Methods of solving eigenvalue problems

Anna Lina P. Sjur and Jan-Adrian H. Kallmyr

October 1, 2018

Abstract

1 Introduction

A great variety of problems in the physical sciences can be represented as eigenvalue problems, which generally takes the form:

$$O\mathbf{v} = \lambda\mathbf{v}, \tag{1}$$

where O is an operator, λ is an eigenvalue, and $\mathbf{v}$ is an eigenvector. Such problems can easily be solved in terms of linear algebra, and is therefore of great use in simplifying complicated problems, as well as providing a framework for creating efficient algorithms. In particular, we will look at the motion of a fixed buckling beam, and show that this is numerically identical to the motion of quantum dots. Specifically, we will solve the following eigenvalue problem

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} + f(x)\right)u(x) = \lambda u(x). \tag{2}$$

Starting with the methods section, we will present the buckling beam problem, as well as the quantum dot problem, scaling and generalising them to the form of eq. 1. Using different numerical methods, we will use unit testing to make sure that each algorithm is implemented correctly. Moving on to the results section, we will present the efficiency and error of each algorithm in terms of CPU-time and relative error respectively. Then finally, in the discussion section, we will compare the different methods, and look at more possiblities for problems that can be solved using the same general algorithms.

2 Methods

We will FLAGG

2.1 The buckling beam

Considering first the buckling beam problem, we have

$$\gamma\frac{\mathrm{d}^2u}{\mathrm{d}x^2} = -Fu(x), \quad x \in [0, L] \tag{3}$$

where γ is a property constant, $u(x)$ the vertical displacement, and F the force applied at (L, 0) towards the origin. We can scale this equation by defining a parameter $\rho = \frac{x}{L}$. Inserting, we get

$$-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}u(\rho) = \lambda u(\rho), \quad \rho \in [0, 1] \tag{4}$$

where $\lambda = \frac{FL^2}{\gamma}$. Now we see that this equation is on the form of eq. 2. However, enforcing Dirichlet boundary conditions $u(0) = u(1) = 0$ and using a 2nd order central approximation for n integration steps we obtain

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} + \mathcal{O}\big(h^2\big) = \lambda_i v_i, \tag{5}$$

where $h = \frac{\rho_n - rho_0}{n}$. Disregarding the boundaries (which are set to 0) we obtain the eigenvalue equation

$$A\mathbf{v} = \lambda\mathbf{v}. \tag{6}$$

Here

$$A = \begin{bmatrix} d & a & 0 & \dots & \dots & 0 \\ a & d & a & 0 & \dots & 0 \\ 0 & a & d & a & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \ddots & a & d & a \\ 0 & \dots & \dots & 0 & a & d \end{bmatrix}, \tag{7}$$

is an tridiagonal matrix where $d = \frac{2}{h^2}$ and $a = -\frac{1}{h^2}$, λ is an eigenvalue, and $\mathbf{v} \in (0, n)$ is an eigenvector. The analytical eigenvalues are given by

$$\lambda_j = d + 2a\cos\left(\frac{j\pi}{n+1}\right), \tag{8}$$

$j = 1, 2, \dots n-1$.

2.2 Single electron in an harmonic oscillator potential

We want to model an electron in a three dimensional harmonic oscillator potential

$$V(r) = \frac{1}{2}m\omega^2 r^2, \quad r = \sqrt{x^2 + y^2 + z^2} \tag{9}$$

$r \in (0, \infty)$ FLAGG, m is the mass, and ω is the frequency. The quantum state can then be represented as the wavefunction

$$|\Psi\rangle \simeq \Psi(r, \phi, \theta) = R(r)Y_l^m(\phi, \theta), \tag{10}$$

where $R(r)$ is the radial part, and $Y_l^m(\theta, \phi)$ are the spherical harmonics. What we need to solve then is the radial equation

$$\left(\frac{-\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + V(r) + \frac{l(l+1)}{r^2}\right)u(r) = Eu(r). \tag{11}$$

(See the appendix for more details on the wavefunction and the radial eq.) Here l is the orbital momentum, $u(r) = rR(r)$, and E are the eigenvalues of $\Psi(r, \theta, \phi)$. We will assume our electron has no orbital momentum ($l = 0$), and scale eq. 11 by substituting $\rho = \frac{r}{\alpha}$ and $\lambda = \frac{E}{\epsilon}$, inserting $V(r)$, and get

$$-\frac{\hbar^2}{2m\alpha^2}\left(\frac{\mathrm{d}^2}{\mathrm{d}\rho^2} - \frac{m^2\omega^2\alpha^4}{\hbar^2}\rho^2\right)u(\rho) = \epsilon\lambda u(\rho), \tag{12}$$

where we can define a natural energy scale $\epsilon = \frac{\hbar^2}{2m\alpha^2}$, and a natural length scale $\alpha = \sqrt{\frac{\hbar}{m\omega}}$, yielding the dimensionless equation

$$\left(-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2} + \rho^2\right)u(\rho) = \lambda u(\rho). \tag{13}$$

Discretising as in eq. 5, the equation becomes

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} + V_i v_i = \lambda v_i, \tag{14}$$

where $V_i = \rho_i^2 = (\rho_0 + ih)^2$. Enforcing the Dirichlet boundary conditions, we see that 14 can be written on matrix form as

$$A\mathbf{v} = \lambda\mathbf{v}. \tag{15}$$

Here

$$A = \begin{bmatrix} d_1^e & a & 0 & \dots & \dots & 0 \\ a & d_2^e & a & 0 & \dots & 0 \\ 0 & a & d_3^e & a & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \ddots & a & d_{n-1}^e & a \\ 0 & \dots & \dots & 0 & a & d_n^e \end{bmatrix}, \tag{16}$$

where $d_i^e = \frac{2}{h^2} + V_i$ and a is as before. The analytical eigenvalues are here given by:

2.3 Two electrons in an harmonic oscillator potential

We will now consider the problem of two electrons in the aforementioned potential. As the

electrons are interacting, we will have to modify the radial equation by adding the Coloumb interaction term

$$V(r_1, r_2) = \frac{\beta e^2}{|\mathbf{r}_1 - \mathbf{r}_2|}, \tag{17}$$

where $\beta e^2 = 1.44 eV nm$, e is the electron charge, and $\mathbf{r}_1$ and $\mathbf{r}_2$ are the positions of electron 1 and 2 respectively. To get the modified radial equation on the form of eq. 2, we use the relative distance $\mathbf{r} \equiv \mathbf{r}_1 - \mathbf{r}_2$, center of mass $\mathbf{R} \equiv \frac{1}{2}(\mathbf{r}_1 + \mathbf{r}_2)$ reference system. We will further only consider the radial solution of the relative distance, and not the center of mass. Using the same scaling parameters α, ϵ, as before (see appendix for the full procedure), we obtain the equation:

$$\left(-\frac{\mathrm{d}^2}{\mathrm{d}\rho^2} + \omega_r^2\rho^2 + \frac{1}{\rho}\right)\psi(r) = \lambda\psi(r). \tag{18}$$

Here $\rho = \frac{r}{\alpha}$, $\omega_r^2 = \frac{1}{4}\frac{m^2\omega^2}{\hbar^2}\alpha^4$, and $\psi(r)$ is the relative distance part of the radial solution. Discretising as usual, and enforcing the Dirichlet conditions, we get our equation on matrix form

$$A\mathbf{v} = \lambda\mathbf{v},$$

where the diagonal elements are now given by $d_i^{2e} = \frac{2}{h^2} + \omega_r^2\rho^2 + \frac{1}{\rho}$.

2.4 Algorithm: Jacobi's method

The strategy of Jacobi's method is to perform a series of similarity transformations on the matrix A in order to diagonalize the matrix. We say that matrix B is similar to A if

$$B = S^{-1}AS, \tag{19}$$

and that the transformation from A to B is a similarity transformation. If S is a real orthogonal matrix we have that

$$S^{-1} = S^T \quad \text{and} \quad B = S^TAS. \tag{20}$$

Since A is a real symmetric matrix there exists a real orthogonal matrix P such that

$$D = P^TAP \tag{21}$$

is a diagonal matrix. Furthermore, the entries along the diagonal of D is the eigenvalues of A, and the column vectors of P are the corresponding eigenvectors. Since the matrix product of two orthogonal matrices is another orthogonal matrix, we can preform a series of similarity transformations until we (ideally) get a diagonal matrix. That is

$$S_j^TS_{j-1}^T\cdots S_1^TAS_1\cdots S_{j-1}S_j = D \tag{22}$$

where $S_i, \quad i = 1, \ldots, j$ are real orthogonal matrices. In Jacobi's method, the matrix S_i is on the form

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix}, \tag{23}$$

where

$$\begin{aligned} s_{kk} &= s_{ll} = \cos\theta \\ s_{lk} &= -s_{kl} = \sin\theta \\ s_{ii} &= 1, \quad i \neq k, l \end{aligned} \tag{24}$$

and elsewhere zero. The matrix S_i is a rotational matrix, which performs a plane rotation around an angle θ in the n-dimensional Euclidean space. For the full transformation expressions, see the appendix.

The idea now is to set k and l such that $a_{kl} = a_{lk}$ is the largest (not considering the sign) non-diagonal element in A, and choose θ so that $b_{kl} = 0$. See the appendix for the expression for θ. Notice that when rotating A such that a_{kl} is set to zero, other matrix elements previously equal to zero may change. However, the frobenius norm of an orthogonal transformation is always preserved, so no elements will blow up. Using the same procedure

on the new matrix, we repeat until all non-diagonal element are essentially zero. Since we want the eigenvectors as well, we need to compute $S_1 \cdots S_{j-1} S_j$ continuously. This gives us the following algorithm

initialise A, S;
while $max > tol$ **do**
    find max, k, l;
    compute $\tau, \sin\theta, \cos\theta$;
    rotate A;
    update S;

Now, the diagonal elements of A and the column vectors of S are the eigenvalues and the corresponding eigenvectors, respectively.

2.5 Error

When the analytical solution is known, we define the error in the eigenvalue as

$$\epsilon_i = \left(\left| \frac{\lambda_i - \lambda_i^{comp}}{\lambda_i} \right| \right) \tag{25}$$

where λ_i is the analytical value and λ_i^{comp} is the computed value.

2.6 Comparing CPU time

To get an ide how the algorithm perform we compare the CPU time for our implementation of Jacobi's method and Armadillo's `eig_sym` function applied at the buckling beam problem. In both cases we time the computation of both the eigenvalues and the eigenvectors, not including the initialisation of the program. The mean time of ten runs is recorded for each n, and we measure the standard deviation for each sample.

2.7 Programming technicalities

All our programs are written in C++, using python3.6 to produce figures and tables. We use armadillo with LAPACK to define matrices and vectors, as well as comparing LAPACK's eigenvalue solver with the Jacobi algorithm. All our code can be found in a github repository "FYS3150" by janadr[1].

3 Results

The first thing we measured was the convergence rate of Jacobi's algorithm, see Figure 1. We see that after some fluctuations for low n, the curve becomes linear as n increases, with a slope of approximately 2. As this is in logspace, we have that the number of iterations increases as $\sim n^2$.

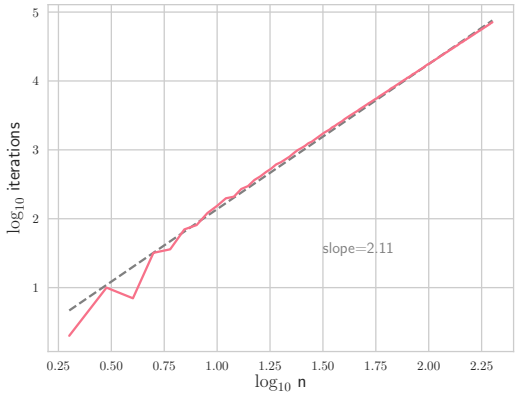

Figure 1: Number of iterations used for solving the buckling beam problem as a function of the matrix dimension n with a logarithmic scale. The slope of the linear fit is 2.11.

In Figure 2 we compare the CPU time between Jacobi's method and armadillo's built-in eigenvalue solver, where we see that the CPU time of Jacobi's method has a similar shape as in Figure 1. Looking at the graph for the armadillo function we see that the slope is rising slower, but have a similar shape, except for the fluctuations occuring after the $\log_{10} n = 1.25$

[1] https://github.com/janadr/FYS3150/tree/master/prosjekt2

mark. We also calculated the standard deviation for these values, but they were on the order of FLAGG, and are thus not visible in the figure.

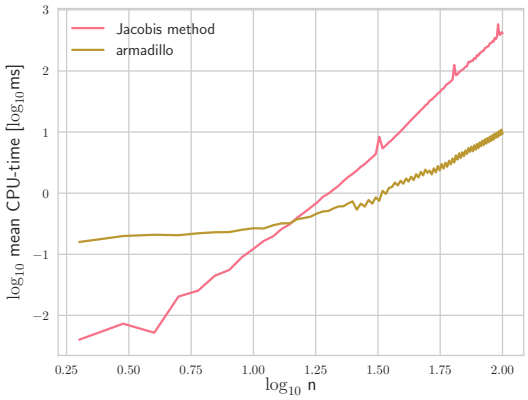

Figure 2: CPU time in ms for solving the buckling beam as a function of the matrix dimension n. Both Jacobi's method (red) and armadillo's `eig_sym` function (yellow) is shown. The scale is logarithmic.

Here (Figure 3), we show the maximum relative error of the Jacobi algorithm. We see that the error increases linearly with a steep slope for low n, before converging towards a constant error for increasing n.

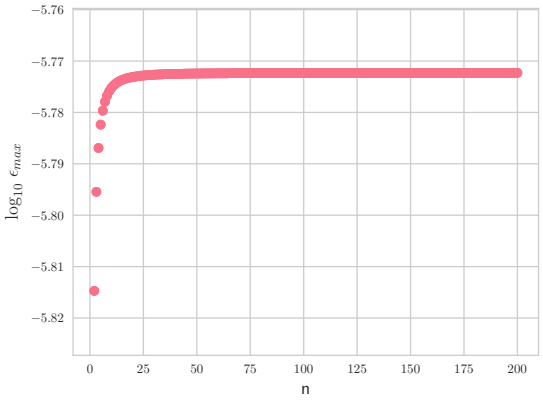

Figure 3: Maximum relative error, ϵ_{max} of Jacobi's algorithm applied on the buckling beam problem. Here n is the matrix dimension, and the scale is logarithmic.

4 Discussion

References

A

$$examplematrix = \begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & 0 \\ a_1 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_2 & b_3 & c_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \ddots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & \dots & \dots & 0 & a_{n-1} & b_n \end{bmatrix},$$

Figure 4: The numeric solution using different solving algorithms. The graphs for n=100 and n=1000 are so similar that they are not distinguishable.

Derivation of the radial equation

Considering the case of a particle in a three dimensional harmonic oscillator potential, we have the Hamiltonian

$$H = -\frac{\hbar^2}{2m}\nabla^2 + V(r), \quad r = \sqrt{x^2+y^2+z^2} \tag{A.1}$$

which in spherical coordinates is given by

$$H = -\frac{\hbar^2}{2m}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + V(r) + \frac{L^2}{2mr^2}, \tag{A.2}$$

where

$$L^2 = -\hbar^2\left[\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right], \tag{A.3}$$

is the squared angular momentum operator. Since L^2 does not act on r, we have that H and L^2 commute, and neglecting spin, a quantum state can then be represented as a wavefunction

$$|\Psi\rangle \simeq \Psi(r,\phi,\theta) = R(r)Y_l^m(\phi,\theta), \tag{A.4}$$

where $R(r)$ is the radial solution, and $Y_l^m(\theta,\phi)$ are the eigenfunctions of L^2 (spherical harmonics). The Schrodinger equation is then

$$HR(r)Y_l^m(\phi,\theta) = ER(r)Y_l^m(\phi,\theta). \tag{A.5}$$

Here E are the eigenvalues of $\Psi(r,\theta,\phi)$. Letting L^2 act on $Y_m^l(\theta,\phi)$ we obtain its eigenvalues $\hbar^2 l(l+1)$. Substituting $R(r) = \frac{u(r)}{r}$, we see that

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right)\frac{u(r)}{r} = \frac{1}{r^2}\frac{\partial}{\partial r}\left[r^2\left(\frac{\partial u}{\partial r}\frac{1}{r} - \frac{u(r)}{r^2}\right)\right] = \frac{1}{r^2}\left(\frac{\partial u}{\partial r} + r\frac{\partial^2 u}{\partial r^2} - \frac{\partial u}{\partial r}\right) = \frac{1}{r}\frac{\partial^2 u}{\partial r^2}.$$

Multiplying by r on both sides, the problem then reduces to the radial equation

$$-\frac{\hbar^2}{2m}\left(\frac{\mathrm{d}^2}{\mathrm{d}r^2} - V(r) - \frac{l(l+1)}{r^2}\right)u(r) = Eu(r). \tag{A.6}$$

Expressions for computing the Jacobi rotation

The resulting matrix from the transformation in eq. 20, written on the same form as eq. 23, is

$$\begin{aligned}
b_{ii} &= a_{ii}, \quad i \neq k, l \\
b_{ik} &= a_{ik}\cos\theta - a_{il}\sin\theta, \quad i \neq k, l \\
b_{il} &= a_{il}\cos\theta + a_{ik}\sin\theta, \quad i \neq k, l \\
b_{kk} &= a_{ll}\cos^2\theta - 2a_{kl}\cos\theta\sin\theta + a_{kk}\sin^2\theta \\
b_{ll} &= a_{kk}\cos^2\theta + 2a_{kl}\cos\theta\sin\theta + a_{ll}\sin^2\theta \\
b_{kl} &= (a_{kk} - a_{ll})\cos\theta\sin\theta + a_{kl}\left(\cos^2\theta - \sin^2\theta\right)
\end{aligned} \tag{A.7}$$

Using eq. A.7, and letting $\tau = \cos 2\theta$, we get

$$\begin{aligned}
\tau &= \frac{a_{ll} - a_{kk}}{2a_{kl}} \\
\tan\theta &= -\tau \pm \sqrt{1 + \tau^2} \\
\cos\theta &= \frac{1}{\sqrt{1 + \tan^2\theta}} \\
\sin\theta &= \tan\theta\cos\theta
\end{aligned} \tag{A.8}$$