Eigenvalue Problems:
From Buckling Beam to Schroedinger's equation in three dimensions
FYS3150: Computational Physics

Sigurd Sandvoll Sundberg*
Department of Geosciences, University of Oslo

(Dated: September 30, 2020)

In this project we study eigenvalue and eigenvector problems numerically using Jacobi's method and Bisection method[3]. We study two different physical cases: The buckling beam and quantum dots with one electron. The system are quite different, but as eigenvalue problems their equations are almost identical. We discretized the infinite dimensional problem, to a finite problem which we could solve with linear algebra. We used Jacobi's method to find the eigenvalue and eigenvectors for both the buckling beam and the quantum dots problem, and found that the eigenvectors and eigenvalue where could be approximated with a low number of integration points. Jacobi's method is not the best method for tridiagonal Toeplitz matrix as it needs $O(n^2)$ iterations and has a time complexity of $O(n^{\sim 4})$. In comparison to that of the bisection method, which has a run time of $O(n^2)$, out performing Jacobi's method by a factor of approximately n^2. When studying quantum dots we compare the analytical eigenvalues to those we found numerically for value of $\rho_{max} = \infty$. We checked different values to find what gives the best trade off between computational time and accuracy, and found that for the first eigenvalue that $\rho = 3.8$ gives an answer to at least 4 leading digits.

I. INTRODUCTION

Ordinary differential equation shows up in all parts of physics, with few of them having closed-form solutions, such that approximation schemes are necessary to solve more complicated problems. These problems are eigenvalue problem to begin, or can be scaled to down to one. Where the physical systems are quite different, but the equations needed to solve them are almost identical. Two seemingly unrelated problems, of a buckling beam and an electron in an harmonic oscillator, when reduced to eigenvalue problems, only have a differing factor of the harmonic oscillator potential. Thus having methods to solve equation on the form

$$\mathbf{A}\vec{x} = \lambda\vec{x} \tag{1}$$

numerically, is of high relevance. Such methods include Jacobi method, QR factorization, Lanczos' algorithm, bisection method and more. With each method having its own trade off of precision against run time.

In this study, we will look at two different methods for finding eigenvalues, Jacobi method[5] and bisection method[3]. The importance of unit tests in code development to make sure algorithms are implemented correctly, especially in large numerical projects, is also discussed alongside its implementations in this project. Secondly, the two eigenvalue problems, buckling beam and one electron in three dimensions are described and discretized, turning them into eigenvalue problems for us to solve. As a benchmark of performance the Armadillo library and its function *eigsym*() is used. The mathematical properties and the scaling of Schroedinger's equation is discussed in the appendix. Finally, we discuss our findings and give some concluding remarks.

II. ALGORITHMS

Implementation of the code can be found at: Source Code

A. Unit tests

Unit tests serve a as an easy way of testing code implementation. In large projects, testing all the code at once is not feasible. So rather it is common practice to write unit tests, which ensure that code implementation is done correctly. In addition unit tests help prevent bugs go unnoticed during development this can be critical if multiple people are working on the same project. In this study three unit tests have been implemented. The unit tests covers important mathematical properties of the project. The tests are as follows;

- Setup of Toeplitz matrix - Looking at analytical eigenvalues against computed.
- Finding the largest non-diagonal
- Preservation of eigenvector orthogonality under unitary transformation.

* https://github.com/SigurdSundberg/FYS3150/tree/master/project2

B. Jacobi Rotations

Jacobi's method for finding eigenvalues and eigenvectors is based on a similarity transformations, that is $A' = S^T A S$. Where A is a real and symmetric matrix of dimensions $A \in \mathbb{R}^{n\times n}$, and the transformation matrix S[1], is given by

$$G(i,j,\theta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & \cos\theta & \dots & -\sin\theta & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & \sin\theta & \dots & \cos\theta & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix} \tag{2}$$

We will refer to the matrix by S from here on. It has the following properties $SS^T = \mathbb{I}$ and $S^T = S^{-1}$, which are easy to check. Our goal is to go from A to a diagonal matrix D, which has the eigenvalues of A along its diagonal, in unordered form. This is done by doing multiple similarity transform upon A

$$S_N^T \dots S_1^T A S_1 \dots S_N = D \tag{3}$$

where $N = 1, 2, 3, \dots$. The eigenvectors given by $Ax = \lambda x$, and multiplying by S^T and inserting $S^T S = \mathbb{I}$ we have

$$(S^T A S)(S^T x) = \lambda (S^T x) \tag{4}$$

We recognize $S^T A S$ as A', so it follows that A' has the same eigenvalues as A with the rotated eigenvectors $S^T x$, the eigenvalues preserve their orthogonality as shown in Appendix A.

The following notation will be used from now on, $c = \cos\theta$, $s = \sin\theta$ and $t = \tan\theta = s/c$. The elements of the matrix S are indexed by $s_{kk} = c = s_{ll}$, $s_{kl} = -s$ and $s_{lk} = s$. The similarity transformation $A' = S^T A S$, we denote the elements of A as a_{ij} and the elements of A' as b_{ij} for all $i, j = 1, 2, 3, \dots, N$. We then have [7]

$$\begin{aligned} b_{ii} &= a_{ii},\ i \neq k, i \neq l \\ b_{ik} &= a_{ik}c - a_{il}s,\ i \neq k, i \neq l \\ b_{il} &= a_{il}c + a_{ik}s,\ i \neq k, i \neq l \\ b_{kk} &= a_{kk}c^2 - 2a_{kl}cs + a_{ll}s^2 \\ b_{ll} &= a_{ll}c^2 + 2a_{kl}cs + a_{kk}s^2 \\ b_{kl} &= (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2) \end{aligned} \tag{5}$$

The eigenvectors are found by performing $S^T R$ where $R = \mathbb{I}_N$, its elements are given by r_{ij}. The transformed elements are then

$$\begin{aligned} r_{ik} &= cr_{ik} - sr_{il} \\ r_{il} &= cr_{il} + sr_{ik}. \end{aligned} \tag{6}$$

In the end the eigenvectors will be row wise in matrix R.

After N similarity transforms we want a diagonal matrix D, in other words, we want to set the non-diagonal elements to zero. We require

$$b_{kl} = b_{lk} = (a_{kk} - a_{ll})cs + a_{kl}(c^2 - s^2) = 0 \tag{7}$$

Reshuffle 7;

$$\left(\frac{a_{ll} - a_{kk}}{a_{kl}}\right) cs + c^2 - s^2 = 0 \tag{8}$$

We define

$$\tau = \frac{a_{ll} - a_{kk}}{2a_{kl}} \tag{9}$$

we obtain the quadratic equation

$$t^2 + 2\tau t - 1 = 0 \tag{10}$$

using $t = s/c$.

Solving for t, we find

$$t = -\tau \pm \sqrt{1 + \tau^2} \tag{11}$$

and we can easily obtain c and s via

$$c = \frac{1}{\sqrt{1 + t^2}} \tag{12}$$

and $s = tc$.

The solutions to the quadratic equation is well known, however when τ becomes large $\sqrt{1+\tau^2}$ tends to τ. So the entire expression tends to zero. This can be avoided by solve the quadratic equation given by

$$1 + \tau\frac{1}{t} - \frac{1}{t^2} = 0 \tag{13}$$

we find

$$t = \frac{1}{-\tau \mp \sqrt{1 + \tau^2}} \tag{14}$$

to help with loss of numerical precision. Also we want to choose τ such that our matrix A tends to a diagonal matrix quickly. This is done by choosing the largest value for expression 14.

In addition to have Jacobi method to converge the fastest, we choose the largest non-diagonal term[5]. We define M_{ij} as the largest non-diagonal element, let δ be the required tolerance. Also let R be given by $n \times n$ identity matrix. The Jacobi algorithm then reads as follows

Algorithm 1: Jacobi Algorithm

```
initialization;
while |M_ij^2| ≥ δ do
    update M_ij;
    find τ, c, s, t;
    compute the similiarity transform on A and R
      for i = k and j = l;
end
```

[1] Often referred to as a Givens rotation[12], therefore given by G instead of S.

Looking at the convergence rate of the Jacobi method, it is typically very poor. On average it needs $3n^2 - 5n^2$ rotations, with each rotations requiring $4n$ floating point operations(FLOPs). Resulting in a total of $12n^3 - 20n^3$ FLOPs in order to zero out non-diagonal elements[7]. See a general estimate on the number of iterations(similarity transformations) are needed goes like $O(n^2)$.

C. Bisection

Bisection method is way of root finding, in this case it is used to find the roots of the characteristic polynomial(CP). This implementation is done based on the paper *Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection*[3].

1. Sturm Sequence

We are interested in solution to the general polynomial characteristic polynomial found by $det(A - \lambda\mathbb{I})$. It can be shown [4] that the characteristic polynomial can be expressed by

$$\begin{aligned} P_0(\lambda) &= 1 \\ P_1(\lambda) &= a_1 - \lambda \\ P_i(\lambda) &= (a_i - \lambda)P_{i-1}(\lambda) - b_{i-1}^2 P_{i-2}(\lambda),\ i = 2, 3, \ldots, N \end{aligned} \tag{15}$$

where a_i are the diagonal elements and b_i are the non-diagonal elements defined by $(A - \lambda\mathbb{I})$. This means that $a_i =$ (diagonal element $- \lambda$). We define $d_i = P_i(\lambda)$ in the same way as characteristic polynomial is defined.

We are only interested in the roots of the polynomial, or in other words, we want to look at the number times sign changes[2]. We define $q_i = d_i/d_{i-1}$ to be the sequence of the number of sign changes in CP. We can rewrite 15 as

$$\begin{aligned} q_1 &= a_1 \\ q_i &= a_i - \frac{b_{i-1}^2}{q_{i-1}} \quad i = 2, 3, \ldots, N \end{aligned} \tag{16}$$

The number of negative value in r_i is equal to the number of roots up to a the given value λ. Note that λ is not the eigenvalues.

$P_n(\lambda) = d_n$ can be evaluated in $O(n)$ FLOPs. We define $M(x)$ as the function which returns the number of root up to a value x.

2. Bisection Algorithm

The general bisection method bases it self on root finding. Given an interval $x \in [x_0, x_n]$, for a given polynomial $f(x)$ where $f(x_0) * f(x_n) < 0$, there exist a root in the range of x. In it simplicity, choose a point $x_{mid} \in (x_0, x_n)$ and check $f_m = f(x_{mid})$. If $a = f_m * f(x_0) < 0$ then there is a root in the interval $x \in [x_0, x_{mid})$. Similar if $a > 0$ then we have $x \in (x_{mid}, x_n]$.

The problem has no info about absolute upper or lower bounds for the eigenvalues and where the roots appear. The absolute bounds can be described by Gerschorin's theorem[2].

So to find absolute upper and lower bounds for eigenvalues, y, z, respectively, we have that

$$\begin{aligned} y &= \min_{1 \le i \le n} a_i - |b_i| - |b_{i-1}| \\ z &= \max_{1 \le i \le n} a_i + |b_i| + |b_{i-1}| \end{aligned} \tag{17}$$

Given a matrix A of dimension $N \times N$ with $k \le n$ eigenvalues. We need to find the roots of the characteristic polynomial of dimension k. In the algorithm, k denotes the current number of roots left to find. n goes as $n = 1, 2, ...N$

Let ϵ be our tolerance between each iterations, such that $|x_0 - x_n| < \epsilon$ then we are satisfied with the answer. We define x_0 to be our current smallest value and x_{max} to be our largest current value, those being $x_0 = y$ and $x_{max} = z$. Have v be the empty vector of length n, where we store the found eigenvalue.

The bisection algorithm then reads

Algorithm 2: Bisection Algorithm

```
initialization;
for k = n : −1 : 1 do
    while x_mid = |x_0 − x_n|/2 > ε do
        if M(x_mid) < k then
            x_0 = x_mid else
            x_max = x_mid
        end
    end
    v_k = |x_0 − x_mid|/2
end
```

The bisection method can be evaluated in $O(n)$[3].

III. IMPLEMENTATION

A. Classical mechanics: Buckling Beam

The first eigenvalue problem we look at is a classical mechanics problem, a buckling beam. Let $u(x)$ describe the vertical displacement of the beam in the y direction. Then we have the following differential equation(DE)

$$\gamma \frac{d^2u(x)}{dx^2} = -Fu(x), \tag{18}$$

where γ is a constant defined by properties like the rigidity of the beam. The beam has length L, $x \in [0, L]$ and a force F applied at $(L, 0)$ on the direction towards the origin. We solve this equation with Dirichlet boundary conditions, $u(0) = u(L) = 0$ in other words, the end

points of the beam are fixed. By defining a dimensionless variable $\rho = x/L$, $\rho \in [0,1]$, our equation becomes

$$\frac{d^2u(\rho)}{d\rho^2} = -\frac{FL^2}{\gamma}u(\rho) = -\lambda u(\rho) \tag{19}$$

where $\lambda = FL^2/\gamma$. Our equation 19 when discretized becomes an eigenvalue problem which can be solved numerically. Expanding the above equation using Taylor series to fourth order, gives us the following expression

$$u''(\rho) = \frac{u(\rho+h) - 2u(\rho) + u(\rho-h)}{h^2} + O(h^2) \tag{20}$$

where h is the step. By defining minimum and maximum values for ρ, $\rho_{min} = 0$ and $\rho_{max} = 1$, respectively. We have for a given number of grid points N, that step length h is defined by

$$h = \frac{\rho_{max} - \rho_{min}}{N} \tag{21}$$

The value of ρ at point i is then

$$\rho_i = \rho_0 + ih \quad i = 1, 2, \ldots, N. \tag{22}$$

We can rewrite 19 for ρ_i as

$$-\frac{u(\rho_i+h) - 2u(\rho_i) + u(\rho_i-h))}{h^2} = \lambda u(\rho_i) \tag{23}$$

or in a more compact way as

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \lambda u_i \tag{24}$$

Our differential equation 19 can now be rewritten as a matrix problem on the form $A\mathbf{v} = \lambda\mathbf{v}$. Where A is a tridiagonal Toeplitz matrix.

The general form goes as follows

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \ldots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \ldots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & e_{N-3} & d_{N-2} & e_{N-2} \\ 0 & \ldots & \ldots & \ldots & 0 & e_{N-2} & d_{N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \ldots \\ \ldots \\ \ldots \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \ldots \\ \ldots \\ \ldots \\ u_{N-1} \end{bmatrix} \tag{25}$$

where the endpoints u_0 and u_N are excluded. In our case, $d = 2/h^2$ and $a = -1/h^2$. The eigenvalue problem has analytical eigenpairs[9], with eigenvalues given as

$$\lambda_j = d + 2a\cos\left(\frac{j\pi}{N}\right), \quad j = 1, 2, \ldots, N-1. \tag{26}$$

The eigenvectors are

$$\mathbf{u}_j = \left[\sin\left(\frac{j\pi}{N}\right), \sin\left(\frac{2j\pi}{N}\right), \ldots, \sin\left(\frac{(N-1)j\pi}{N}\right)\right]^T \tag{27}$$

for $j = 1, 2, \ldots, N-1$.

This matrix will function as a baseline for run time and iterations, when comparing Jacobi rotations II B and Bisection II C later.

B. Quantum mechanics: One electron in three dimensions

The second eigenvalue problem comes from Schroedinger's equation for one electron. The radial part of Schroediner's equation for one electron is

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\Psi(r) + V(r)\Psi(r) = E\Psi(r)\right) \tag{28}$$

We assume $l = 0$ and scale the equation as seen in Appendix B we have

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2 u(\rho) = \lambda u(\rho) \tag{29}$$

This equation has analytical solutions for its eigenvalues with $l = 0$ [1]. The four first eigenvalues λ, as follows $\lambda_0 = 3, \lambda_1 = 7, \lambda_3 = 11, \lambda_4 = 15, \ldots$.

We define minimum and maximum values for ρ, $\rho_{min} = 0$ and $\rho_{max} = \infty$, such that the ρ has range $\rho \in [0, \infty)$. As setting any value to infinity on a computer is impossible, ρ_{max} is chosen to different values between $[0, \infty)$, such that we reach a desired precision for our eigenvalues. With a given number of grid points N, we define the step length h,

$$h = \frac{\rho_{max} - \rho_{min}}{N} \tag{30}$$

with ρ at point i given by

$$\rho_i = \rho_{min} + ih, \quad i = 1, 2, 3, \ldots, N \tag{31}$$

Using the same approach as for 23 we have

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \rho_i^2 u_i = \lambda u_i \tag{32}$$

Where $\rho_i^2 = V_i$ is the harmonic oscillator potential. We define diagonal matrix elements $d_i = 2/h^2 + V_i$ and non-diagonal matrix elements $e_i = -1/h^2$. Schroedinger's equation now becomes

$$d_i u_i + e_i u_{i+1} + e_i u_{i-1} = \lambda u_i \tag{33}$$

The problem of one electron moving in three dimensions is now simplified down to a matrix equation on the same form as for the Buckling Beam, with different diagonal elements and boundary conditions. The eigenvalue problem we are now looking at is on the same form as for the Buckling Beam, namely 25. The results in the next section for Schroedinger's equation is found by using Jacobi rotations II B to find the eigenvalues.

IV. RESULTS

A. Iterations and run time

To compare run time between Jacobi method, Bisection method and our baseline being Armadillos $eigsym()$

[10, 11]. We have compared the number of iterations to reach value which satisfies a predetermined tolerance ϵ. For the run time comparison twenty(20) runs for different values of N where completed. This is to reduce the effects of cold starts, and get a better base line for the data.

Looking at the iteration comparison shown in figure 1, we can deduce that for the bisection method that it follows $O(n)$ iterations. For Jacobi method, it follows $O(n^2)$ iterations before it converges. This coincides with the estimated results for the number of iterations we can expect with the different methods.

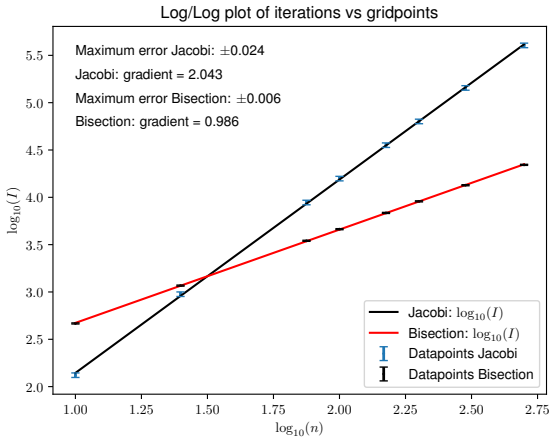

FIG. 1: log / log plot of the number of iterations used by algorithm 1 and 2

plotted against the number of grid points n. Annotated is the gradient of the linear regression curve and max error between the linear regression and data points.

We also compared the run time between the three eigensolvers, algo 1, algo 2 and $eigsym()$ from Armadillo. The results from the run time tests are shown in figure 2. From the figure we have that for the Jacobi method, has a run time which goes $O(n^{\sim 3.76})$ with n. With a fairly stable variation between runs, with a maximum deviation of ± 0.12.

For the bisection method, it has a run time which goes $O(n^{\sim 1.98})$ or just $O(n^2)$. It in turn has a very low deviation between runs, and its run time extremely stable at a deviation of ± 0.02.

For the last method, $eigsym()$ which is used as a baseline for testing the algorithms, it has a run time which goes $O(n^{1.75})$. Which is to be expected from a highly optimized linear algebra library, namely Armadillo, and its accompanying libraries LAPACK and BLAS. It has however the largest deviation in run times at ± 0.38.

B. Buckling Beam: Eigenvectors

The problem with the buckling beam has known eigenvalues and eigenvectors, thus it is easy to check that an

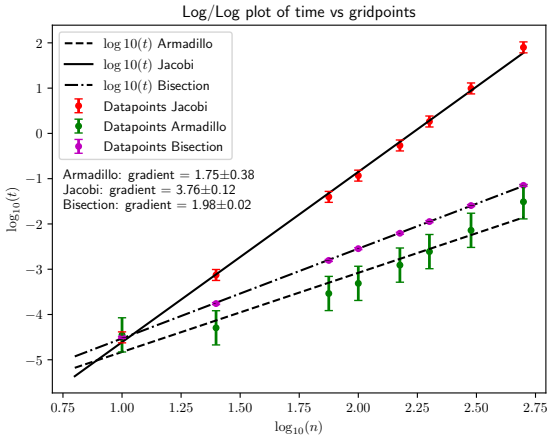

FIG. 2: log / log plot of the time used by algorithms 1, 2 and Armadillos $eigsym()$. The gradient of each linear regression curve is shown along side the error bars of the maximum error between the data points and best fit curve. The max error is also shown after the gradient.

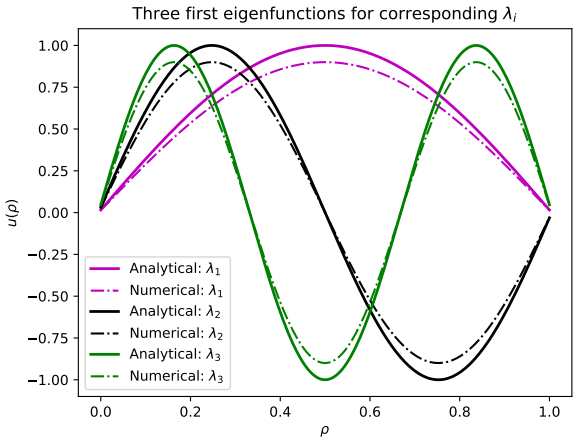

FIG. 3: The eigenvectors given by $u(\rho)$ for the three first eigenvalues λ_i against the analytical counterparts. The numerically found eigenvectors are multiplied by a factor of the relation between the numerical and analytical values times a constant 0.9 to separate the two graphs. The chosen value for n is 200. It is clear that the two graphs are the same.

eigenvalue solver is implemented correctly by comparing the analytical solutions to those found numerically. The analytical and numerically found eigenvectors are shown in figure 3. It is to note that eigenvectors found numerically are scaled to match the analytical eigenvectors by $relation = \frac{v_{analytical}}{v_{numerical}} \times 0.9$. The behavior of eigenvectors is kept under multiplication with a number $c \neq 0$.

TABLE I: The table below shows for what value of ρ and n in which we reached a tolerance between the analytical and numerical eigenvalues with an $\epsilon < 10^{-5}$ used for the calculations. To note n was ran for the following value $n = \{100, 200, 300, 400\}$ and for values of $\rho = 2 : 0.2 : 10$.

ρ	n	Numeric	Analytic	$\left\|\frac{(x_{num}-x_{ana})}{x_{ana}}\right\|$
3.8	200	2.999991	3	3×10^{-6}
4.4	400	7.000046	7	6.5×10^{-6}
4.8	400	11.00021	11	1.9×10^{-5}
5.2	400	15.00025	15	1.6×10^{-5}

C. Quantum dots

We approximated the eigenvalues of one electron in harmonic oscillator with different values for ρ and n. The values of ρ used where in the range $[2, 10]$ with a step length of 0.2. For n four different values where used $[100, 200, 300, 400]$. The fairly low resolution is due to lack of computational power and limitations with mainly the Jacobi method, used to find the eigenvalues and eigenvectors. One could choose a higher resolution in either ρ or n, but at the lack of time. However when looking at eigenvalues to four significant digits, high resolution of a high value for the estimate of $\rho_{max} = \infty$ is not required. The results are displayed in table I. The first four eigenvalues can be approximated by using $\rho_{max} < 6$ and with $n < 500$.

We can see that all four eigenvalues are estimated to at least $1e - 5$ relative precision. Which in our case is within our accepted value for the eigenvalues. Also to note, for the first eigenvalue $n = 200$ was sufficient to reach the wanted precision.

V. DISCUSSION

A. Run time

When it comes to run time, the clear looser is the Jacobi method, which is not surprising as it has a slow convergence rate, and is prone to changing elements set to zero, to non-zero values. Such as to repeat already done processes. With the increase in computing power, this algorithm has not aged well, with its main limiting factor being its high run time. It is an easy to implement algorithm, which is both stable and is suited to implementation on parallel processors[7]. Choosing an $n > 1000$ is in most cases not feasible on normal pc, due to the $O(\sim n^4)$ increase in computation time.

The bisection method performs well, and is also an easy implementation. Its run time of roughly $O(n^2)$ comes from having to loop over k eigenvalues, and root search each polynomial roughly k times. It is also numerically stable in its implementation[3], dealing with division by zero and negative tolerances. This algorithm can be easily adapted to be used with high and low accuracy by simply changing the wanted tolerance, without risking eigenvalues being inaccurate. It can also be parallelized on a top level. However this algorithm only gives the eigenvalues, so finding the eigenvectors would need to be done in a different implementation.

For the baseline we have Armadillos eigenvalue solver in $eigsym()$, which performs the fastest as expected by a optimized library. This however had great variation in run time which goes like $O(n^{1.75})$, however from the figure 2 it is to be noted that for $n = 10(\log_{10}(n) = 1)$ that we most likely have an outlier due to low amount of integration points. And that the actual run time of $eigsym()$ is closer to $O(n^2)$. The method chosen by $eigsym()$ to find the eigenvalues are unknown, but can be assumed to be the most efficient methods Armadillo finds after a quick analysis of the given matrix. For a low number of integration points, this could lead to extended run times compared to for larger values of n, as a smaller percentage of the run time is used for non-computational time.

B. Jacobi method: Eigenvectors

We have shown in figure 3 that our implementation of the Jacobi method, also finds the eigenvalues of our system. The eigenvalues may not be on the form of a traditional eigenvector solver where answer are normally returned in a standardized fashion.

C. Quantum dots: Eigenvalues

We have a trade off between precision, speed and relevance of the data we found. In most cases only the first couple of eigenvalues have any realistic meaning in our world, see buckling beam and figure 3 where we would get a system of sine functions with a lower and lower period. Where the higher eigenvectors and values have no real representation.

The trade off we make here, is mainly between our estimate to $\rho_{max} = \infty$ and the number of grid points we look at. These two are directly related through $\frac{\rho_{max}-\rho_{min}}{n}$ thus leading to a higher estimate for ρ_{max}, needing a larger number of grid points n. If a large value for ρ_{max} is chosen without an appropriate value for n, we have loss of numerical precision, due to lack of integration points. If n is chosen to be too large for the chosen value of ρ_max, we can suffer from loss of numerical precision from numbers close to either zero or ∞. Including, if the Jacobi method is used we reach a limit around $n = 10000$, where it is no longer possible in a reasonable amount of time to perform the computations without a super computer. Using methods such as the bisection method, can overcome this problem, especially due to the nature of our tridiagonal Toeplitz matrix[6].

VI. CONCLUSION

In this project we studied ordinary differential equations and their solutions as eigenvalue problems through the use of the numerical methods. We saw the similarity between a problem from classical mechanics: Buckling beam, and quantum mechanics: Quantum dots with one electron, and how similar to seemingly unrelated problems can become when transformed to an eigenvalue problem. Taking an infinite dimensional problem and through discretization had a finite dimensional problem, that could be solved with the use of linear algebra.

We implemented two different methods for finding eigenvalues, Jacobi method and bisection method, where only the Jacobi method also gives us the eigenvectors of the problem. The Jacobi method, being an iterative method, we ran it up a a predetermined tolerance for the off-diagonal elements of the matrix close to zero. With the nature of the method, it is prone to set already zeroed out elements to non-zero values, thus increasing the number of iterations needed before the matrix is diagonalized. It was expected that the method would converge in $O(n^2)$ iterations, which coincides with the numerical results. However the method performs very poorly when it comes to time complexity, which goes $O(\sim n^4)$. This makes the algorithm a very poor choice for large n. Already for $n = 500$, we see a time usage of ~ 2 minutes. If a want for precision, by choosing a large n, other more efficient algorithms, especially seen as we have a tridiagonal Toeplitz matrix to begin with is favourable. One such algorithm is the bisection method which we also implemented.

Bisection method has a time complexity of roughly $O(n^2)$, as the characteristic polynomial can be evaluated in $O(n)$ and our need to search for n roots, thus $O(n^2)$. This outperforms the Jacobi method with a factor of $O(n^2)$, which means we can choose large values for n to achieve higher precision. There are upper limits to what values of n that can be chosen, as if n gets too large, one will suffer from loss of precision.

Looking at our baseline for run time we can conclude that even though the bisection method is more efficient than Jacobi method, there are still improvements that can be made. This is shown by the use of Armadillos $eigsym()$, even though we can approximate its run time to $O(n^2)$, it is safe to assume that Armadillo has a run time for finding eigenvalues for tridiagonal Toeplitz matrices $< O(n^2)$, if not by much, at least slightly.

Also found is that we can find precise with low amount of integration points, for the three first eigenvectors by use of the Jacobi method, with a relatively low amount of integration points, with $n = 200$, which even though the Jacobi method has a horrible time complexity, it can be used to find eigenvectors with fairly few integration points, for the first couple of eigenvalues. To note is that no tests where done for eigenvectors of higher or lower values of n, to compare them to the analytical solutions. However for $n << 200$, we would get a worse approximation of the analytical eigenvectors in the same interval.

Lastly we looked at the first four eigenvalues for our quantum case. Where we took a continuous problem and approximated it with a finite one. Here we found that ρ can be chosen to a value of at max 5.6 with $400 \leq n < 500$ and still have good approximations for the first four eigenvalues. As discussed earlier the correlation between ρ and n, prevents us from choosing large values of ρ, for small values for n. This means that if we are interested in later eigenvalues we would need to choose an appropriate n, which leads to our Jacobi method being practically unusable.

The algorithms can be optimized by parallelized computing, even still the Jacobi method has a slow convergence rate. There could be minor improvements to memory in the bisection algorithm, but nothing which would out weight any kind of loss to computation time. For future use, the bisection method is a great implementation if the matrix A in $Ax = \lambda x$ is on the form of a tridiagonal Toeplitz matrix. As there is no need to reduce the matrix to a tridiagonal form, which most other algorithms do, such as Householders[8].

Further one could think of extending the quantum dots with one electron to include multiple electrons.

Appendix A: Preservation of orthogonality and dot product

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

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \dots \\ \dots \\ v_{in} \end{bmatrix} \tag{A1}$$

We assume that the basis is orthogonal, that is

$$\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij} \tag{A2}$$

where δ_{ij} is the Kronecker-delta.

Let U be a unitary matrix, where $U^T U = \mathbb{I}_N$, where N notes the dimension of the identity matrix $\mathbb{I}$. Let U operate on $\mathbf{v}_i$ such that

$$\mathbf{w}_i = U\mathbf{v}_i \tag{A3}$$

Then

$$\mathbf{w}_i^T \mathbf{w}_i = (U\mathbf{v}_i)^T U\mathbf{v}_i = \mathbf{v}_i^T U^T U \mathbf{v}_i = \mathbf{v}_i^T \mathbf{v}_i = \delta_{ij} \tag{A4}$$

This shows that a unitary transformation of $\mathbf{v}_i$ both preserves the orthogonality and the dot product. ■

Appendix B: Scaling of Schoedinger's equation

In most cases we are not interested in the actual numerical values of physical problems, but rather the behavior

of said problems. This makes scaling the equations a central part of natural sciences.

Given Schroedinger's euqation for one electron

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr}-\frac{l(l+1)}{r^2}\Psi(r)+V(r)\Psi(r)=E\Psi(r)\right) \tag{B1}$$

where we assume spherical symmetry. In our case $V(r)$ is the harmonic oscillator potential $(1/2)kr^2$ with $k = m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is ω and the energies are

$$E_{nl}=\hbar\omega\left(2n+l+\frac{3}{2}\right) \tag{B2}$$

with $n = 0, 1, 2, \ldots$ and $l = 0, 1, 2, \ldots$. By defining $u(r) = r\Psi(r)$, and substituting into B1 we obtain

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r)+\left(V(r)+\frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}u(r)\right)=Eu(r) \tag{B3}$$

Since we made a transformation to spherical coordinates we have $r \in [0, \infty)$. We assume $l = 0$.

Introduce the dimensionless variable $\rho = (1/alpha)r$, which gives $\Psi(\rho) = (1/2)k\alpha^2\rho^2$. Our equation now reads

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho)+\frac{k}{2}\alpha^2\rho^2u(\rho)=Eu(\rho) \tag{B4}$$

Or equivalently

$$-\frac{d^2}{d\rho^2}u(\rho)+\frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho)=\frac{2m\alpha^2}{\hbar^2}Eu(\rho) \tag{B5}$$

By defining $\lambda = \frac{2m\alpha^2}{\hbar^2}E$ and fixing $\alpha = 1$ such that

$$\alpha=\left(\frac{\hbar^2}{mk}\right)^{\frac{1}{4}}=1 \tag{B6}$$

we can rewrite Schroedinger's equation as

$$-\frac{d^2}{d\rho^2}u(\rho)+\rho^2u(\rho)=\lambda u(\rho) \tag{B7}$$

[1] Project 2, fys3150. *Department of Physics, University of Oslo*, 2020.

[2] James T. Albrecht, Cy P. Chan, and Alan Edelman. Sturm sequences and random eigenvalue distributions. *Found. Comput. Math.*, 9(4):461–483, 2009.

[3] W Barth, RS Martin, and JH Wilkinson. Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. *Numerische Mathematik*, 9(5):386–393, 1967.

[4] Biswa Nath Datta. *Numerical linear algebra and applications*, volume 116. Siam, 2010.

[5] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, fourth edition, 2013.

[6] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, fourth edition, 2013.

[7] Morten Hjorth-Jensen. Computational physics, lecture notes fall 2015. *Department of Physics, University of Oslo*, pages 215–220, 2015.

[8] Morten Hjorth-Jensen. Computational physics, lecture notes fall 2015. *Department of Physics, University of Oslo*, page 173, 2015.

[9] Tom Lyche. Lecture notes for mat-inf 4130, 2017. *Department of Mathematics, Mechanics, Statistics, University of Oslo*, 2017.

[10] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

[11] Conrad Sanderson and Ryan R. Curtin. A user-friendly hybrid sparse matrix class in C++. *CoRR*, abs/1805.03380, 2018.

[12] G.W. Stewart. *Matrix Algorithms Volume 2: Eigensystems*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2001.