Quantum Information and Computing - Assignment 4

For this assignment, it was required to write an algorithm to numerically solve the eigenvalue problem of a harmonic oscillator Hamiltonian.

I have developed a function with as arguments the density of points per unit distance, the number of eigenvalues to find and two thresholds for eigenvalue deviation and eigenvector deviation, respectively; the last argument is the order at which to approximate the kinetic energy: k=1 takes only first-neighbors into consideration, k=2 takes values up to 2 steps away. All of the arguments are optional.

The thresholds, if present, only come into play at the very last step, printing the index of the first case in which either of the two is surpassed; if they are not present, the function will print out all of the eigenvalue and eigenvector deviations.

To have a cleaner function signature, the integration interval and the harmonic oscillator frequency have been kept as internal parameters, but moving them to function arguments is immediate.

The creation of the diagonal and tridiagonal (or 5-diagonal in the k=2 case) matrices is achieved using the diags function of the scipy.sparse module. The same module contains a submodule for linear algebra (namely, linalg), which takes care of most of the possible operations on such matrices.

Since the algorithm returns the vectors without normalizing, the next step is to find the current norms and renormalize each of them; the result is then ready to be compared to the exact value, given by

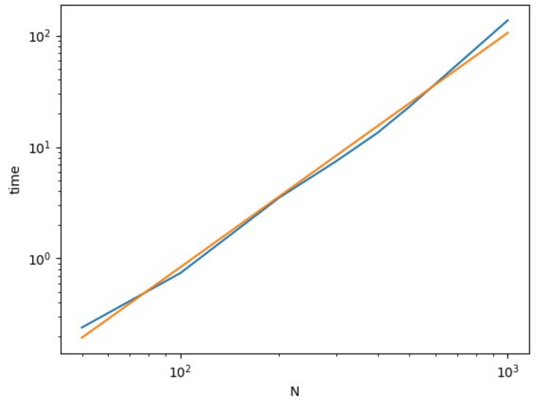
$$\psi_n(x)=\frac{1}{\sqrt{2^n n!}}\left(\frac{m\,\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\,\omega\,x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\,\omega}{\hbar}}x\right),\;\; E_n=\hbar\,\omega\left(n+\frac{1}{2}\right)$$

Where H_n are the Hermite polynomials given by $H_n(z)=(-1)^n e^{z^2}\frac{d^n}{dz^n}\left(e^{-z^2}\right)$.

Throughout the program, the values for m and ħ are set to 1.

The results for 500 points per unit length with the kinetic energy at order 1 are good, with a deviation of less than .0005% with respect to omega of the eigenvalues up to the 10th value. As expected, higher energies correspond to wavefunctions that tend more and more towards the infinite well wavefunctions.

The scaling of the time with respect to N (point density) seems to be a power law with exponent ~2.1, but higher values of N would be needed to confirm the trend.

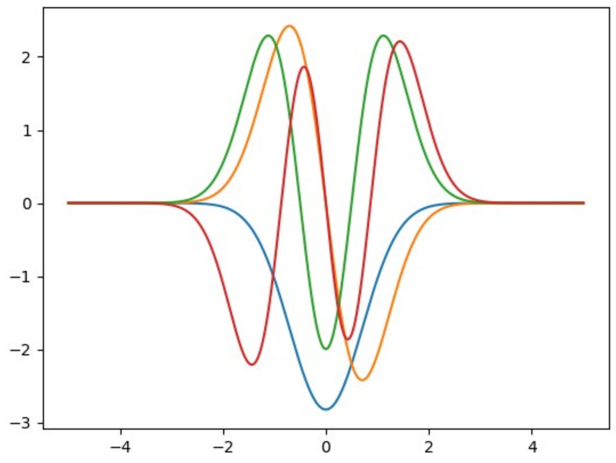


The solution is stable over different runs with K at order 1, with at most a change of overall sign that lead to the decision of squaring the vectors (thus finding the probability density) before calculating the difference from the exact vector (the difference reported is the maximum of the discrepancies at each point).

When increasing N, the number of solution close to being exact increases: for example, with N=1000 we get a .0005% deviation over the eigenvalue only at the 16th value. For low N, the relation between this index and N seems to be linear, with two more eigenvalues within tolerance for every 200 points per unit length; this is not confirmed by the N=1000 point, needing more trials at different values to be estimated better. The behavior of the deviations in the eigenvectors seems to be similar.

As of the date of submission of the assignment, the higher-order approximation for K is already implemented. While currently commented out, the code for obtaining the matrix for the potential given any functional form is already in place, only requiring the addition of U as an argument to the function; this U should be a function itself, and it should be able to operate on a numpy array, giving a numpy array as an output. Should this change be put in place, the part that checks against the exact solution would need to be either changed or removed as no longer relevant.

On the right is the output of the example run:



2
1
0
−1
−2
−3
−4
−2
0
2
4