Eigenvalues and Eigenvectors

Aleksandar Ivanov

November 4, 2020

1 Problem statement

Using diagonalization, find some of the eigenvalues and eigenvectors of the perturbed Hamiltonian H, defined as

$$H = H_0 + \lambda q^4, \qquad H_0 = \frac{1}{2}\left(p^2 + q^2\right) \tag{1}$$

in the basis $|n^{(0)}\rangle$ of the unperturbed Hamiltonian H_0's eigenvectors, for the values of $\lambda \in [0, 1]$. Program at least on diagonalization method by hand and find the dependence of the results on the matrix size $N \times N$. What's the difference in using $\hat{q}^4$, $\hat{q^2}^2$ and $\hat{q^4}$?

2 Implementation of the Algorithms

The algorithms implemented by hand were Jacobi iteration, Householder reflections for QR decomposition and power iteration. Because we are solving the eigenvalue problem for symmetric matrices, the Householder algorithm for QR decomposition already returns the diagonalized matrix in R and eigenvectors in Q, so further work is not required.

Testing the speed of the implementations versus the built-in algorithm `numpy.linalg.eigh` [1] we get figure 1. In it we see that as expected the optimized, built-in function `numpy.linalg.eigh` performed the best, and the power iteration, being the least optimized of the bunch, performed the worst. We also see that Jacobi performed worse than Householder, which agrees with theory. [3]

A comparison of the errors of the implementations is given in figure 2. The estimate of the error used was the maximal deviation from the actual value of the matrix after multiplying the

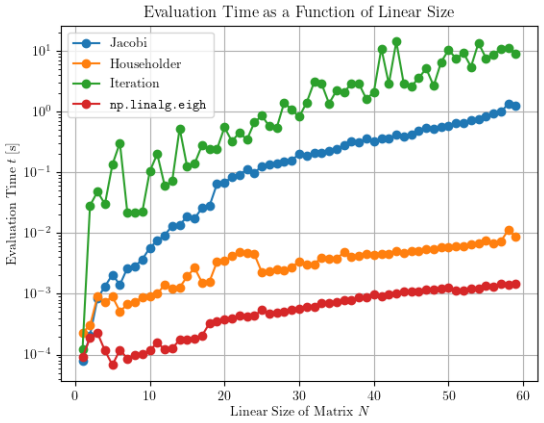

Figure 1: Evaluation times as a function of linear matrix size N for the different implemented algorithms, tested on random dense symmetric matrices.

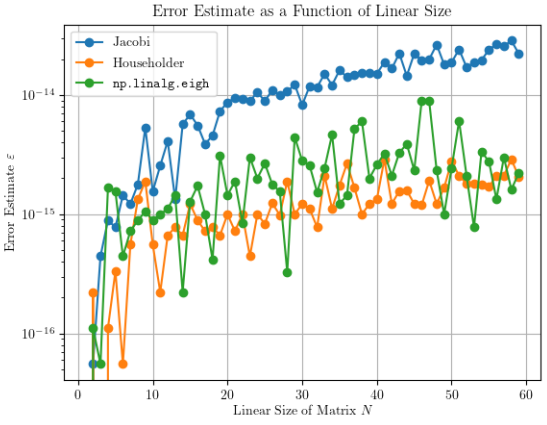

Figure 2: Error as a function of linear matrix size N for the different implemented algorithms, tested on random dense symmetric matrices.

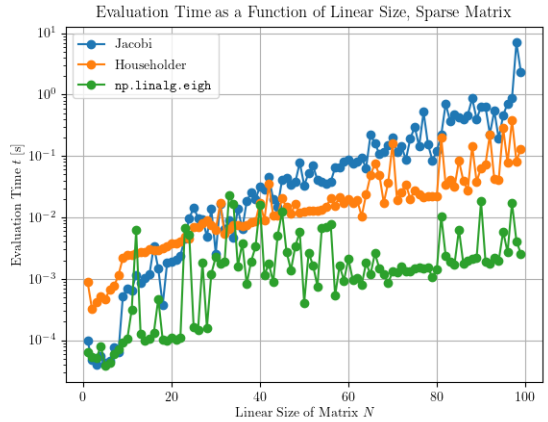

Figure 3: Evaluation times as a function of linear matrix size N for the different implemented algorithms, tested on random sparse symmetric matrices with density $\rho = 0.01$.

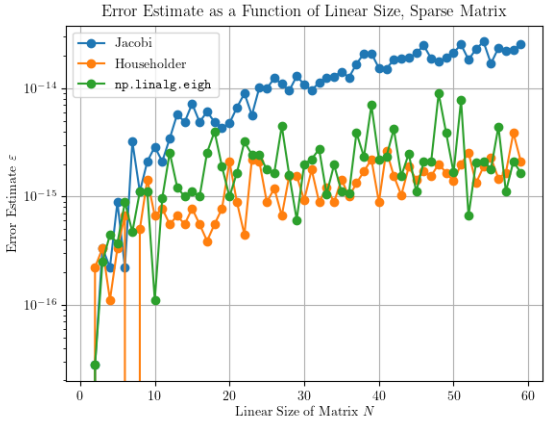

Figure 4: Error as a function of linear matrix size N for the different implemented algorithms, tested on random sparse symmetric matrices with density $\rho = 0.01$.

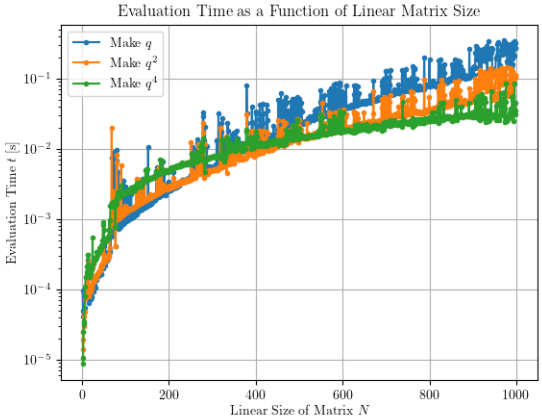

Figure 5: Evaluation time for the creation of the matrix q^4 in the three aforementioned ways, as a function of linear matrix size N.

decomposed element to get the matrix. We see that the built-in `numpy.linalg.eigh` function and the Householder algorithm have comparable errors while the Jacobi iteration process has a somewhat larger, but still quite negligible, error.

Since in our case the matrices are not that dense, a further test for the evaluation times and precision on sparse matrices with density $\rho = 0.01$ was done and the results of that are shown in figures 3 and 4. We see similar results as in the case of dense matrices.

3 On $\widehat{q}^4$ vs. $\widehat{q^2}^2$ vs. $\widehat{q^4}$ [1]

When it comes to the question of which way to implement the perturbation we have three obvious options. One is to implement the creation of the matrix $\widehat{q}$ and multiply that four times to get $\widehat{q}^4$, another one is to implement the matrix $\widehat{q^2}$ and square that to get $\widehat{q^2}^2$ and yet another is to just implement the matrix $\widehat{q^4}$ from the beginning. We do this using one of the three formulae derived from the actions

[1] In this section, hats are used on operators to differentiate the operators that we are implementing, which are under the hat, from the full operator that we use.

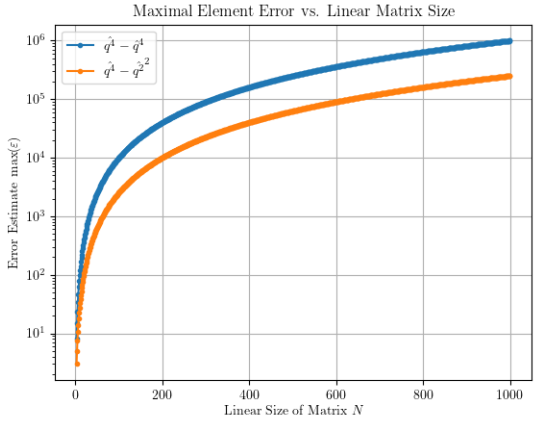

Figure 6: Maximal element error for the aforementioned ways of creating q^4 as a function of the linear matrix size N.

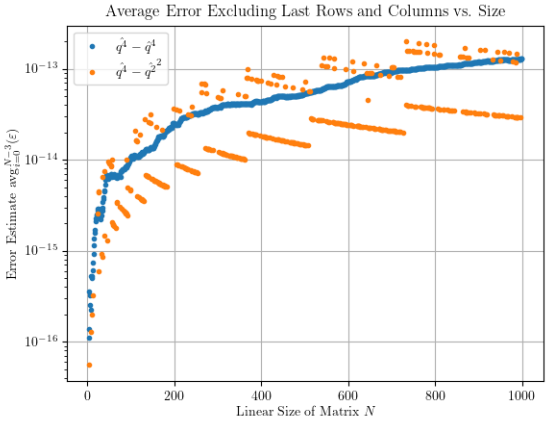

Figure 7: Average element error excluding the last two rows and columns for the aforementioned ways of creating q^4 as a function of the linear matrix size N.

of the creation and annihilation operators [2]

$$\langle i|q|j\rangle = \frac{1}{2}\sqrt{i+j+1}\,\delta_{|i-j|,1}, \tag{2}$$

$$\langle i|q^2|j\rangle = \frac{1}{2}\Big[\sqrt{j(j-1)}\,\delta_{i,j-2} + (2j+1)\,\delta_{i,j} + \sqrt{(j+1)(j+2)}\,\delta_{i,j+2}\Big], \tag{3}$$

$$\langle i|q^4|j\rangle = \sqrt{\frac{2^{i-8}i!}{2^j j!}}\Big[\delta_{i,j+4} + 4(2j+3)\,\delta_{i,j+2} + 12(2j^2+2j+1)\,\delta_{i,j} + 16j(2j^2-3j+1)\,\delta_{i,j-2} + 16j(j^3-6j^2+11j-6)\,\delta_{i,j+4}\Big]. \tag{4}$$

A reasonable assumption would be that matrix multiplication, being an expensive operation, would make the $\widehat{q^4}$ method the most efficient.

The test of this is shown in figure 5 and surprisingly we see that the $\widehat{q^4}$ method only becomes better after $N \approx 500$ and that for matrices smaller than $N \approx 200$ the $\widehat{q}^4$ method is comparable with or even better than the $\widehat{q^2}^2$ method.

Figures 6 and 7 try to show the error we get by using the methods above compared to the definitionally correct $\widehat{q^4}$. At first figure 6, plotting the maximal element deviation from $\widehat{q^4}$, is a bit worrying since the error grows to extremely large values, but 7 provides consolation; it plots the average element deviation from $\widehat{q^4}$, excluding the last two rows and columns. We see that the errors of the $\widehat{q}^4$ and $\widehat{q^2}^2$ are now comparable and hover around 10^{-14} for all tested cases. These two graphs show us that by multiplying the matrices we are propagating the cut-off error up the matrix, but we are saved by the fact that the matrices are almost diagonal, which limits the propagation per multiplication.

Since all three methods are comparable, we will choose to use the $\widehat{q^4}$ method in the rest of this work.

4 Eigenpairs of H

Using the previous algorithms we can diagonalize our matrix H to get its eigenvalues and eigenvectors. We will do this diagonalization using the built-in function `numpy.linalg.eigh` because it did turn out to be the fastest of the tested algorithms.

First we want to test how matrix size N influences the results of the diagonalization, i.e., how fast the eigenpairs converge to their actual values as N becomes larger and larger.

Figure 8 shows us the situation for the eigenvalues. In it we see that higher eigenvalues are more affected than lower ones. This is in agreement with the error propagation argument from section 3. The lesson is then, that we need to use much

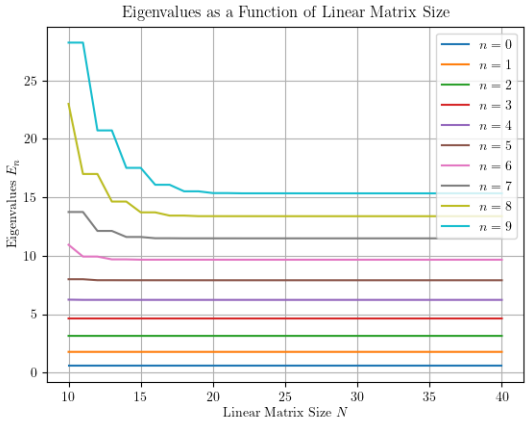

Figure 8: The first 10 eigenvalues for $\lambda = 0.1$ as a function of the linear matrix size N.

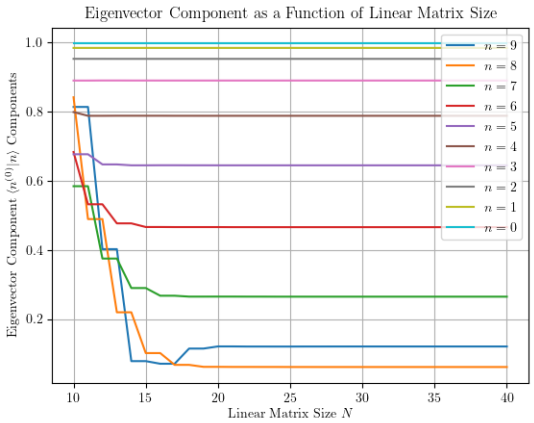

Figure 9: The first 10 $\langle n^{(0)}|n\rangle$ for $\lambda = 0.1$ as a function of the linear matrix size N.

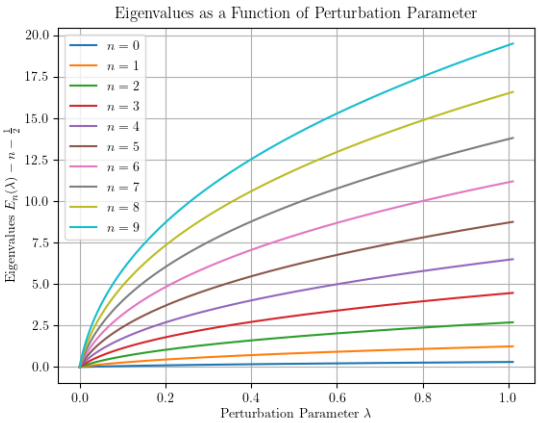

Figure 10: The first 10 eigenvalues E_n minus their unperturbed counterparts $E_n^{(0)} = n + \frac{1}{2}$ for $N = 100$ as a function of the perturbation parameter λ.

bigger matrices than the number of eigenvalues we want to get correctly.

The eigenvector calculation is linked to the eigenvalue one so we expect a similar behavior. We expect the last few components to be the biggest problem, and that is exactly what figure 9 shows; when plotting the largest component of the first 10 eigenvectors, vectors with larger n need bigger matrices to converge to their actual values.

Using a matrix of size 100×100 we compute the first 10 eigenvalues' and eigenvectors' dependence on the perturbation parameter λ. The eigenvalues are shown in figure 10, where we see that they are growing functions of lambda, which is to be expected since q^4 is a positive definite perturbation. We also see that at $\lambda = 0$ we get agreement with the unperturbed eigenvalues $E_n^{(0)}$.

For the eigenvectors, we can plot their components in the original basis as functions of lambda. Figures 11, 12 and 13 do this for $|0\rangle$, $|3\rangle$ and $|4\rangle$, respectively. The first thing we notice is that there is no mixing between even and odd states, which makes sense since the perturbation is an even addition to the potential and we are continuously performing the perturbation from 0 to λ. We also see that the biggest increase goes into components that are close in n to the state and above it, as long as λ is on the small side. So, for example, the biggest

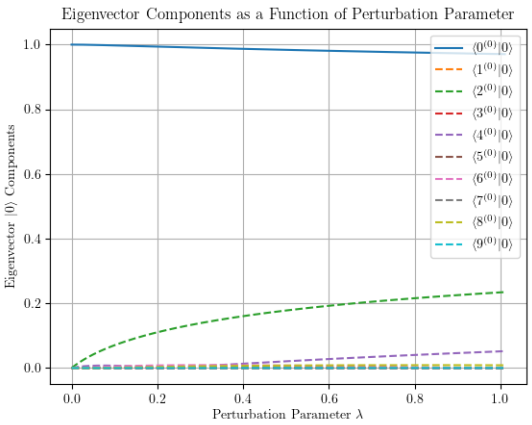

Figure 11: The first eigenvector $|0\rangle$'s components in the original basis $\{|n\rangle\}_{i=0}^{\infty}$ for $N = 100$ as functions of the perturbation parameter λ.

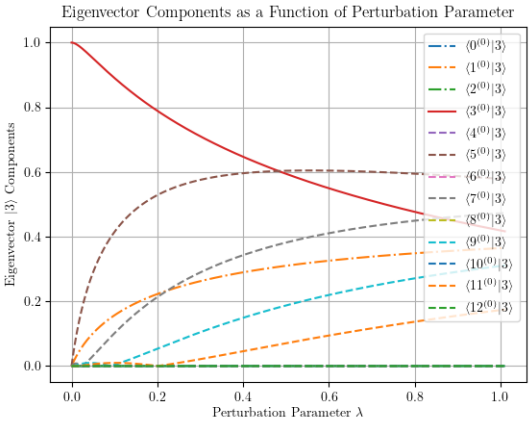

Figure 12: The fourth eigenvector $|3\rangle$'s components in the original basis $\{|n\rangle\}_{i=0}^{\infty}$ for $N = 100$ as functions of the perturbation parameter λ.

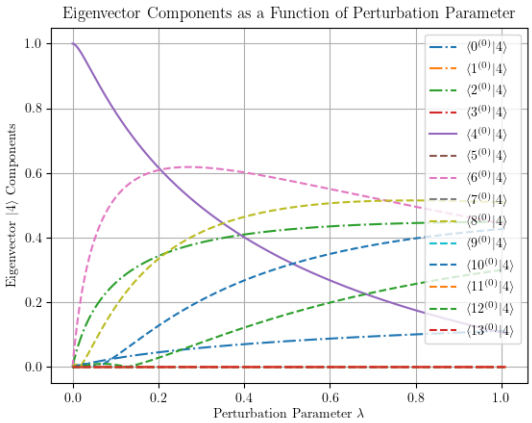

Figure 13: The fifth eigenvector $|4\rangle$'s components in the original basis $\{|n^{(0)}\rangle\}_{i=0}^{\infty}$ for $N = 100$ as functions of the perturbation parameter λ.

component of $|4\rangle$ that isn't in the direction of $|4^{(0)}\rangle$ is in the direction $|6^{(0)}\rangle$. Another thing to notice is that the bigger n is, the faster $\langle n^{(0)}|n\rangle$ is falling.

5 Additional problem statement

Find some of the low-lying eigenvalues and eigenvectors for the double minimum Hamiltonian

$$H' = \frac{p^2}{2} - 2q^2 + \frac{q^4}{2}. \tag{5}$$

6 Eigenpairs of H'

To begin, we rewrite the Hamiltonian H' as a perturbation of the harmonic oscillator Hamiltonian H_0

$$H' = \frac{p^2}{2} - 2q^2 + \frac{q^4}{2} = H_0 - \frac{5}{2}q^2 + \frac{q^4}{2}, \tag{6}$$

we then plug this into our algorithm for $N = 100$ and generate figures 14 and 15. The first few eigenstates are approximately, as expected, just sums of two harmonic oscillator states around each minimum of the potential. Because the perturbation is again even, we see the states keeping their parity distinction.

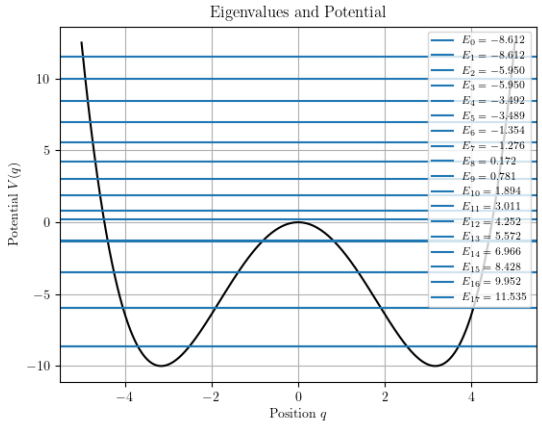

Figure 14: The potential and first 18 eigenvalues of the Hamiltonian H' for $N = 100$.

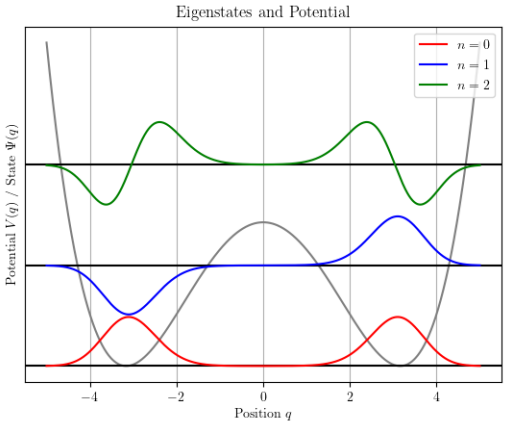

Figure 15: The potential and first 3 eigenstates of the Hamiltonian H' for $N = 100$.

References

[1] The SciPy community. ***numpy.linalg.eigh***. [Last accessed: 28.10.2020]. URL: `https://numpy.org/doc/stable/reference/generated/numpy.linalg.eigh.html`.

[2] David J. Griffiths. "The Harmonic Oscillator". In: *Introduction to Quantum Mechanics*. Pearson Education, 1995, pp. 40–50. ISBN: 0-13-191175-9.

[3] Emil Žagar. *Linearni problem najmanjših kvadratov*. Slovene. 2019.