# Eigenvalue problems, comparison of Jacobi's and Lanczos' algorithms applied to Schroedinger's equation for an electron in a threedimensional harmonic oscillator well

Johannes Sørby Heines

September 29, 2020

**Abstract**

## Introduction

In physics, we often encounter systems whose behaviour depends on their current state. In many cases these can be described or approximated by equations of the form

$$\frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = \lambda u(x).$$

When such equations are discretized they give rise to an eigenvalue problem

$$\mathbf{A}u = \lambda u.$$

There are many different algorithms for solving eigenvalue problems numerically. We explored two methods for cases in which $A$ is a real symmetrical matrix: Jacobi's and Lanczos' algorithms.

We outline the mathematical workings of each algorithm, describe their numerical implementation in C++ and apply them to the simple case of a single electron in a harmonic oscillator potential.

## Theory

Numerical solutions to eigenvalue problems make use of similarity transformations:

$$\mathbf{B} \ = \mathbf{S}^T\mathbf{A}\mathbf{S}, \quad \text{where} \quad \mathbf{S}^T\mathbf{S} = \mathbb{1}.$$

These are useful because they preserve the eigenvalues [**?**] and the orthogonality of the eigenvectors.

*Proof.* Consider an orthogonal basis of vectors

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \vdots \\ v_{in} \end{bmatrix}, \quad \mathbf{v}_j^T\mathbf{v}_i = \delta_{ij}.$$

Let $\mathbf{w}_i = \mathbf{S}^T\mathbf{v}_i\mathbf{S}$. Then

$$\begin{aligned} \mathbf{w}_j^T\mathbf{w}_i &= (\mathbf{S}^T\mathbf{v}_j\mathbf{S})^T(\mathbf{S}^T\mathbf{v}_i\mathbf{S}) \\ &= \mathbf{S}^T\mathbf{v}_j^T\mathbf{S}\mathbf{S}^T\mathbf{v}_i\mathbf{S} \\ &= \mathbf{S}^T\mathbf{v}_j^T\mathbf{v}_i\mathbf{S} \\ &= \mathbf{S}^T\delta_{ij}\mathbf{S} = \delta_{ij}. \end{aligned}$$

□

### Jacobi's algorithm

Direct methods determine the eigenvalues of a matrix $\mathbf{A}$ by performing a series of similarity transformations

$$\mathbf{S}_N^T \cdots \mathbf{S}_1^T\mathbf{A}\mathbf{S}_1 \cdots \mathbf{S}_N = \mathbf{D},$$

such that $\mathbf{D}$ is a diagonal matrix. Because similarity transformations preserve the eigenvalues, the diagonal elements of $\mathbf{D}$ are the eigenvalues of $\mathbf{A}$. There is no uniquely defined series of similarity transformations which lead to the matrix **B**. Jacobi's algorithm is one way of determining and performing these similarity transformations. Each iteration finds the maximum off-diagonal element of **A**, and performs a rotation along an axis to set that element to zero. This systematically reduces the Frobenius norm of the off-diagonal elements

$$\mathrm{off}(A) = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} a_{ij}^2}.$$

When this value close enough to zero, the algorithm ends giving approximate eigenvalues of **A**.

### Lanczos' algorithm

### Single electron in a harmonic oscillator potential

An electron in a harmonic oscillator potential can occupy energy levels given by the time independent radial Schrödinger equation [?]

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

Here $R(r)$ is the radial wave function, $V(r)$ is the harmonic oscillator potential, the quantum number $l$ is the electron's orbital angular momentum, This is an eigenvalue problem where the eigenvalues $E$ represent the energy levels. [\!/: scaling in methods]

## Methods

### Jacobi's algorithm

The Jacobi algorithm is implemented in three steps. First the program loops through the off-diagonal elements to find the maximum absolute value $|a_{kl}|$. Since the matrix is symmetric it only checks the lower elements. Second, it calculates the rotation angle. [\!/: add calculation to theory] Here, equation **??** can lead to loss of numerical precision when $\tau^2 >> 1$, so it's implemented as

$$\begin{aligned} t &= \frac{(-\tau \pm \sqrt{1+\tau^2})(\pm\tau + \sqrt{1+\tau^2})}{\pm\tau + \sqrt{1+\tau^2}} \\ &= \frac{\pm 1}{\pm\tau + \sqrt{1+\tau^2}}. \end{aligned}$$

Third, the program performs the rotation. It temporarily stores the current values of **A** when needed, and overwrites the matrix with the new values.

### Lanczo's algorithm

### Scaling of the Schrödinger equation

In order to apply the algorithms above to equation 1, we introduce new variables to obtain a dimensionless equation. We only look at the case where $l = 0$.

By making the substitution $R(r) = u(r)/r$, we get

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2}u(r) + V(r)u(r) = Eu(r)$$

We then define $\rho = t/\alpha$ where $\alpha$ is a constant with dimension length, giving $V(\rho) = k\alpha^2\rho^2/2$, and thus

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

Now we can choose $\alpha$ so that $mk\alpha^4/\hbar^2 = 1$ and define $\lambda = 2m\alpha^2 E/\hbar^2$, and rewrite the Schrödinger equation as

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

This equation can then be discretized, giving the eigenvalue equation $\mathbf{A}u = \lambda u$, where **A** is a tridiagonal matrix

$$\mathbf{A} = \begin{bmatrix} d_1 & e_1 & & \\ e_1 & d_2 & \ddots & \\ & \ddots & \ddots & e_{N-2} \\ & & e_{N-2} & d_{N-1} \end{bmatrix},$$

with $e_i = -1/h^2$ and $d_i = 2/h^2 + \rho_i^2$, where $h = (\rho_N - \rho_0)/N$ is the step length [?].

## Results

## Discussion

## Conclusion

## References