Project 3 FYS4150
Ordinary differential equations

Audun Tahina Reitan and Marius Holm

October 19, 2018

1 Abstract

We give a brief introduction to eigenvalue problems and how the Jacobi method can be used to solve such problems. We also give a brief discussion on scaling equations, and numerical stability for a relevant equation.

We then show that the dot product and orthogonality is preserved for an orthogonal transformation. We then present a small selection of results and two plots of relevance. More of our results can be found in the GitHub repository linked in the implementation section.

2 Introduction

In this project we'll develop code for simulating the solar system, using the Verlet algorithm for solving coupled ordinary differential equations. For this project we will focus on the use of classes in our code, which makes it a lot easier to reuse larger parts of our code for problems of similar character. In our case of the solar system we can create a class describing a planet, a moon, or some other astronomical body which we want to include in our solar system.

In order to test that our algorithm works, we start by looking at a simple hypothetical system consisting of only the Earth orbiting the Sun. We then investigate the stability of our algorithm and plot the position of the Earth orbiting the Sun. Furthermore we consider the initial velocity needed for the planet to escape the gravity of the Sun. Then we'll consider a three-body system consisting of the Earth, the Sun, and Jupiter. We then expand our three-body model to the entire solar system including Pluto. In the final part of our project we look at the perihelion precession of Mercury.

3 Methods

3.1 Eigenvalue problems

Given the eigenvalue problem

$$\mathbf{A}\mathbf{x}^{(v)} = \lambda^{(v)}\mathbf{x}^{(v)} \tag{1}$$

where $\mathbf{A}$ is a matrix of dimension n. We also have that $\lambda^{(v)}$ are the eigenvalues and $\mathbf{x}^{(v)}$ the corresponding eigenvectors. From this we find that the eigenvalues of $\mathbf{A}$ are given by the n roots of the characteristic polynomial given by:

$$P(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A}) = \prod_{i=1}^{n} (\lambda_i - \lambda) \tag{2}$$

This procedure is only valid for problems where we only need a small fraction of the eigenvalues and eigenvectors, or if the matrix is on a tridiagonal form. As our matrix is tridiagonal we can use the above procedure which leads us to the Jacobi method. A general procedure for solving equation (1) is to perform similarity transformations until the original matrix $\mathbf{A}$ is either on diagonal form or is a tridiagonal matrix which then easily can be diagonalized. The general procedure leads to the well known Householder's algorithm.[**?**]

3.2 Jacobi's method

Given an $n \times n$ orthogonal transformation matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & \cos\theta & 0 & \dots & 0 & \sin\theta \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & \sin\theta & \dots & \dots & 0 & \cos\theta \end{pmatrix} \tag{3}$$

with the property $\mathbf{S^T} = \mathbf{S^{-1}}$. This matrix performs a plane rotation around an angle θ in the Euclidean n-dimensional space. Which means that the non-zero matrix elements are given by

$$s_{kk} = s_{ll} = \cos\theta, \;\; s_{kl} = -s_{lk} = -\sin\theta, \;\; s_{ii} = -s_{ii} = 1 \quad i \neq k \;\; i \neq l, \tag{4}$$

A similarity transformation

$$\mathbf{B} = \mathbf{S}_{u\mathbf{T}}\mathbf{A}\mathbf{S}, \tag{5}$$

results in

$$\begin{aligned} b_{ii} &= a_{ii}, \; i \neq k, i \neq l \\ b_{ik} &= a_{ik}\cos\theta - a_{il}\sin\theta, \; i \neq k, i \neq l \\ b_{il} &= a_{il}\cos\theta + a_{ik}\sin\theta, \; i \neq k, i \neq l \\ b_{kk} &= a_{kk}\cos^2\theta - 2a_{kl}\cos\theta\,\sin\theta + a_{ll}\sin^2\theta \\ b_{ll} &= a_{all}\cos^2\theta + 2a_{kl}\cos\theta\,\sin\theta + a_{kk}\sin^2\theta \\ b_{kl} &= (a_{kk} - a_{ll})\cos\theta\,\sin\theta + a_{kl}(\cos^2\theta - sin^2\theta) \end{aligned}$$

We can choose the angle θ as we wish, while making sure that all the non-diagonal matrix elements, b_{kl} become zero. The algorithm which follows is then quite simple. We do a number of similarity transformations until the sum over the squared non-diagonal matrix elements are less than some chosen tolerance, typically less than 10^{-8}. We therefore seek to minimize

$$\text{off}(\mathbf{A}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} a_{ij}^2} \tag{6}$$

3.3 Scaling equations

Transformation matrix For Jacobi's method the the rotational elements $\sin\theta = s$ and $\cos\theta = c$ with $\tan\theta = t = s/c$ is given by:

$$\cot 2\theta = \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}} \tag{7}$$

$$\cot 2\theta = \frac{1}{2}\left(\cot\theta - \tan\theta\right) \implies t^2 + 2\tau t - 1 = 0 \implies t = -\tau \pm \sqrt{1+\tau^2} \tag{8}$$

$$c = \frac{1}{\sqrt{1+t^2}} \tag{9}$$

$$s = t\,c \tag{10}$$

For t we choose the solution to the 2nd degree formula that returns the smallest answer so that t does not reach very high values. We do this because of the risk for loss of significance because of cancellation in the equation for t when τ becomes so big that $\sqrt{1+\tau^2} \approx \tau$.

$$t = \frac{1}{\tau + \sqrt{1+\tau^2}}, \quad \tau > 0.0 \tag{11}$$

$$t = \frac{-1}{-\tau + \sqrt{1+\tau^2}}, \quad \tau \leq 0.0 \tag{12}$$

These two solutions are equivalent to the smallest solution of the 2nd degree equation, and are numerical stable.

Quantum mechanics We are also interested in quantum mechanical problems where we want to solve the radial part of Schrödinger's equation for one electron in a harmonic oscillator potential. The equation is given as:

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = E\,R(r) \tag{13}$$

$V(r)$ for a harmonic oscillator potential is given as $V(r) = \frac{1}{2}kr^2$ with $k = m\omega^2$ and where E is the energy of the harmonic oscillator in three dimensions. ω is the oscillator frequency, and the energies are given as:

$$E_{nl} = \hbar\omega\left(2n + l + \frac{3}{2}\right), \text{ with } n = 0, 1, 2, \dots \text{ and } l = 0, 1, 2, \dots \tag{14}$$

Since we have made the transformation into spherical coordinates we have $r \in [0, \infty)$, and the quantum number l is the orbital momentum of the electron. We then substitute $R(r) = \frac{1}{r} u(r)$ and get

$$\frac{\hbar^2}{2m} \frac{d^2}{dr^2} u(r) + \Big(V(r) + \frac{l(l+1)}{r^2} \frac{\hbar^2}{2m} \Big) u(r) = E\, u(r) \tag{15}$$

With boundary conditions $u(0) = 0$ and $u(\infty) = 0$. We introduce the dimensionless variable $\rho = (1/\alpha)r$ where α is a constant with dimension length. We set $l = 0$, and insert $V(\rho) = (1/2)k\alpha^2\rho^2$. This gives us the following equation

$$-\frac{\hbar^2}{2m\alpha^2} \frac{d^2}{d\rho^2}\, u(\rho) + \frac{k}{2} \alpha^2 \rho^2\, u(\rho) = E\, u(\rho) \quad | \cdot \frac{2m\alpha^2}{\hbar^2} \tag{16}$$

$$-\frac{d^2}{d\rho^2}\, u(\rho) + \frac{mk}{\hbar^2} \alpha^4 \rho^2\, u(\rho) = \frac{2m\alpha^2}{\hbar^2} E\, u(\rho) \tag{17}$$

Fixing the constant α as

$$\alpha = \Big(\frac{\hbar^2}{mk} \Big)^{\frac{1}{4}} \tag{18}$$

We also define $\lambda = \frac{2m\alpha^2}{\hbar^2} E$ which lets us write the Schrödinger equation as

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

A lot of the same steps as above are repeated for the problem considering quantum dots in three dimensions with two electrons. I will therefore only state the final result along with the necessary constants that are defined. We then find the Schrödinger equation as:

$$-\frac{d^2}{d\rho^2} \psi(\rho) + \omega_r^2 \rho^2\, u(\rho) + \frac{1}{\rho} = \lambda\, \psi(\rho) \tag{20}$$

where we have defined ω_r^2 as

$$\omega_r^2 = \frac{1}{4} \frac{mk}{\hbar^2} \alpha^4 \tag{21}$$

and fixing α by requiring

$$\alpha = \frac{\hbar^2}{m\beta e^2}. \tag{22}$$

Finally we have defined λ as

$$\lambda = \frac{m\alpha^2}{\hbar^2} E \tag{23}$$

3.4 Implementation

For the Jacobi's rotation method we implemented the algorithm given on page 217 in [?] and using the scaling and rewriting we have done the transformation matrix. In the github repository cited below the program can be found in the `jacobi.cpp` file. In order to find the highest valued off-diagonal element we took the upper triangular part of the matrix given, removed the diagonal, and then used Armadillos own methods to find the maximum absolute value element. For further details regarding implementation, see comments in the different programs.

All our code, calculations, unittests, and plots used can be found in Auduns GitHub repository.

4 Results

4.1 Preservation of dot product

We want to show that an orthogonal transformation preserves the dot product and orthogonality. Given a basis of vectors $\mathbf{v}_i$

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ v_{i2} \\ \vdots \\ v_{in} \end{bmatrix} \tag{24}$$

And assuming that the basis is orthogonal.

$$\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij} \tag{25}$$

The transformation is given by

$$\mathbf{w}_i = \mathbf{U}\mathbf{v}_i \tag{26}$$

As the transformation matrix $\mathbf{U}$ is orthogonal we have $\mathbf{U}^T\mathbf{U} = \mathbf{I}$.

$$\begin{aligned} \mathbf{w}_i \cdot \mathbf{w}_j &= \mathbf{U}\mathbf{v}_i \cdot \mathbf{U}\mathbf{v}_j = (\mathbf{U}\mathbf{v}_i)^T(\mathbf{U}\mathbf{v}_j) = (\mathbf{U}^T\mathbf{v}_i^T)(\mathbf{U}\mathbf{v}_j) \\ &= \mathbf{v}_i^T(\mathbf{U}^T\mathbf{U})\mathbf{v}_j = \mathbf{v}_i^T\,\mathbf{I}\,\mathbf{v}_j = \mathbf{v}_i^T\,\mathbf{v}_j = \mathbf{v}_i \cdot \mathbf{v}_j = \mathbf{v}_i^T\mathbf{v}_j \end{aligned}$$

As $\mathbf{w}_i \cdot \mathbf{w}_j = \mathbf{v}_i \cdot \mathbf{v}_j$ the dot product is preserved. The same applies for $\mathbf{v}_i \cdot \mathbf{v}_j = \mathbf{v}_i^T\mathbf{v}_j = \delta_{ij}$ which implies that orthogonality is also preserved. This means that the Jacobi's rotational method which multiple orthogonal transfromations preserves the orthogonality and dot product of the columns in the system we transform.

4.2 Eigenvalues of one electron in the harmonic oscillator

Using one electron in the harmonic oscillator and solving using the Jacobi's method we get the following result on the apporixmation of the analytical eigenvalues $\lambda = 3, 7, 11, 15$ for the system, varying the number of integration points N:

	N						
Absolute error	10	25	50	75	100	150	200
$\lvert\lambda_1 - \hat{\lambda}_1\rvert$	8.05E−2	1.25E−2	3.13E−3	1.39E−3	7.81E−4	3.47E−4	1.95E−4
$\lvert\lambda_2 - \hat{\lambda}_2\rvert$	4.16E−2	6.30E−2	1.57E−2	6.95E−3	3.91E−3	1.73E−3	9.74E−4
$\lvert\lambda_3 - \hat{\lambda}_3\rvert$	1.06	1.54E−1	3.81E−2	1.68E−2	9.34E−3	4.04E−3	2.18E−3
$\lvert\lambda_4 - \hat{\lambda}_4\rvert$	2.08	2.83E−1	6.54E−2	2.57E−2	1.18E−2	2.02E−3	1.42E−3

Table 1: Absolute error between the analytically known eigenvalues of the one-electron energies and our numerically calculated eigenvalues.

4.3 Quantum dots, plots

5 Discussion

5.1 Jacobi method

The Jacobi method is generally a slower algorithm than those based tridiagonalization. However the Jacobi method is easy to parallelize which can improve performance significantly. The main reason for the Jacobi methods slow convergence is that for each new rotation, matrix elements which were zero might change to non-zero values.

For the case with one electron we would have needed to choose $N > 200$ in order to get numerical eigenvalues, $\hat{\lambda}_i$ that are able to reproduce the analytical ones, λ_i with four leading digits after the decimal point (which means an absolute error $< 10^{-4}$). The Jacobi method is very slow for too large N values, and doing the calculations with an $N > 200$ a modern laptop takes several minutes to complete the calculations.

It would be of interest to investigate more effective methods like the Householder's method for large values of N, and see what differences a significant increase in integration points would result in.