The Symmetric Eigenvalue Problem

Elijan Jakob Mastnak
Student ID: 28181157

October 2020

Contents

Assignment

Diagonalize and find the eigenvalues, eigenvectors and eigenfunctions of the perturbed Hamiltonian $H = \frac{p^2}{2} + \frac{q^2}{2} + \lambda q^4$ for $\lambda \in [0, 1]$ (explained in the theory section). Diagonalize the matrix $\langle i| H |j\rangle$ using multiple diagonalization algorithms—be sure to implement at least some of the algorithms "by hand". Confirm that $E_n \to E_n^0$ in the limit $\lambda \to 0$. Investigate the dependence of H's eigenvalues and eigenfunctions on the matrix's dimension N. Investigate the convergence of the eigenvalues for large N.

Optional: Perform a similar analysis of the "double-well" Hamiltonian

$$H_{\text{DW}} = \frac{p^2}{2} - 2q^2 + \frac{q^4}{10},$$

which describes a particle in a potential with a double minimum.

---

1 Theory

To jump right to the solution, see Section 2.

1.1 Unperturbed Hamiltonian

In dimensionless form, with energy measured in units of $\hbar\omega$, momentum in units of $(\hbar\omega m)^{1/2}$, and length in units of $\left(\frac{\hbar}{m\omega}\right)^{1/2}$, the one-dimensional quantum harmonic oscillator's Hamiltonian is

$$H_0 = \frac{1}{2}(p^2 + q^2) \tag{1}$$

The unperturbed Hamiltonian eigenstates $\left|n^0\right\rangle$ are

$$\left|n^0\right\rangle = \left(2^n \sqrt{\pi} n!\right)^{-1/2} e^{-\frac{q^2}{2}} \mathcal{H}_n(q) \tag{2}$$

where $\mathcal{H}_n$ denotes the nth Hermite polynomial. These eigenstates satisfy the stationary Schrödinger equation

$$H_0 \left|n^0\right\rangle = E_n^0 \left|n^0\right\rangle, \quad n = 0, 1, 2, \ldots$$

where $E_n^0 = n + \frac{1}{2}$ are the Hamiltonian's non-degenerate eigenvalues in units of $\hbar\omega$. The associated $(N \times N)$ matrix H_0 with elements $\langle i| H_0 |j\rangle$, where $i, j = 0, 1, 2, \ldots, N-1$, is diagonal, with matrix elements $\delta_{ij}(i + \frac{1}{2})$.

1.2 Perturbed Hamiltonian

We add a perturbation term λq^4 to H_0 (Equation 1) to get the perturbed Hamiltonian

$$H = H_0 + \lambda q^4 = \frac{1}{2}(p^2 + q^2) + \lambda q^4$$

where the perturbation parameter λ varies from e.g. 0 to 1. We then ask how the perturbation term changes the original Hamiltonian's energy eigenvalues. Namely, we are interested in the perturbed Hamiltonian H's matrix elements $\langle i| H |j\rangle$ expanded in the basis of unperturbed eigenstates $\left|n^0\right\rangle$. We end up solving the eigenvalue problem

$$H |n\rangle = E_n |n\rangle$$

where the perturbed eigenfunctions $|n\rangle$ are linear combinations of $\left|n^0\right\rangle$.

1.3 Perturbation Matrix Elements

There are a few options for constructing the perturbation matrix Q^4:

1. Find the expected value of $q_{ij} = \langle i| q |j\rangle$ and construct $Q^4 \rightarrow [q_{ij}]^4$
2. Find $(q^2)_{ij} = \langle i| q^2 |j\rangle$ and construct $Q^4 \rightarrow [(q^2)_{ij}]^2$
3. Find $(q^4)_{ij} = \langle i| q^4 |j\rangle$ and construct $Q^4 \rightarrow [(q^4)_{ij}]$

The expected values of the matrix elements are

$$q_{ij} = \frac{1}{2}\sqrt{i+j+1}\delta_{|i-j|,1} \tag{3}$$

$$(q^2)_{ij} = \frac{1}{2}\left[\sqrt{j(j-1)}\delta_{i,j-2} + (2j+1)\delta_{ij} + \sqrt{(j+1)(j+2)}\delta_{i,j+2}\right] \tag{4}$$

$$(q^4)_{ij} = \frac{1}{2^4}\sqrt{\frac{2^i i!}{2^j j!}}\Big[\delta_{i,j+4} + (8j+12)\delta_{i,j+2} + 12(2j^2+2j+1)\delta_{ij} + 16j(2j^2-3j+1)\delta_{i,j-2} + 16j(j^3-6j^2+11j-6)\delta_{i,j-4}\Big] \tag{5}$$

Be sure to experiment with each version of Q^4 when constructing the Hamiltonian H.

2 Solution

2.1 Calculating Perturbation Matrices

To review, the perturbed Hamiltonian can be written $H = H_0 + \lambda Q^4$, where Q^4 can be constructed according to equations 3 through 5. My first step was to decide on an implementation of Q^4 to use for the remainder of the project.

2.1.1 Efficiently Constructing the Q Matrices

The matrices $[q_{ij}]$, $[(q^2)_{ij}]$ and $[(q^4)_{ij}]$ have only two, three and five non-zero diagonals, respectively. Instead of generating the matrices directly with Equations 3 to 5, by calculating each of the N^2 matrix elements in a nested loop, it is more efficient to only initialize the non-zero diagonals and then construct the matrices from these diagonals using e.g Numpy or Matlab's `diag` function. See the Matlab code below for my implementation.

```
function H0=get_H0(N)
% Returns unperturbed QHO Hamiltonian
h_0=0.5:N;  % generates the vector 0.5, 1.5, ..., N-0.5
H0 = diag(h_0);

function Q1=get_Q1(N)
% returns the matrix [q_{ij}]
q1_1 = 1:(N-1);  % first off diagonals--delta(abs(i-j),1) term
q1_1 = 0.5 * sqrt(q1_1+(q1_1+1)+1);
Q1 = diag(q1_1, 1) + diag(q1_1, -1);

function Q2=get_Q2(N)
% returns the matrix [(q^2)_{ij}]
q2_0 = 1:(N);  % main diagonal--delta(i,j) term
q2_0 = 0.5 * (q2_0+(q2_0+1));
```

```
q2_2 = 1:(N-2);  % second off-diagonals--delta(i,j-2) and (i,j+2) terms
q2_2 = 0.5 * sqrt((q2_2+1).*(q2_2+2));
Q2 = diag(q2_0) + diag(q2_2, 2) + diag(q2_2, -2);

function Q4=get_Q4(N)
% returns the matrix [(q^4)_{ij}]
q4_0 = 1:(N);  % main diagonal--delta(i,j) term
q4_0 = 0.75 * (2*(q4_0.^2) + (2*q4_0) + 1);
q4_2 = 1:(N-2);  % second off-diagonals--delta(i,j-2) and (i,j+2) terms
q4_2 = 0.5 * (sqrt((q4_2+1).*(q4_2+2))).*(2*q4_2+3);
q4_4 = 1:(N-4);  % fourth off-diagonal--delta(i,j-4) and (i,j+4) terms
q4_4 = 0.25 * sqrt((q4_4+1).*(q4_4+2).*(q4_4+3).*(q4_4+4));
Q4 = diag(q4_0) + diag(q4_2, 2) + diag(q4_2, -2) + diag(q4_4, 4) + diag(q4_4, -4);
```

2.1.2 Comparing Accuracy and Computation Time for the Q Matrices

I constructed H with each implementation of Q^4 with reference to the above functions:

```
H0 = get_H0(N);  % unperturbed Hamiltonian; N is matrix dimension
H1 = H0 + lambda*get_Q1(N)^4;
H2 = H0 + lambda*get_Q2(N)^2;
H4 = H0 + lambda*get_Q4(N);
```

I then tested the performance of each implementation of Q^4 by measuring the time required to diagonalize a 2000×2000 Hamiltonian for $\lambda = 0.1, 0.4, 0.7, 1.0$ using Matlab's `eig` function. Table 1 shows the results—computation time increases monotonically from `H1` to `H2` to `H4`, which is, I suspect, because of the extra matrix multiplications required when calculating $\left[q_{ij}\right]^4$ and $\left[(q^2)_{ij}\right]^2$.

	t_1 [s]	t_2 [s]	t_3 [s]	t_{avg} [s]
$\left[(q^4)_{ij}\right]$	12.93	12.99	12.91	12.94
$\left[(q^2)_{ij}\right]^2$	13.31	13.53	13.71	13.51
$\left[q_{ij}\right]^4$	14.69	14.05	14.39	14.38

Table 1: Comparison of three runs of the computation time needed to diagonalize a 2000×2000 Hamiltonian H four times for the values $\lambda = 0.1, 0.4, 0.7, 1.0$ using each construction of the matrix Q^4. Using $(q^4)_{ij}$ is fastest. Tested with Matlab's `eig`.

I performed one more test: I found the eigenvalues of a 500×500 Hamiltonian using each implementation of Q^4, calculated the average value of the three implementations, and then plotted the deviation of each implementation from the average value (see Figure 1).
The results surprised me! There are two main takeaways:

1. After a large enough index n (roughly $n \approx \frac{N}{4}$), the eigenvalues begin to both oscillate about and diverge exponentially from the average value. In this regime, the eigenvalues transition from linear to roughly exponential growth and grow non-physically large, for example $E_{500} = 6.57 \times 10^5$ for $\lambda = 0.4$. Table 2 shows the onset index of divergence for various values of the parameter λ and matrix dimension N.

   As a side note, I believe the diverge has something to do with the modification of the Hamiltonian H_0 with the perturbation Q^4, but—under time pressure—I did not investigate further.

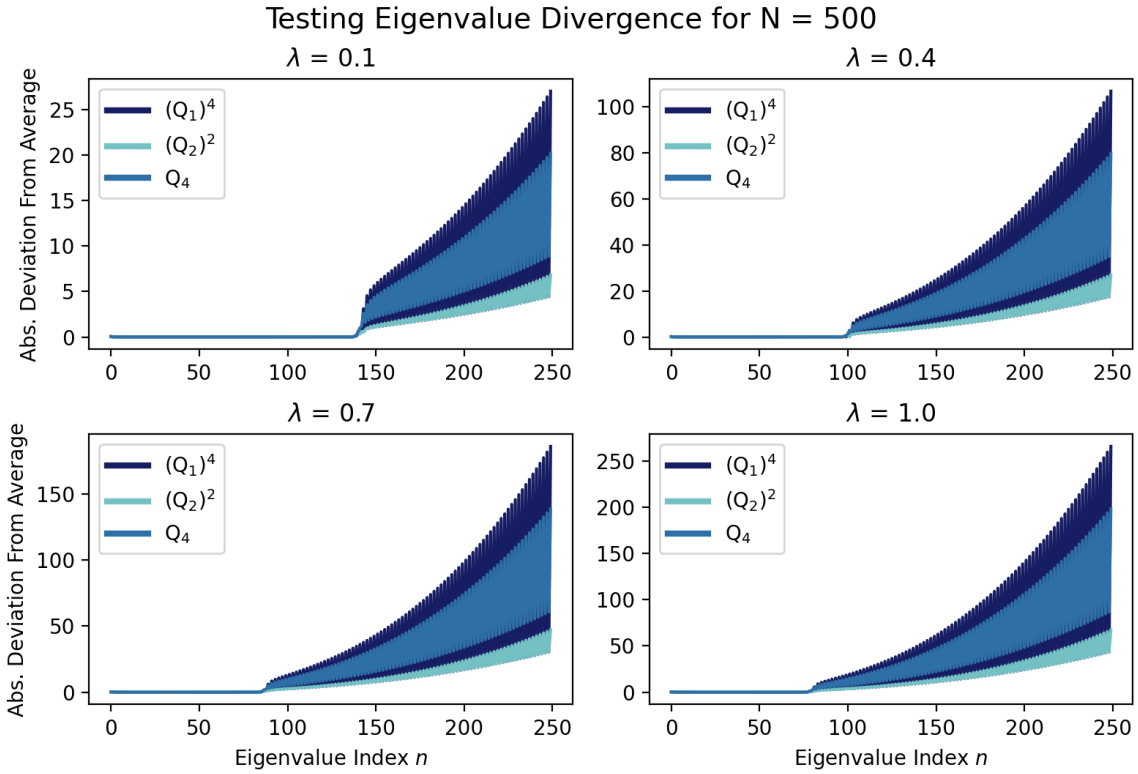

Figure 1: Absolute value of the deviation of eigenvalues found with each implementation of Q^4 relative to the average value of the three methods together. The deviation diverges for large index n—note the earlier divergence for larger λ. See also Table 2.

2. For small indices, before the divergent regime, the deviation of each method from the average is essentially zero.[1] In other words, as long the eigenvalues are small enough, the three implementations of Q^4 give equivalent results.

With these results in mind, I decided to use $Q^4 = \left[(q^4)_{ij}\right]$ simply because it is fastest. As a bonus, eliminating the need for matrix multiplication when calculating the Q^4 should theoretically minimize the accumulation of floating-point arithmetic error relative to the other two options.

λ	$N = 10$	$N = 100$	$N = 500$	$N = 1000$
0.1	4	38	137	237
0.4	4	25	97	170
0.7	3	21	86	147
1.0	3	19	78	136

Table 2: Index at which the eigenvalues found with each implementation of Q^4 begin to diverge from their average value—see also Figure 1. Eigenvalues diverge earlier with both increasing λ at fixed N and increasing N at fixed λ. Tested with Matlab's `eig`.

Two notes: First, for the remainder of this report, I exclusively calculate H with $Q^4 = \left[(q^4)_{ij}\right]$. Second, whenever performing analysis on eigenvalues, I considered only those

[1] To be precise, the deviation from the average is of the order 10^{-5} to 10^{-10}, but this is negligible relative to eigenvalues of order 1 to 100.

values in the non-divergent regime, shown in in Table 2.

2.2 Eigenvalue Methods

2.2.1 Overview of What I Implemented By Hand

Let $\mathbf{H} \in \mathbb{R}^{N \times N}$ be a symmetric matrix. I implemented the following methods by hand:

- Two Jacobi methods for finding eigenvalues and eigenvectors without initial tridiagonalization: the classic method in which rotation matrices target the largest absolute value off-diagonal element, and a cyclic version in which the rotation matrices proceed cyclically through $\mathbf{H}$.

  Applied to $\mathbf{H}$, both methods return an eigenvalue matrix $\mathbf{D}$ and orthogonal transformation matrix $\mathbf{V}$ such that $\mathbf{D} = \mathbf{V}^T\mathbf{A}\mathbf{V}$ and $\mathbf{V}$'s column vectors are $\mathbf{A}$'s eigenvalues.

- QR decomposition using Givens rotations. Returns an orthogonal transformation matrix $\mathbf{Q}$ and upper-triangular matrix $\mathbf{R}$ such that $\mathbf{H} = \mathbf{QR}$.

- An iterative QR eigenvalue method using the above Givens rotation for the QR decomposition. When applied to $\mathbf{H}$ returns matrices $\mathbf{V}$ and $\mathbf{D}$ as described above.

- A two-phase algorithm using Householder reflections for tridiagonalization, followed by QR iteration for the final diagonalization. The Householder method returns a tridiagonal matrix $\mathbf{T}$ and the corresponding orthogonal transformation matrix $\mathbf{P}$ such that $\mathbf{T} = \mathbf{P}^T\mathbf{A}\mathbf{P}$ and the QR method returns $\mathbf{V}$ and $\mathbf{D}$ as described above. The complete transformation matrix from $\mathbf{H}$ to $\mathbf{D}$ is thus $\mathbf{Z} = \mathbf{PV}$.

These methods are all generic, so I see no added value in describing them in greater detail here—see the attached source code files for the exact implementation.

2.2.2 Stopping Conditions for Iterative Methods

I exited the iterative QR method when the Hamiltonian matrix's offset value, defined as

$$\text{Off}(\mathbf{H}) \equiv \sum_{i \neq j} |\mathbf{H}_{ij}|^2,$$

(i.e. the absolute value sum of the off-diagonal elements) fell below a prescribed tolerance $\epsilon = 10^{-10}$. The choice of ϵ is somewhat arbitrary here. I exited the two Jacobi methods on the condition

$$\text{Off}(\mathbf{H}_k) < \frac{1}{n-1}\left(\frac{\epsilon_J}{2n-1}\right)^2$$

with $\epsilon_J = 10^{-4}$. This condition guarantees an upper bound of ϵ_J on the accuracy of all of $\mathbf{H}$'s eigenvalues—see page 25 of [1].

2.3 Eigenvalue Computation Time and Accuracy

Figure 2 shows the computation time each method required to diagonalize a single $N \times N$ Hamiltonian. Unsurprisingly, Matlab's `eig` superlatively outperforms all of my "hand-made" methods. Out of my implementations, the one-phase QR method performed fastest and could reasonably handle matrices up to about $N \approx 1000$. The two-phase Householder-QR method completed in a reasonable amount of time up to about $N \approx 500$, while the classical Jacobi method failed after $N \approx 100$.
To my pleasant surprise, although the methods' computation times differed by multiple orders of magnitude, the values themselves were nearly identical, as shown in Figure 4.

2.3.1 Brief Case Study: Comparison of Jacobi Methods

The classical Jacobi method, in which the rotation matrix targets the largest element by absolute value, impressed me by considerably outperforming the cyclic variation, in which the rotation matrix proceed in a prescribed order through the to-be-diagonalized matrix. Figure 3 compares the two methods—the cyclic method is useful up to about $N \approx 30$, while the classical method performs reasonably up to about $N \approx 100$.

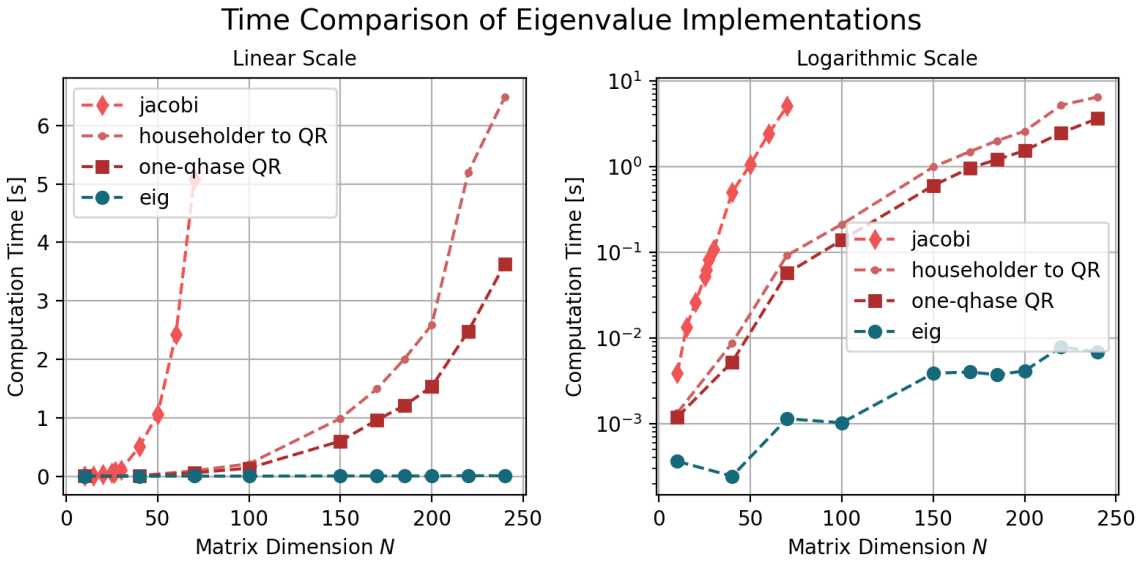

Figure 2: Time required to find the eigenvalues and eigenvectors of an $N \times N$ Hamiltonian. Unsurprisingly, Matlab's `eig` vastly outperforms each of my "handmade" implementations.

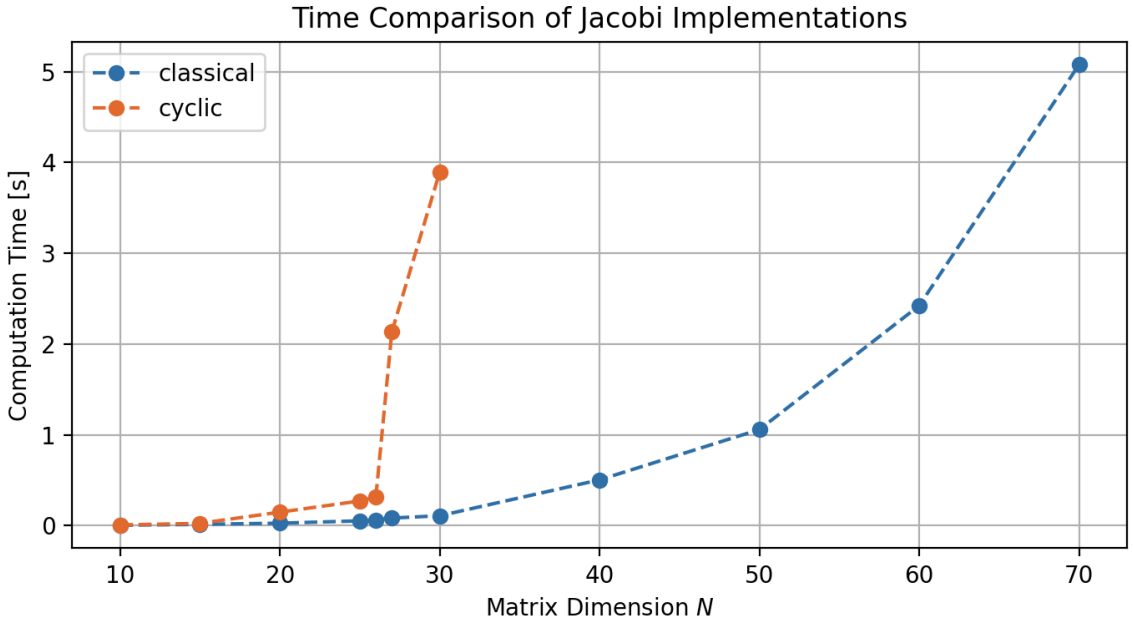

Figure 3: Comparison of the classical and cyclic Jacobi methods (see Section 2.2.1), showing the time required to diagonalize an $N \times N$ Hamiltonian.

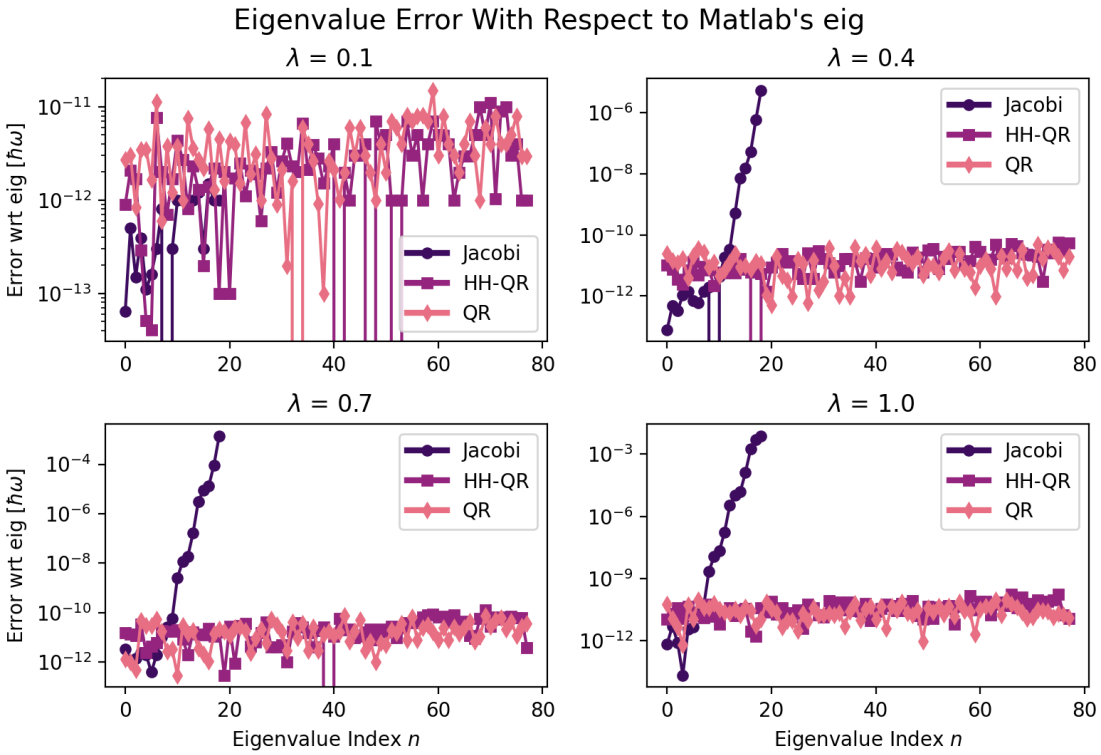

Figure 4: Absolute deviation of eigenvalues from Matlab's `eig` for various "handmade" implementations. Although the Jacobi method fails for large N, both the Householder and QR methods give the same results as `eig` to within an impressive 10^{-11}.

2.4 Eigenvalue, Eigenvector and Eigenfunction Plots

Note: All eigenvalues (and their associated eigenvectors) displayed in the following figures have been sorted from smallest to largest value. Additionally, all eigenvectors are processed so that the signs of the vector components agree across the `eig`, QR, two-phase Householder to QR, and Jacobi methods.

Figure 5 shows H's first ≈ 100 eigenvalues, while Figure 6 shows H's first 20 eigenvalues using a colormap on a two dimensional grid. See Appendices A and B for many more graphs, including the first 100 eigenvectors for both a single and double-well oscillator; I moved the graphs to the appendix to avoid cluttering the main report.
Figures 10, 11 and 12 show the single and double-well oscillator's first few eigenfunctions and probability densities. After diagonalizing the Hamiltonian, I expanded the double-well eigenfunctions in the unperturbed single-well basis from Equation 2, using the components of the nth eigenvector, as abstracted in the following Python code:

```python
def get_psi_expanded(eigenvector, q):
    """
    Returns an oscillating system's eigenfunction corresponding to the inputted
    ↪ eigenvector by expanding the function in the unperturbed QHO basis
    """
    psi = np.zeros(len(q))  # preallocate
    for i in range(0, len(eigenvector)):  # loop through each eigenvector component
        psi += eigenvector[i] * get_psi_qho(i, q)   # linear combination of QHO's
        ↪ i-th basis functions
    return psi
```

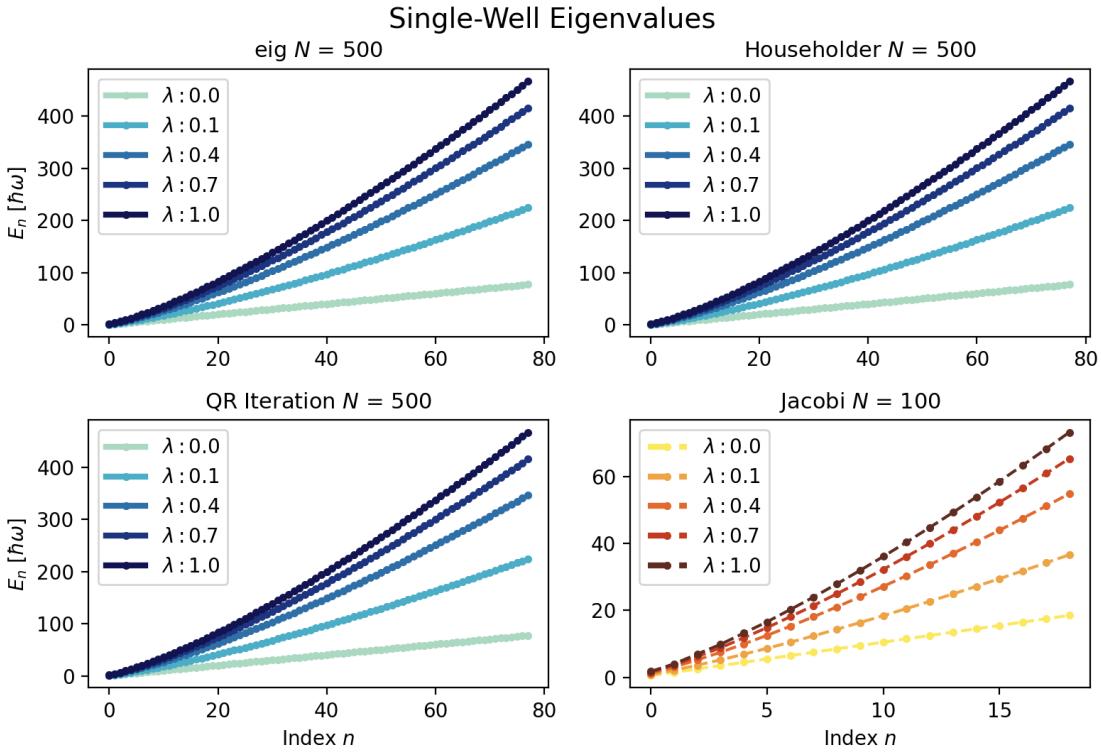

Figure 5: Eigenvalues found with various methods. Although $N = 500$ (100 for the slower Jacobi method) I only plotted eigenvalues in the non-divergent regime shown in Table 2. For small n, values found with each method are nearly identical, as shown in Figure 4.

2.5 Double-Well Oscillator

We can model a double-well oscillator the Hamiltonian

$$H_{\text{DW}} = \frac{p^2}{2} - 2q^2 + \frac{q^4}{10} = \frac{1}{2}(p^2 + q^2) - \frac{5}{2}q^2 + \frac{q^4}{10} = H_0 - \frac{5}{2}q^2 + \frac{q^4}{10}$$

I constructed the Hamiltonian in the unperturbed single-well basis using an analogous procedure to the one described in Subsubsection 2.1.1, using $\left[(q^2)_{ij}\right]$ for the q^2 term and $\left[(q^4)_{ij}\right]$ for the q^4 term. The rest of the analysis for the double-well oscillator is analogous to the single-well version described above, and I see no reason to reiterate it here.
Figure 7 shows the double-well oscillator's first ≈ 100 eigenvalues, while Figure 9 (in the Appendix) shows the first 20 and 100 eigenvectors, respectively. Note that—unlike for the single-well case—the double-well oscillator's first few eigenvalues are negative.

2.6 Some Ideas I Ran Out of Time For

- How would tridiagonalization with Householder reflections compare (speed, accuracy...) to a Hessenberg decomposition, e.g. with Matlab's `hess`, which returns a tridiagonal matrix for symmetric inputs?
- Similarly, how does a QR decomposition with Givens rotations compare to Matlab's built-in `qr`?
- Compare the speed of a traditional QR iterative eigenvalue method to an implementation using Wilkinson shifts (see e.g. [2]).

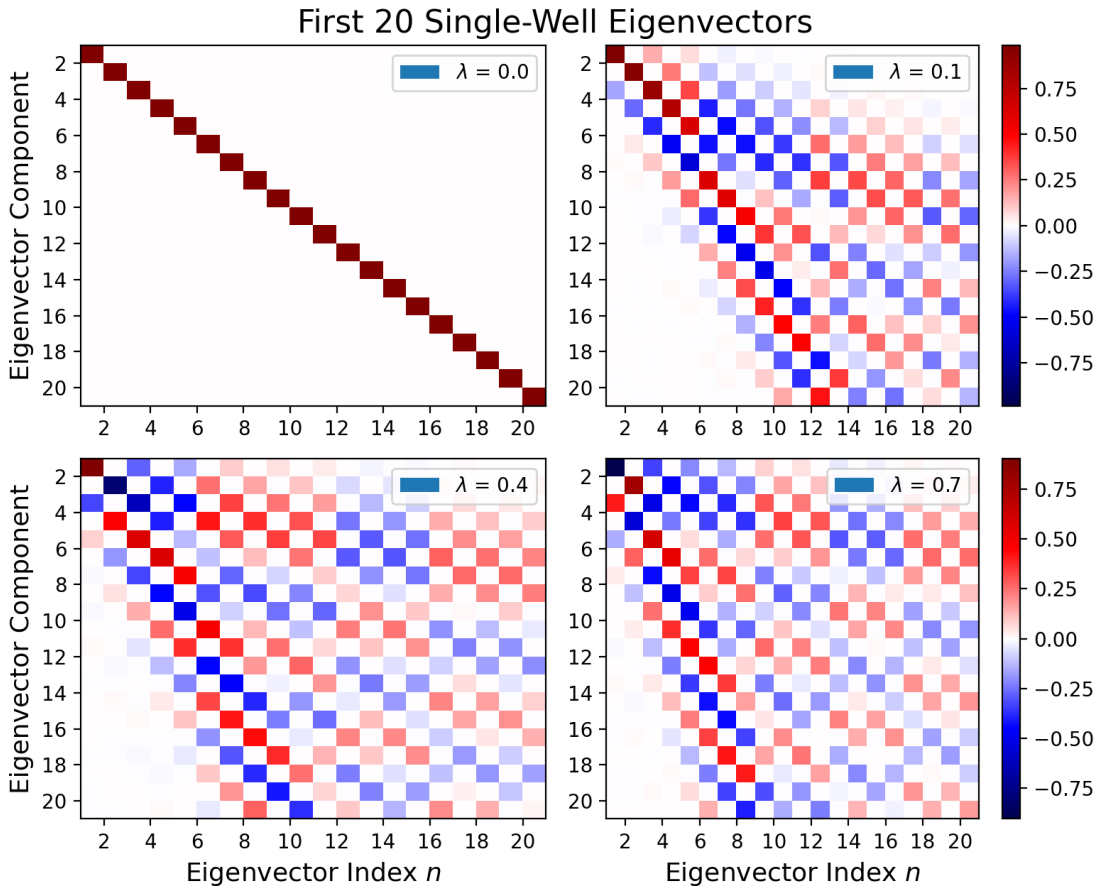

Figure 6: The Hamiltonian's first 20 eigenvectors for various λ. The nth eigenvector corresponds to the grid's nth column; red values are positive and blue values are negative. See Figure 8 in the Appendix for a plot of H's first 100 eigenvalues.

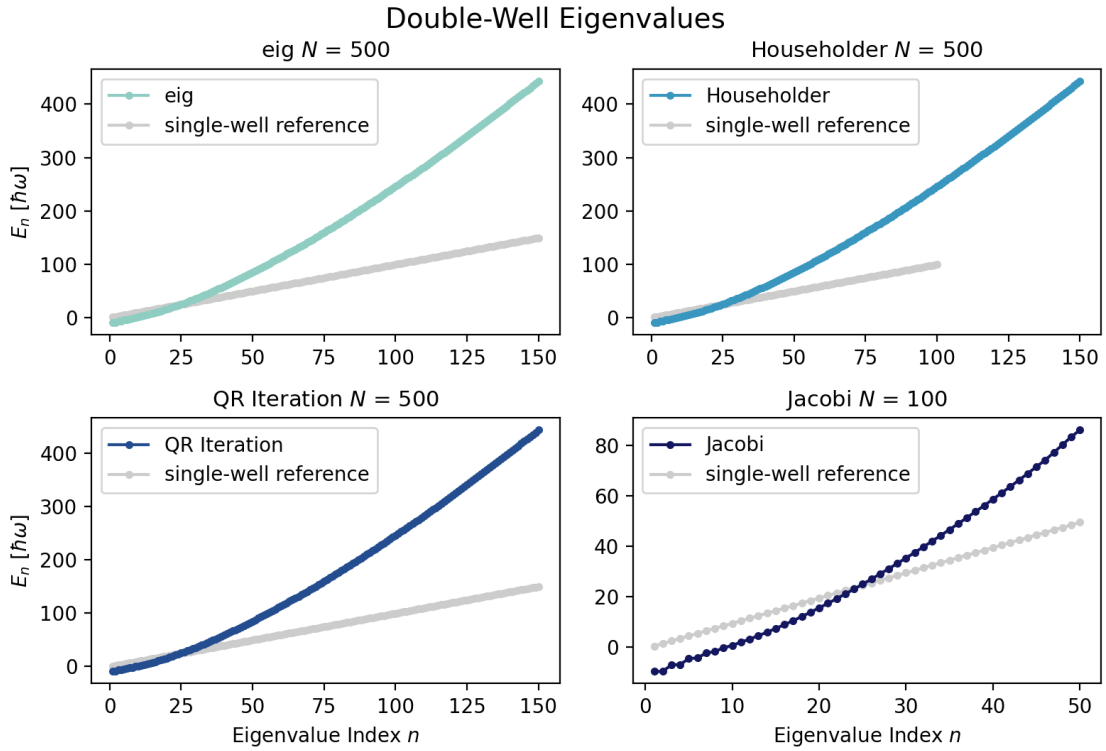

Figure 7: Eigenvalues of the double-well oscillator H_{DW} found with various methods. Although $N = 500$ (100 for the slower Jacobi method) I only included eigenvalues in the non-divergent regime. The unperturbed single-well eigenvalues are shown for reference.

A More Eigenvector Plots

I'm including these here to avoid cluttering the main report with erratically placed figures.

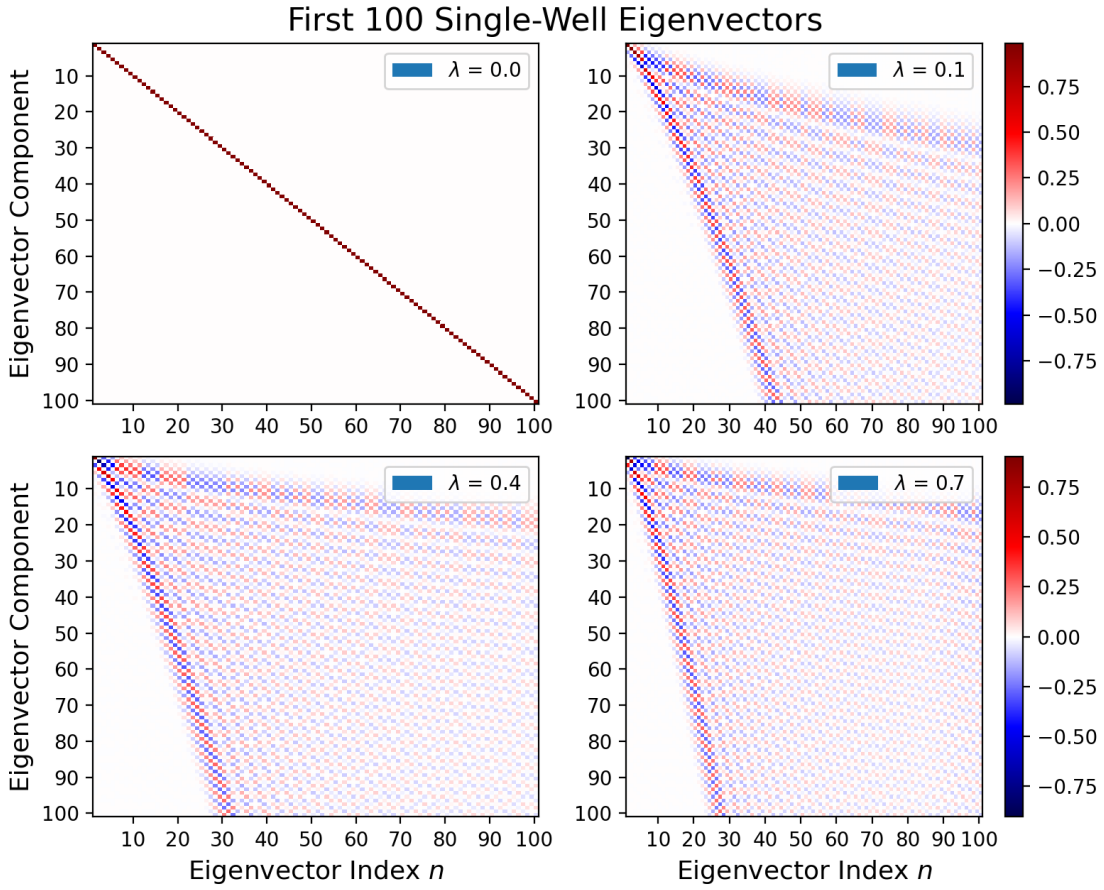

Figure 8: The perturbed Hamiltonian's first 100 eigenvectors for various λ. The nth eigenvector corresponds to the grid's nth column; red values are positive and blue values are negative. Note the principle (more intensely colored) components' increasing divergence from the unperturbed linear trend as λ increases.

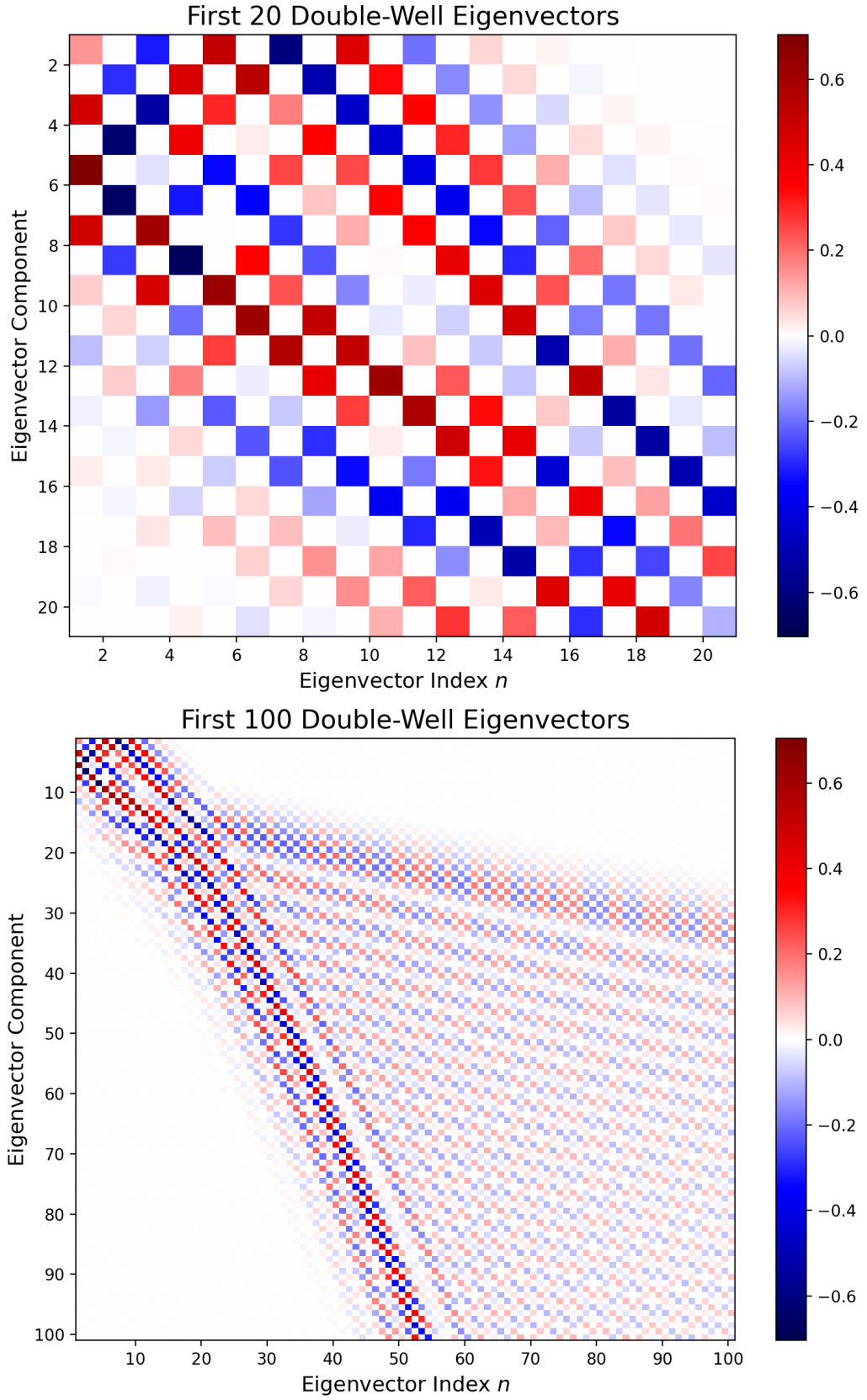

Figure 9: The double-well Hamiltonian's first 20 and 100 eigenvectors. The nth eigenvector corresponds to the grid's nth column. The two diagonal bands of principle (more intensely-colored) components appear slightly wider than in the single-well case in Figure 8, indicating more $|n^0\rangle$ basis functions contribute appreciably to the double-well oscillator.

B Eigenfunction Plots

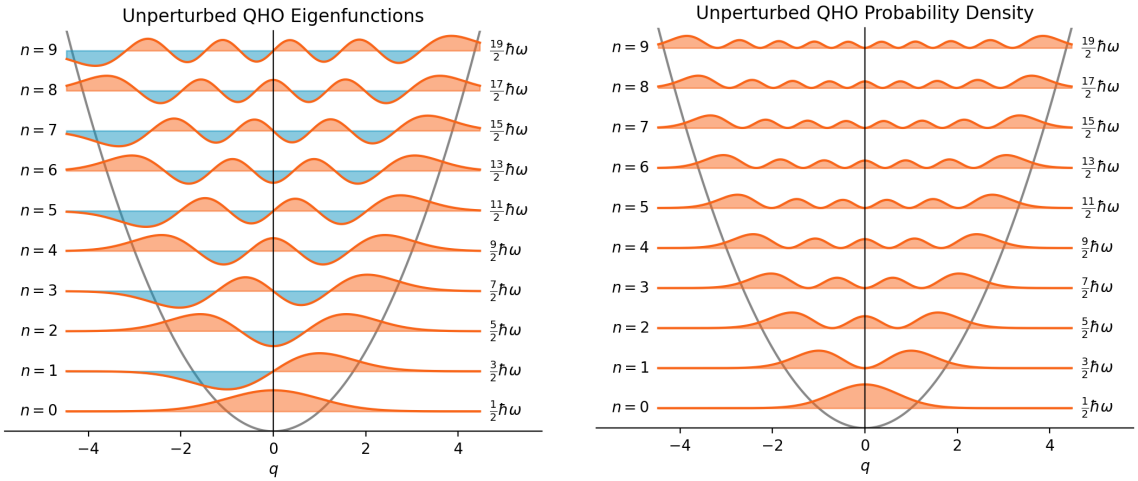

Figure 10: The unperturbed harmonic oscillator's first few eigenfunctions ψ_n and probability densities $|\psi_n|^2$, which nicely show the QHO's linear increase of E_n with n.

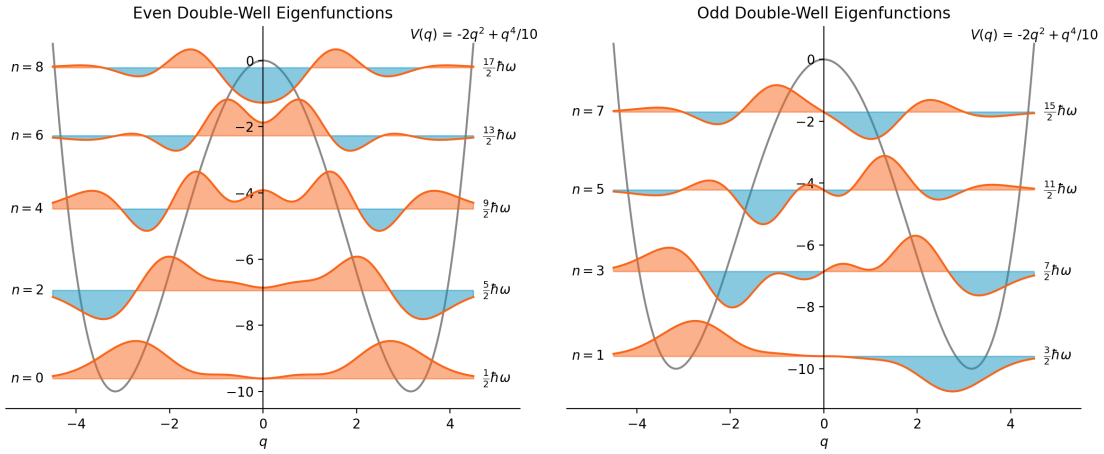

Figure 11: The double-well Hamiltonian's first few eigenfunctions—note that the first few eigenvalues are negative. The even and odd functions are separated to avoid overlap.

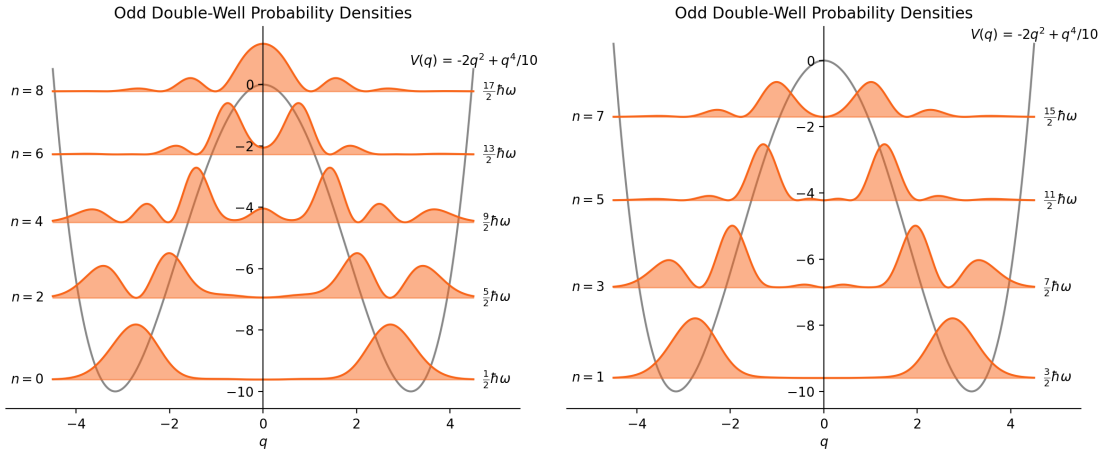

Figure 12: The double-well Hamiltonian's first few probability densities, i.e. $|\psi_n|^2$. The even and odd functions are separated to avoid overlap.

References

[1] Carlsen, Eric. *Chapter 3 of Calculus^{++}: The Symmetric Eigenvalue Problem*. Georgia Institute of Technology. (2006). https://www.math.wustl.edu/~wick/teaching/Math2605Notes/chap3.pdf.

[2] Townsend, Alex. *The QR Algorithm*. (2019). http://pi.math.cornell.edu/~web6140/TopTenAlgorithms/QRalgorithm.html

[3] William H. Press, et al. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge; New York: Cambridge University Press, 1992.

[4] Wilkinson, J. H.. *The Algebraic Eigenvalue Problem*. (1965).