Project 2

Linus Ekstrøm, Joakim Flatby

September 2018

Abstract

The goal of this project is to numerically solve two seemingly unrelated eigenvalue problems: a buckling beam and the Schrödinger equation for two electrons in a three-dimensional harmonic oscillator well. The two systems are related by the fact that solving them both require solving a tridiagonal Toeplitz-matrix equation.

Contents

Introduction

In this article we are going to be looking into a method of solving differential equations formulated as an eigenvalue problem. First we have a section with the necessary theoretical background for our algorithm and the physical systems we will be working with. Here we look at discretization of differential equations as a symmetric Toeplitz matrix, as well as the Jacobi rotation algorithm for solving such problems. Next we have section outlining the Jacobi rotation algorithm in detail. Following this we will be presenting our results, and finally a brief discussion of the method as an addition in our simulational toolbox.

Theory

The buckling beam problem

The behaviour of a buckling beam can be modelled with the following differential equation

$$\gamma \frac{d^2u(x)}{dx^2} = -Fu(x) \tag{1}$$

where $u(x)$ is the vertical displacement of the beam in the y direction. We say the beam has a length L with $x \in [0, L]$. F is a force applied at the edge of the beam at $(L, 0)$ with direction towards the origin. γ is a structural constant defined by properties like the rigidity of the beam. To obtain a solvable system of equations we apply the Dirichlet boundary condition: $u(0) = u(L) = 0$. Next, we define a dimensionless variable ρ

$$\rho = \frac{x}{L} \tag{2}$$

which implies that $\rho \in [0, 1]$. Rewriting the buckling beam equation with this in mind we formulate our eigenvalue problem

$$\frac{d^2u(\rho)}{d\rho^2} = -\frac{FL^2}{R}u(\rho) = -\lambda u(\rho) \tag{3}$$

Next, we wish to discretize this equation so that we are able to solve it numerically. We use the following approximation for the second derivative of $u(\rho)$ with respect to ρ

$$\frac{d^2u(\rho)}{d\rho^2} = \frac{u(\rho + h) - 2u(\rho) + u(\rho - h)}{h^2} + \mathcal{O}(h^2) \tag{4}$$

where h is the step size, for a given number of mesh points N, given by

$$h = \frac{\rho_N - \rho_0}{N} \tag{5}$$

The value of ρ at point i is then

$$\rho_i = \rho_0 + ih \qquad i \in [1, N] \tag{6}$$

With all of this in mind the differential equation can be rewritten as

$$-\frac{u(\rho_i + h) - 2u(\rho_i) + u(\rho_i - h)}{h^2} = \lambda u(\rho_i) \tag{7}$$

Alternatively, in a more compact form

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \lambda u_i \tag{8}$$

It is known that a system of equations on this form can be rewritten in a more general form as a tridiagonal Toeplitz matrix equation

$$\begin{bmatrix} d & a & 0 & 0 & \dots & 0 & 0 \\ a & d & a & 0 & \dots & 0 & 0 \\ 0 & a & d & a & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & a & d & a \\ 0 & \dots & \dots & \dots & \dots & a & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}. \tag{9}$$

where we have defined $d = 2/h^2$ and the upper and lower diagonal as $a = -1/h^2$. This eigenvalue problem has analytical eigenpairs making testing of our implementation possible. The eigenvalues are given analytically by

$$\lambda_j = d + 2a\cos\big(\frac{j\pi}{N+1}\big) \qquad j \in [1, N] \tag{10}$$

Orthogonal Transformations

An orthogonal or unitary transformation is a transformation whose matrix representation has determinant equal to 1 or -1. They represent either pure rotations or reflections. Thus if we say $\det(\mathbf{A}) = \pm 1$ we have

$$\mathbf{A}\vec{u} \cdot \mathbf{A}\vec{v} = (\mathbf{A}\vec{u})^T(\mathbf{A}\vec{v}) = \vec{u}^T(\mathbf{A}^T\mathbf{A})\vec{v} = \vec{u}^T\vec{v} = \vec{u} \cdot \vec{v} \tag{11}$$

where we have used the fact that an $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ for an orthogonal matrix.

Jacobi's Rotation Algorithm

Jacobi's rotation algorithm, or just Jacobi's method, is a method to find the eigenvalues for a real, symmetric matrix $\mathbf{A}$. Jacobi's method introduces an $(n \times n)$ orthogonal transformation matrix, which performs a plane rotation around an angle θ in the Euclidean $n-$dimensional space.

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \cos\theta & 0 & 0 & \dots & \sin\theta \\ 0 & 0 & 0 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & 1 & 0 \\ 0 & \dots & -\sin\theta & \dots & \dots & 0 & \cos\theta \end{bmatrix} \tag{12}$$

To simplify we define the quantities $\tan\theta = t = s/c$. The non-zero elements of the matrix $\mathbf{S}$ are then given by

$$s_{kk} = s_{ll} = c, \;\; s_{kl} = -s_{lk} = -s, \;\; s_{ii} = 1 \tag{13}$$

where $i \neq k$, $i \neq l$ and i, j are the indices of the matrix $\mathbf{S}$. Thus, the similarity transformation

$$\mathbf{B} = \mathbf{S^T A S} \tag{14}$$

results in

$$\begin{aligned} b_{ii} &= a_{ii} \;\; i \neq k, i \neq l \\ b_{ik} &= ca_{ik} - sa_{il} \;\; i \neq k, i \neq l \\ b_{il} &= ca_{il} - sa_{ik} \;\; i \neq k, i \neq l \\ b_{kk} &= c^2 a_{ll} - scsa_{kl} + s^2 a_{ll} \\ b_{ll} &= c^2 a_{ll} + 2csa_{kl} + s^2 a_{kk} \\ b_{kl} &= cs(a_{kk} - a_{ll}) + (c^2 - s^2) a_{kl} \end{aligned} \tag{15}$$

We wish to choose θ such that all the non-diagonal matrix elements become zero, $b_{kl} = 0$

$$b_{kl} = cs(a_{kk} - a_{ll}) + (c^2 - s^2) a_{kl} = 0 \tag{16}$$

Solving and introducing a new variable $\tau = \cot 2\theta$ we are able to bring $b_{kl} = 0$ over to a quadratic form

$$\begin{aligned} 2\tau = \frac{c^2 - s^2}{cs} &= \frac{c^2}{cs} - \frac{s^2}{cs} = \frac{1}{t} - t \\ &\implies 2\tau t = 1 - t^2 \\ &\implies t^2 + 2\tau t - 1 = 0 \end{aligned} \tag{17}$$

which has the solutions

$$t = -\tau \pm \sqrt{1 + \tau^2} \tag{18}$$

Choosing t to be the smaller of the roots lets us ensure that $|\theta| \leq \pi/4$ and has the effect of minimizing the difference between the matrices $\mathbf{B}$ and $\mathbf{A}$ since

$$||\mathbf{B} - \mathbf{A}||_F^2 = 4(1 - c) \sum_{i=1, i \neq k,l}^{n} (a_{ik}^2 + a_{il}^2) + \frac{2a_{kl}^2}{c^2} \tag{19}$$

The convergence rate of Jacobi's method is rather low, for a given matrix one typically needs between $3n^2$ and $5n^2$ rotations with each rotation requiring $4n$ operations, resulting in a total of between $12n^3$ and $20n^3$ operations in order to zero out non-diagonal matrix elements.

In our particular case we wish to solve our system of equations for the eigenvalues. However, Jacobi's method alters the eigenvector for each similarity transformation. The eigenvector is changed from $\mathbf{x}$ to $\mathbf{S^T x}$, however if we keep track of the changes made to the eigenvector we can still solve for the original vector.

Discretizing the Schrödinger Equation

In much the same fashion as the buckling beam problem, electron motion in a harmonic oscillator potential can also be discretized as a tridiagonal Toeplitz eigenvalue problem. We start by rewriting the radial Schrödinger equation

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r) \tag{20}$$

Next we make the substitution $R(r) = 1/ru(r)$ and obtain

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r) \tag{21}$$

with accompanying boundary conditions $u(0) = u(\infty) = 0$. Again, we introduce a dimensionless variable $\rho = (1/\alpha)r$ where α is a constant with dimension length. The Schrödinger equation now reads

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \Big(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\Big)u(\rho) = Eu(\rho) \tag{22}$$

In this article we are only interested in solving the Schrödinger equation for the ground state $l = 0$ for the harmonic oscillator potential $V(\rho) = (1/2)k\alpha^2\rho^2$ and thus we can simplify the equation further

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \frac{k}{2}\alpha^2\rho^2u(\rho) = Eu(\rho) \tag{23}$$

Further we multiply through by $2m\alpha^2/$

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho) \tag{24}$$

The constant α can now be fixed as

$$\alpha = \Big(\frac{\hbar^2}{mk}\Big)^{1/4} \tag{25}$$

Next, defining

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E \tag{26}$$

we can now rewrite the Schrödinger equation as

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2u(\rho) = \lambda u(\rho) \tag{27}$$

As with the buckling problem we can now discretize the Schrödinger equation in the same way

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \rho_i^2u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + V_iu_i = \lambda u_i \tag{28}$$

where $V_i = \rho_i^2$ is the harmonic oscillator potential. We first define the main diagonal matrix elements $d_i = (2/h^2) + V_i$ and the upper- and lower-diagonal elements as $e = -(1/h^2)$. Notice that for single electron motion in a harmonic oscillator potential all the non-main diagonal elements are constant and equal. With these newly defined quantities the Schrödinger equation takes the following form

$$d_iu_i + eu_{i-1} + eu_{i+1} = \lambda u_i \tag{29}$$

where u_i is unknown. We can formulate this latest form as a Toeplitz tridiagonal eigenvalue problem

$$\begin{bmatrix} d_0 & e & 0 & 0 & \dots & 0 & 0 \\ e & d_1 & e & 0 & \dots & 0 & 0 \\ 0 & e & d_2 & e & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & e & d_{N-1} & e \\ 0 & \dots & \dots & \dots & \dots & e & d_N \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \dots \\ u_{N-1} \\ u_N \end{bmatrix} = \lambda \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \dots \\ u_{N-1} \\ u_N \end{bmatrix}. \tag{30}$$

Multiple Electrons in Three-Dimensional Harmonic Oscillator Potential

The electrons motion with no repulsion is obtained by solving the Schrödinger equation

$$\Big(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2} - -\frac{\hbar^2}{2m}\frac{d^2}{dr_2^2} + \frac{1}{2}kr_1^2 + \frac{1}{2}kr_2^2\Big)u(r_1,r_2) = E^{(2)}u(r_1,r_2) \tag{31}$$

with a two-electron wavefunction $u(r_1, r_2)$ and the two-electron energy $E^{(2)}$. Because there is no repulsion the Schrödinger equation is separable with a closed form solution. To obtain this solution we first introduce the relative coordinate $\vec{r} = \vec{r_1} - \vec{r_2}$ and the centre-of-mass coordinate $\vec{\mathbf{R}} = 1/2(\vec{r_1} + \vec{r_2})$. With these new coordinates, the radial Schrödinger equation reads

$$\Big(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} - -\frac{\hbar^2}{4m}\frac{d^2}{dR^2} + \frac{1}{4}kr^2 + kR^2\Big)u(r,R) = E^{(2)}u(r,R) \tag{32}$$

Next we assume that we can separate the two particle wavefunction as $u(r, R) = \psi(r)\phi(R)$ with the total energy equalling the sum of the relative energy E_r and the centre-of-mass energy E_R. For simplicity's sake we omit the centre-of-mass energy in our calculations. Thus, the Schrödinger equation reads

$$\Big(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}kr^2\Big)\psi(r) = E_r\psi(r) \tag{33}$$

Next we add the Coulomb interaction between the two electrons

$$V(r_1,r_2) = \frac{\beta e^2}{|\vec{r_1} - \vec{r_2}|} = \frac{\beta e^2}{r} \tag{34}$$

where $\beta e^2 = 1.44$eVnm Like before we introduce the dimensionless variable $\rho = r/\alpha$ and end up with

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2\psi(\rho) + \frac{m\alpha\beta e^2}{\rho\hbar^2}\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho) \tag{35}$$

We define a 'frequency' which reflects the strength of the oscillator potential

$$\omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4 \tag{36}$$

and again constrain the constant α such that

$$\alpha = \frac{\hbar^2}{m\beta e^2} \tag{37}$$

Next, we define λ

$$\lambda = \frac{m\alpha^2}{\hbar^2}E_r \tag{38}$$

Finally we have the Schrödinger equation on a similar form as we had for the one-electron system

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \lambda\psi(\rho) \tag{39}$$

Method

Implementing Jacobi's algorithm to solve the Schrödinger equation

The matrix equation (30) is what we will use to solve for the motion of the single and double electron case. In practice, the only difference between the two systems is the potential V_i and thus the Toeplitz matrix. We solved the matrix equation numerically by implementing it as follows

- First we choose a tolerance ϵ typically $\epsilon < 10^{-8}$
- Next, we compare the norm of the newly computed off-diagonal matrix elements $\max(a_{ij}^2) > \epsilon$
- Then we choose the matrix elements a_{kl} so that we have those with largest value, that is $|a_{kl}| = \max_{i \neq j} |a_{ij}|$
- After this compute $\tau = (a_{ll} - a_{kk})/2a_{kl}, \tan\theta, \cos\theta$ and $\sin\theta$
- With these values in hand compute the similarity transformation for the set of values (k, l). Resulting in the new matrix $\mathbf{B} = \mathbf{S}(k, l, \theta)^T \mathbf{A} \mathbf{S}(k, l, \theta)$
- Iterate this procedure until $\max(a_{ij}^2) \leq \epsilon$

After the program has run its course, we are left with a matrix containing the eigenvalues on the diagonal, and can extract these to get a list of our eigenvalues. Using numpy's built in eigenvalue-function we can find the analytical eigenvalues and compare them to our own result. Furthermore, we can count and save the amount of iterations the program had to go through before $\max(a_{ij}^2) \leq \epsilon$. Comparing the amount of iterations when running the program with different dimensions for the toeplitz-matrix, we can see how the amount of iterations needed changes based on the dimensions of the matrix.

No Electron Repulsion

Once we had tested our Jacobi implementation against the buckling beam problem, all we had to do was add the potential term ρ_i^2 to the main diagonal of the Toeplitz matrix.

Adding the Coulomb Repulsion

Again, all that is necessary to calculate the eigenvalues of this new system is altering the main diagonal elements of the Toeplitz matrix. This time the potential term is $\rho_i^2 \omega_r + 1/\rho_i$

Results

Jacobi's Rotation Algorithm

After running our program using matrix dimensions $(100, 100)$ the difference between the analytical and our numerical eigenvalues was an average of $8.7 \cdot 10^{-10}$, which means we have a pretty good approximation.

We let the program run using several different dimensions for the toeplitz-matrix, and recorded the amount of similarity transformations needed before our maximum was below our chosen tolerance. The results are shown here

n	iters	n	iters
10	118	110	17334
20	547	120	20510
30	1198	130	22969
40	2239	140	27806
50	3389	150	31100
60	5055	160	36895
70	6838	170	40329
80	8961	180	44047
90	11179	190	51059
100	13978	200	55119

Table 1: Regression points for analysis of number of similarity transformations as a function of the matrix's dimensionality

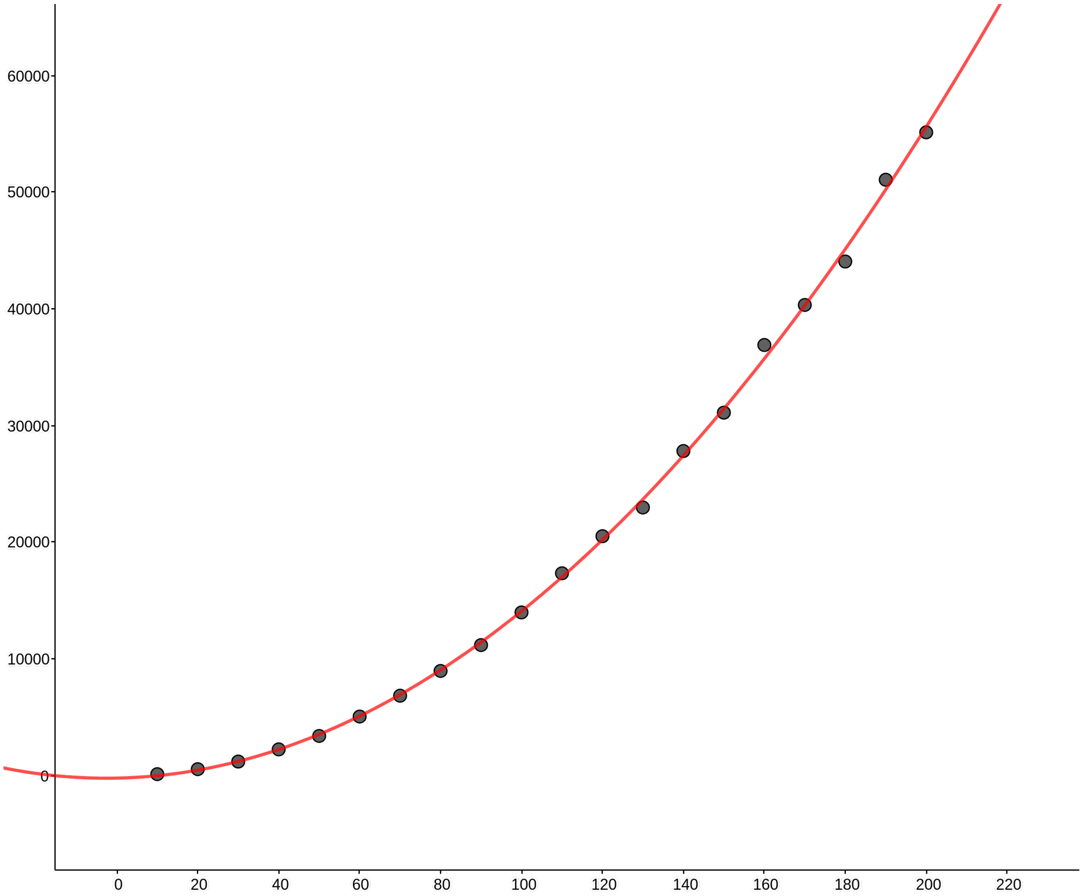

Figure 1: Regression plot for the number of similarity transformations as a function of the matrix's dimensionality. The quadratic polynomial best fitting our regression points is: $f(x) = 1.36x^2 + 6.96x - 222$

Single Electron Eigenvalues

In the table below are the first four eigenvalues from our algorithm compared to eigenvalues obtained using numpy's linalg.eig method. We used values $N = 100$ and $\rho_{max} = 5$ to obtain eigenvalues which agreed to four decimal places with numpy's method.

Numerical	Analytical	Difference
3.0057	3.0063	$5.2262 \cdot 10^{-4}$
7.0058	7.0066	$7.6660 \cdot 10^{-4}$
11.0028	11.0028	$3.5789 \cdot 10^{-5}$
15.0007	15.0011	$3.8917 \cdot 10^{-4}$

Table 2: Numeical

Double Electron Eigenvalues

Below are our calculated values for the first four eigenvalues for four different ω_r values. We see that by increasing ω_r the eigenvalues increase. In other words, by increasing the potential strength we increase the energy of the electrons.

ω_r	0.01	0.5	1	5
λ_1	0.9123	3.0003	4.0579	8.3681
λ_2	2.2191	5.7090	7.9083	17.0929
λ_3	4.2360	8.4838	11.8127	25.8955
λ_4	7.0035	11.4589	15.7482	34.7253

Table 3: We used $N = 100$ integration points with $\rho_{max} = 5$. Again, the error between our Jacobi implementation and numpy's linalg.eig was small.

Conclusion

After creating a script whose primary purpose was to find the eigenvalues of a specific type of matrix, a symmetric Toeplitz matrix with three diagonals, we are left with a platform with which we can solve two very different physical situations. By discretizing the differential equations which describe the behavior of our two systems we were able to create a toeplitz matrix for each of them which we can plug in to our eigenvalue-solver. By comparing with the values for each matrix given by numpy's built-in eigenvalue-function, we can also see that our program works, and gives reasonably accurate values depending on the dimensions of the matrices. We also programmed in the analytical solution ourselves, which confirmed the results from our approximation and from numpy.