & 0 & \dots & \cos\theta \end{bmatrix}$$

where U is an $n \times n$ matrix, whose elements are given by

$$\begin{aligned} U_{ij} &= 0, \quad i \neq j, k, l \\ U_{ii} &= U_{jj} = 1, \quad i, j \neq k, l \\ U_{kl} &= -U_{lk} = -\sin\theta, \quad k \neq l \\ U_{kk} &= U_{ll} = \cos\theta, \end{aligned}$$

where i, j are the general matrix indices, while k, l are the indices of two select rows and columns, at which there are cos and sin terms. θ is in fact an angle, through which U *rotates* the elements it acts on. In fact, for the two-dimensional case, U acting on a vector $\vec{v}$ is nothing more than a rotation in the plane. Another useful feature of U, is that it is unitary, in the sense that all its elements are real, and that $UU^T = I$, where U^T denotes the transpose of U. Using the two-dimensional case as an example, it fairly simple to see why this must be so; if a vector is rotated through an angle θ by applying U, and then subsequently rotated by applying U^T, then the vector must necessarily be rotated back to where it started, as only the non-diagonal sin elements are switched by transposing U, and $-\sin\theta = \sin(-\theta)$. For less intuitive cases, one could easily compute the matrix product UU^T and find the identity matrix.

Given a unitary transformation U of this kind, we might be tempted to see whether or not we could use the transformation $U^\dagger PU$ to somehow find the eigenvalues of P. The elements of $A = U^\dagger PU$ can

be found to be

$$\begin{aligned} a_{ii} &= p_{ii} \quad i \neq k, i \neq l \\ a_{ik} &= p_{ik}c - p_{il}s \quad i \neq k, i \neq l \\ a_{il} &= p_{il}c + p_{ik}s \quad i \neq k, i \neq l \\ a_{kk} &= p_{kk}c^2 - 2p_{kl}cs + p_{ll}s^2 \\ a_{ll} &= p_{ll}c^2 + 2p_{kl}cs + p_{kk}s^2 \\ a_{kl} &= (p_{kk} - p_{ll})cs + p_{kl}(c^2 - s^2), \end{aligned} \tag{1}$$

where c and s are short-hand for $\cos\theta$ and $\sin\theta$, respectively. Luckily, the transformed matrix is also symmetric, so we may set the remaining elements $a_{ji} = a_{ij}$. In order to obtain the eigenvalues of P, we can imagine that we can apply subsequent transformations to P, in order to obtain a diagonal matrix

$$D = S_n^T S_{n-1}^T ... S_1^T P S_1 ... S_{n-1} S_n,$$

whose eigenvalues, as we have argued, must be the same as those of P. However, since D is diagonal, its eigenvalues are given by its diagonal entries d_i, and its eigenvectors are simply the corresponding basis vectors e_i, as

$$D\vec{e}_i = d_i\vec{e}_i,$$

but it should be specified that these eigenvectors are generally not eigenvectors of P, also. In other words, if we can diagonalize P, finding its eigenvalues is as simple as reading off a set of diagonal elements. To do so, however, we will need a strategy for eliminating all non-diagonal elements. One way of doing so, is by setting the angle θ in our transformation U, such that the non-diagonal elements $a_{kl} = a_{lk}$ from (1) become zero. Luckily, (1) tells us that we only need to find $t = \tan\theta$, such that

$$t^2 + 2\tau t - 1 = 0, \tag{2}$$

where $\tau = (p_{ll} - p_{kk})/(2p_{kl})$. The solution of (2) is given by $t = -\tau \pm \sqrt{1+\tau^2}$, and selecting the smaller of these solutions (by absolute value), i.e. $t = -\tau + \sqrt{1+\tau^2}$ when $\tau > 0$, and $t = -\tau - \sqrt{1+\tau^2}$ otherwise, ensures that our method converges [1]. One important point to make here, is that the form given for the solution of t here is not numerically ideal, as we might end up with subtraction of two almost identical terms for large τ. Therefore, to avoid round-off errors of this kind, we can rather use

$$t = \frac{\text{sign}(\tau)}{\text{sign}(\tau)\cdot\tau + \sqrt{1+\tau^2}},$$

which can be shown by multiplying each solution of t with its respective conjugate (and converting it to a fraction, in the process). Note that this expression does not involve subtraction of similar terms in any way.

In order to finalize the algorithm, we require one final puzzle piece, and that is a method for selecting the special elements p_{kl}. As our goal is to minimize the non-diagonal elements, we simply select the largest non-diagonal element of the matrix in question. However, judging from (1), zeroed-out non-diagonal elements may be become non-zero upon a new iteration, and therefore we will most likely not be able to remove the non-diagonals completely. In stead, we repeat the algorithm until

$$\max(|a_{ij}|) \leq \varepsilon, \quad i \neq j$$

where ε is some set tolerance. At this point, we have actually arrived at Jacobi's method for finding eigenvalues. For a more exhaustive exploration of this method, refer to [1], for example.

One of the most interesting applications of eigenvalue solvers such as Jacobi's method, is finding the eigenvalues of quantum systems. Such systems obey the famous Schrödinger equation

$$H\psi = E\psi, \tag{3}$$

where H is the Hamiltonian of the system, ψ an eigenstate, and E the associated energy eigenvalue of said state. Determining the behaviour of a system is then only a matter of determining the eigenstates of the operator H. Looking at this from a more technical point of view, we can see that this just means that we for a given matrix H, need to find its eigenvalues, and eigenvectors. In this text we will consider a special case of (3), in which a particle is trapped in a harmonic oscillator potential, $V(r) = 1/2 \cdot m\omega^2 r^2$, where m is the mass of the particle, and ω the oscillation frequency. To further simplify things, we will only consider the radial part of the resulting equation, which then reads as

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(\frac{1}{2}m\omega^2 r^2 + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r),$$

where $\hbar$ is the reduced Planck constant, $u(r)$ a radial eigenstate, and l the orbital angular momentum quantum number. In order to simplify things, we will only consider the case where $l = 0$, such that

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \frac{1}{2}m\omega^2 r^2 = Eu(r).$$

In order to generalize this equation somewhat, we can introduce the dimensionless quantity $\rho = r/\alpha$, where α is some arbitrary constant with dimension length. By rearranging a bit, we find that

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{m^2\omega^2\alpha^4}{\hbar^2}\rho^2 = \frac{2m\alpha^2}{\hbar^2}Eu(r),$$

which motivates us to select α such that

$$\alpha^2 = \frac{\hbar}{m\omega},$$

which finally leads us to the dimensionless equation

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2 u(\rho) = \lambda u(r), \tag{4}$$

where we define the eigenvalue

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E = \frac{2}{\hbar\omega}E.$$

Note that the harmonic oscillator energy eigenvalues, E_n, are given by

$$E_n = \hbar\omega(2n + l + \frac{3}{2}),$$

[2]but we are restricting ourselves to the case where $l = 0$, and given our chosen scaling, we find that the analytical eigenvalues are

$$\lambda_n = 4n + 3. \tag{5}$$

Tantalizingly, (4) is nothing more than a second order differential equation, which is easily discretized. For the second derivative, we may approximate it using a second-order central difference

$$\frac{d^2u}{d\rho^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2},$$

where u_i is the function u evaluated at a discretized point ρ_i, while

$$h = \frac{\rho_{max} - \rho_{min}}{N}$$

is the step size with which we hope to approximate u, given by the maximum and minimum values of ρ, as well as the number of calculation points N we wish to use. Using the same logic for the remaining terms, we can discretize (4) as

$$-\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\rho_i^2 u_i=\lambda u_i, \tag{6}$$

For our problem, we know that ρ is related to the distance our particle resides from the origin. As such the smallest value of ρ, ρ_{min} must necessarily be zero for all cases. ρ_{max} is slightly more tricky, as we can imagine that our particle could tend towards being infinitely far away, and that $\rho_{max} \to \infty$. While this is not a problem mathematically, it is most certainly a problem numerically, as $\rho_{max} \to \infty$ would require either that $h \to \infty$ or that we would need an infinite amount of calculation points N in order to maintain numerical precision.

This means that we have to approximate infinity, be setting it to some finite value, and later on we will see that this greatly influences the accuracy of the resulting eigenvalue calculations. The keen reader may have already noticed, but if we consider setting up (6) for all indices i simultaneously, we can actually write it as a matrix equation

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \dots & \dots & \dots & \dots & e_{N-2} & d_{N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ \dots \\ \dots \\ u_{N-1} \end{bmatrix}, \tag{7}$$

where the diagonal elements become $d_i = 2/h^2 + \rho_i^2$, and the non-diagonal elements are simply $e_i = -1/h^2$. Note how we are not including u_0 and u_N, as the wave function of a physical particle has to vanish at the boundaries. If this were not the case, the probability of finding the particle *somewhere* (given by the integral of the squared magnitude of the wavefunction, over all space), could be more than 100 %, which is nonsensical. In other words, $u_0 = u(0) = 0$, and $u_N = u(\rho_{max}) = 0$, where ρ_{max} is used as an approximation to infinity.

Now that the Schrödinger equation has been rephrased as a matrix eigenvalue problem, we can apply Jacobi's method to (7), in order to find these eigenvalues. Note, however, that we have scaled our equation, and any eigenvalues we find must be appropriately re-scaled, in order to find actual solutions to the regular Schrödinger equation.

As a last exercise, we can consider the case of two particles caught in a harmonic oscillator potential, that also interact through a Coloumb potential. For brevity, we will not outline the complete derivation for the scaled, radial Schrödinger equation in this case, but rather state the result

$$-\frac{d^2}{d\rho^2}u(\rho)+\omega_r^2\rho^2 u(\rho)+\frac{1}{\rho}u(\rho)=\lambda u(\rho), \tag{8}$$

where $\lambda = \frac{m\alpha^2}{\hbar^2}E$, with a slightly different scaling term $\alpha = \hbar^2/(m\beta e^2)$, where $\beta e^2 = 1.44$ eVnm is due to the repulsive Coloumb potential (and e is the elementary charge, so we are in fact considering two particles of the same charge). $\omega_r = m\omega/(2\hbar)\alpha^2$ is, as we can see, a factor reflecting the strength of the harmonic oscillator potential. For a more complete derivation, see [2].

3 Methods

Jacobi's method, as detailed in the previous section, was implemented in c++, in order to ultimately determine the eigenvalues of a quantum mechanical problem. This was accomplished merely by implementing the steps outlined in the algorithm description, and creating a simple algorithm to find the maximum non-diagonal element, by absolute value. The maximum non-diagonal element was found quite simply be traversing the elements of the provided matrix, and comparing its absolute value to previously greatest value, and storing the maximum element, along with its indices. See appendix for additional details on the programs.

In order to verify the algorithm's integrity, several unit tests were devised. The first of these unit tests was a simple check of the setup of a tridiagonal matrix, by way of determining its eigenvalues. The test itself was performed by using an external eigenvalue solver, from the Armadillo c++ linear algebra library (available at http://arma.sourceforge.net/), to determine the eigenvalues of a sample 30×30 tridiagonal matrix with diagonal elements $d_i = 2/h^2$, and non-diagonal elements $e_i = -1/h^2$. For this test case, there exists analytical eigenvalues

$$\lambda_i = d_i + 2e_i \cos\left(\frac{i\pi}{N+1}\right) \quad i = 1, 2, ...N,$$

when the boundary conditions are taken to be $u(0) = u(1) = 0$ [2]. If the matrix is set up correctly, one would expect that these are the eigenvalues found, and as a measure of unit test success, the mean absolute difference between the exact and calculated eigenvalues was required to be smaller than a tolerance $\varepsilon = 1 \cdot 10^{-10}$.

In order to see that the Jacobi algorithm functioned as expected, the Frobenius norm of a tridiagonal matrix was calculated before and after it had been diagonalized using Jacobi's method. This exploits the fact that unitary transformations should not change the (Frobenius) norm. The 30×30 tridiagonal test matrix featured diagonal elements $d_i = 2/h^2$ and non-diagonal elements $e_i = -1/h^2$, where the stepsize h was taken to be $1/N$. The test success measure, was that the Frobenious norm had changed less than a tolerance $\varepsilon = 1e^{-10}$.

Another test was conducted to verify that the maximum non-diagonal elements were indeed found. This was done very simply by unleashing the algorithm on a 30×30 matrix with pseudorandom elements, generated by the external Armadillo library, and then comparing the algorithms findings with that of the maximum-element function of the external library. Note that for the external maximum element function, the diagonal elements of the random matrix were set to zero, and the absolute value was taken of all remaining matrix elements. The test success measure was that the difference in the maximum element value had to be less than a tolerance $\varepsilon = 10^{-10}$.

The final unit test consisted of comparing eigenvalues found using Jacobi's method, to those found using the Armadillo library's eigenvalue solver. As before, the test matrix was a 30×30 tridiagonal matrix, whose diagonal elements were given as $d_i = 2/h^2$, and the non-diagonal elements $e_i = -1/h^2$, where h was taken to be $h = 1/N$. The success measure was that the mean difference between the computed eigenvalues was less than a tolerance $\varepsilon = 10^{-10}$. Note that Jacobi's method does not provide an ordered set of eigenvalues, and the eigenvalues found using Jacobi's method were sorted before comparison.

In order to determine the error induced in eigenvalue calculations as a function of the infinity approximation ρ_{max} and the number of calculation steps N, the eigenvalues found using Jacobi's method, for the one electron quantum case were compared to the lowest analytical eigenvalue in (5), $\lambda = 3$, for a grid of n and ρ_{max} values. In this case, the tridiagonal matrix was initialized with diagonal elements $d_i = 2/h^2 + \rho_i^2$, and non-diagonal elements $e_i = -1/h^2$. For each combination of n and ρ_{max}, h was taken to be $\rho_{max}/(n+1)$, and the absolute error was logged. The n values ranged from $n = 5$ to $n = 355$ in steps of 35, and ρ_{max} ranged from $\rho_{max} = 1$ to $\rho_{max} = 20$ in steps of two.

The same error calculations were also performed for the mean absolute error for the five lowest eigenvalues found using Jacobi's method, in order to find values of n and ρ_{max} which might give useful results for a range of eigenvalues.

Finally, a two electron quantum case was studied. This was done by finding the eigenvalues of the tridiagonal matrix with diagonal elements $d_i = 2/h^2 + \omega_r^2\rho_i^2 + 1/\rho_i$, and non-diagonal elements $e_i = -1/h^2$, where $h = \rho_{max}/(n+1)$. Based on the findings from the error analysis of the lowest energy eigenstates, n was chosen to be 350, and ρ_{max} to be 8. Using the external Armadillo library, the eigenvectors, corresponding to quantum eigenstates, of the tridiagonal matrix were also found, and compared to analytical findings from the literature. In order for these states to correspond to eigenstates of the Hamiltonian, they were first normalized by dividing by their norm, which was computed using trapezoidal integration over all ρ.

4 Results

Fig. 1 shows the measured execution time for the Jacobi rotation algorithm, as a function of calculation points n ranging from $n = 5$ to $n = 400$. Also shown is a polynomial fit of the execution time curve, of degree 3. We can observe that the computation time goes approximately as $0.4n^3$. Note that the execution time shown was logged without vectorization.

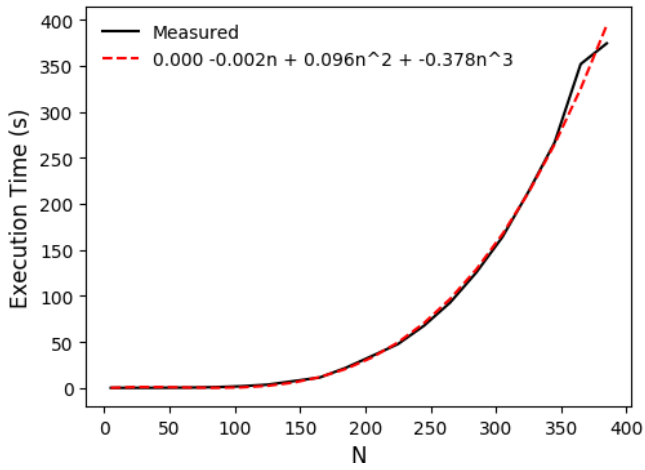

Figure 1: Execution time for the Jacobi algorithm, as a function of the number of calculation points, N. Also shown is a polynomial fitting of the same curve, of degree 3.

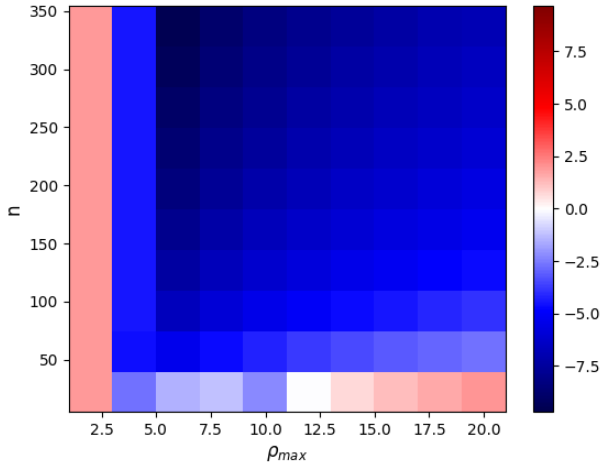

Figure 2: Computed absolute error (natural log) of the lowest lying eigenvalue, $\lambda = 3$, for the scaled, radial Schrödinger equation for a single particle in a harmonic oscillator potential. Eigenvalues were found using Jacobi's method, and the error is shown for a grid of combinations of calculations points n and infinity approximations ρ_{max}.

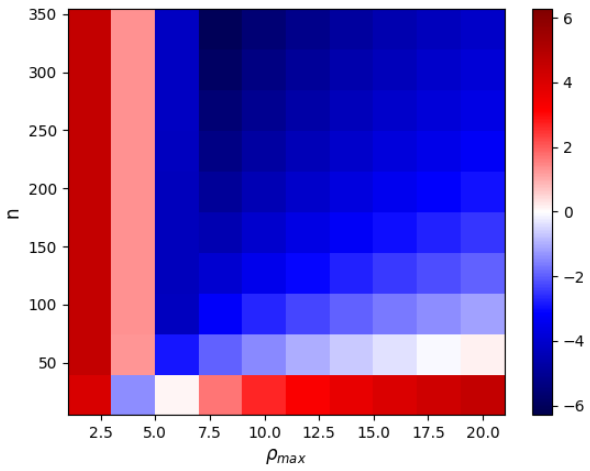

Figure 3: Computed mean absolute error (natural log) of the 5 lowest lying eigenvalues, $\lambda = 3, 7, 11, 15$ and 19, for the scaled, radial Schrödinger equation for a single particle in a harmonic oscillator potential. Eigenvalues were found using Jacobi's method, and the error is shown for a grid of combinations of calculations points n and infinity approximations ρ_{max}.

Fig. 2 and 3 show the computed (natural log) error surface for eigenvalues of the single particle case, for the scaled, radial Schrödinger equation, found using Jacobi's method. Fig. 2 shows the absolute error of the lowest lying eigenvalue, $\lambda = 3$, while Fig. 3 shows the meaned absolute error, for the five lowest lying eigenvalues, $\lambda = 3, 7, 11, 15$ and 19. For $\lambda = 3$, we can see how a rather low value of ρ_{max}, approximately 5-6, combined with a high number of calculation points, $n \geq 300$, yields the lowest error.

For the (natural log) mean absolute error made for the first five eigenvalues, we can see that the lowest error is found for a slightly higher value of ρ_{max} around 8, and values of n greater than 300.

Fig. 4 shows the squared probability amplitude of normalized wavefunctions of the two-particle system, for different values of w_r. For low values of ω_r, corresponding to weak oscillator potential, the probability of finding the particle is greatest further away from the origin. In the extreme case of $\omega_r = 5$, the probability of finding the particle upon measurement, is greatest very close to the origin, peaking at about $\rho = 0.5$. We can also observe that width of the particle's wavefunction decreases with increasing strength of the harmonic oscillator potential.

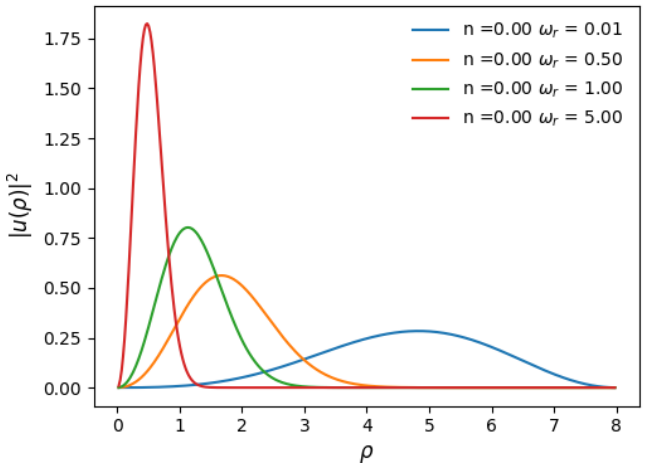

Figure 4: Squared amplitude of eigenstates for the scaled Schrödinger equation, for the case of two interacting particles in a harmonic oscillator potential. All wavefunctions shown are in the ground state $n = 0$, but with different values of w_r, reflecting the strength of the harmonic oscillator potential. $u(\rho)$ is the radial wavefunction, while ρ is a radial coordinate.

Fig. 5 Shows the squared probability amplitude, $|u(\rho)|^2$ of a ground state for the scaled Schrödinger equation, in the two-particle case. Also plotted is an analytical solution[3] $u(\rho) = re^{-r^2/8}\,(1 + r/2)$, of the same equation, for the same value of $\omega_r = 0.25$. The analytical solution was, however, normalized on the shown interval of ρ-values. After normalization, one can see that there is excellent agreement between the analytical and numerical solutions. Note that all numerical eigenstates were found using the external Armadillo library function *eig_sym*.

Finally, table 1 shows the eigenvalues found using Jacobi's method, for the two-particle, case, for different strengths of the harmonic oscillator potential. Also indicated is one eigenvalue for the same problem and state, obtained from [3], once again showing excellent agreement.

Table 1: Eigenvalues for the lowest energy state, $n = 0$ for the two-particle case of the scaled radial Schrödinger equation, for various harmonic oscillator potential strengths, as reflected by the size of ω_r. *Also indicated is the approximate eigenvalue from [3] for the same case where $\omega_r = 0.25$, but it should be noted that this author uses a different scaling term, and that the value listed has been multiplied by a factor of two.

Eigenvalue	ω_r	M. Taut [3]
0.424	0.01	-
1.25	0.25	1.250*
2.23	0.5	-
4.05	1	-
17.4	5	-

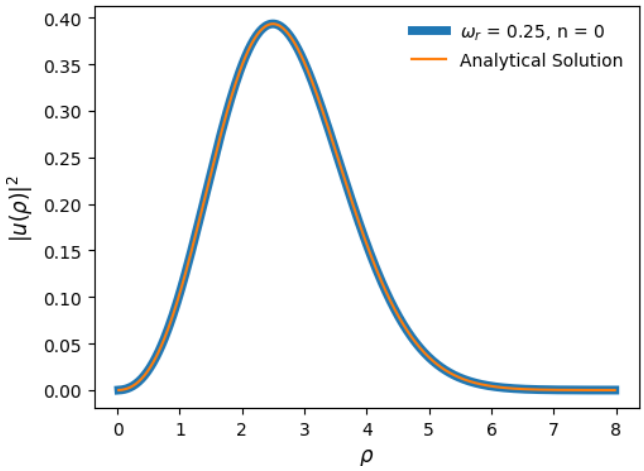

Figure 5: Squared amplitude of an eigenstate for the scaled Schrödinger equation, for the case of two interacting particles in a harmonic oscillator potential. Also shown is an analytical eigenstate of the same problem, $u(\rho) = re^{-r^2/8}(1 + r/2)$, obtained from [3].

5 Discussion

For all unit tests, each was passed to a tolerance of 10^{-10} before the main calculations were performed. However, this was only true for matrices of size $n \times n$, where $n < 50$, as the norm of the tridiagonal failed to be preserved to this tolerance for larger matrix sizes. It is not entirely clear why this is so, except for the fact that Jacobi's method involves many floating point operations, roughly on the order of n^3, if we take the timing of Fig. 1 at face value. At each operation, some loss of numerical precision may occur, which might help to explain why the norm is not preserved. The same lack of precision was also found for the eigenvalue test, as the eigenvalues found using Jacobi's method did not match those found using Armadillo library functions within the set tolerance. Both of these might hint that there is room for improvement in either the implementation of, or in Jacobi's method itself, as far as numerical precision is concerned. An attempt was made to lower the minimum value for the non-diagonal elements (which are to be eliminated) in Jacobi's method, but this did not help to improve the problem.

Besides telling us something about the number of floating point operations, the execution time is

also a useful marker for algorithm integrity. While we found that the execution time could be fitted as going like $0.4n^3$, [1] reports that it typically goes like $12n^3 - 20n^3$ in total, which is at least, in the same region.

As for the error surfaces, they provide a useful way of visualizing the possible combinations of ρ_{max} and n, and how they affect calculation error. It also seems reasonable that a lower eigenvalue would require a lower value of $\rho_{max} \approx 6$, and that the mean error over several somewhat higher eigenvalues decreases if ρ_{max} is increased to match them. It also seems that choosing ρ_{max} to be smaller than the eigenvalues we are looking for, creates consistently poor results, which is seen by the uniformly shaded columns in both Fig. 2 and 3. It could have been advantageous to create error surfaces for higher, single eigenvalues also, not just the mean over several. On the other hand, as we have already seen, Jacobi's method is somewhat slow, and creating an error surface on a large grid can be very time-consuming. In all, it would seem that using the mean error over several low lying eigenvalues is an appropriate compromise, as it gives us useful ρ and n values for a range of eigenvalues.

As shown in Fig. 5, we can be fairly certain that our eigenstates match those found in the literature, at least as far as its shape is concerned. It could have been useful to quantify the difference between these two results, but as a simple measure of solution integrity, it should suffice. As for the other eigenstates, it is interesting to note that we might be able to make some sense of our results. For a strong harmonic oscillator potential, a particle seems to be much more likely to be close to $\rho = 0$, i.e. closer to the other particle, as we might expect. For a very strong oscillator potential, the particles are brought very close, and their position is more sharply defined. It could therefore be very interesting to also investigate the momentum eigenstates of the same system, as we would expect that they would be less sharply defined under the same potential, due to the uncertainty principle.

It could also have been interesting to plot higher energy eigenstates in order to see how interactions affect systems at higher energies. On the other hand, this would be an interesting case for future enquiries, especially if the model is fleshed out, by taking more effects into account, such as real-world boundary conditions, or the motions of the center of mass, or even the time-evolution of the system.

6 Conclusion

By implementing Jacobi's method for finding eigenvalues, several aspects of numerical eigenvalue problems have been explored. Mainly, these have been formulated as quantum mechanical problems, and solutions of a scaled Schrödinger equation, for a particle subject to a harmonic oscillator potential. In addition to solving physical problems, several unit tests were also devised, in order to ensure algorithm integrity. Also, the execution time of the program was investigated as a function of the number of calculation points n, and found to go approximately as $0.4n^3$, which is somewhat in keeping with findings in the literature.

In the simplest quantum case, only one particle was present, and an error analysis for the 5 lowest lying eigenvalues found using Jacobi's method, showed that $n > 300$ calculation points, and the infinity approximation $\rho_{max} \approx 8$ yielded the lowest absolute error, while a smaller eigenvalue required a similar amount of calculation points, but a lower $\rho_{max} \approx 6$ to achieve the lowest possible error. In the future, one could possibly conduct such an error analysis for more individual eigenvalues, possibly using a finer grid of ρ_{max} and n values.

As a development, a two-particle quantum system was also considered, and the radial part of the Schrödinger equation was solved by finding its eigenvalues using Jacobi's method, and determining its eigenvectors using the external Armadillo linear algebra library. These eigenstates were then normalized. Their squared magnitude revealed that particles in such a system are more likely to be found close to each other, if the harmonic oscillator potential is strong, and less likely to be found close to each other otherwise. It also appeared that the their positions became more sharply defined when the harmonic oscillator potential was stronger. Finally, a ground state of this system, and its corresponding eigenvalue, $\lambda = 1.25$ was compared to findings in the literature, showing excellent agreement.

As an interesting case for future developments, more detail, such as exotic boundary conditions could be added to the two-particle case. It would also be interesting to explore momentum eigenstates of the same system, to see if the uncertainty relation would become apparent as the particle's position became more sharply defined for greater oscillator potentials. Finally, higher energy eigenstates could also be investigated, to see if these behave differently from low energy ones, due to particle interactions.

7 References

References

[1] M. Hjort-Jensen, "Jacobi's Method", in Computational Physics - Lecture Notes Fall 2015, Oslo: Department of Physics, University of Oslo, 2015, p. 215-218.

[2] M. Hjort-Jensen, "Project 2", Computational Physics I FYS3150 - Project 2, Oslo: Department of Physics, University of Oslo, Sep. 18, 2019

[3] M. Taut, "Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb correlation problem", Physical Review A, vol. 48, no. 5, pp. 3561-3566, 1993.

8 Appendix

All code is freely available at https://github.com/markusbp/fys3150/tree/master/project2 and contains the following programs:

- `harmonic_oscillator/harmonic_oscillator.cpp : Main program, contains methods for computing one- and two-particle problems, finding a specific error surface, and timing.`
- `conduct_unit_tests/unit_tests.cpp : Files for running unit tests.`
- `tools.cpp/tools.hpp : File/header containing all unit test functions, as well as filewriting tool.`
- `jacobi_rotation.cpp : File/header with implementation of Jacobi's method.`
- `plot_device.py : Python file for plotting in matplotlib.`
- `figures : Contains select figures, showing results of calculations`
- `results : Not added to Github to save space, but originally contains result files, csv.`