Homework No.3

Ruqi Feng

September 2022

1 Time Complexity of Gaussian Elimination

1.1 Problem Description

Prove that the time complexity of the Gaussian elimination algorithm is $O(N3)$.

1.2 Proof

Suppose the linear equations to solve is

$$Ax = b \tag{1}$$

and matrix A is an n by n matrix and has full-rank. Then Gaussian elimination involves 2 major steps:

a. Transform into Row Echelon Form Perform row operations to make entries in the coefficient matrix below the diagonal zero. In the i-th column, there are $n-i$ entries that needs to be set zero, which requires $n-i$ row operations. In each row operation, roughly $n-i$ multiplications and $n-i$ additions are required. Therefore, the total operation number needed is approximately

$$N \approx \sum_{i=1}^{n-1} 2(n-i)(n-i) \approx \frac{2n^3}{3} \tag{2}$$

Therefore, this step requires time complexity of $O(n^3)$

b. Back substitution After reducing the coefficient matrix into REF, the solution can be found easily. From bottom to top, find x_n and insert into $A_{n-1}x_{n-1} + A_n x_n = b_{n-1}$ and so on. When solving the i-th linear equation, about $3i$ operations are need, so the time complexity of this step is $O(n^2)$

Using the addition rule of time complexity, the time complexity of Gaussian elimination is

$$O(n^3) + O(n^2) = O(n^3) \tag{3}$$

2 Transformation into RREF

2.1 Problem Description

Write a general code to transform a n*m matrix into the REDUCED ROW ECHELON FORM, and use the code to obtain the RREF of the following matrix. [2, 8, 4, 2; 2, 5, 1, 5; 4, 10, -1, 1]

2.2 My Approach

The procedure to find RREF is similar to Gaussian elimination. There are two majors steps which I call 'forward elimination' and 'backward elimination'.

In 'forward elimination', entries below the diagonal will be set to zero and the diagonal entries will be set to 1. Before setting the j-th column, the j-th row ($j \leq n$) is rescaled so that the diagonal entry in it is set to 1. If the j-th diagonal entry is zero or too small, appropriate entries (a_{xj} where $x > j$) in rows below will be found and those rows are swapped with the j-th row.

However, if a certain row does not have a valid pivot and every entry below it is invalid as well, the RREF does not exist in this case. This is because the entries the diagonal entry in this row will not be able to be eliminated in the backward elimination. My program will print an error message when calling the print function of the returned instance in this case.

Then the entry a_{ij} (suppose $j < i$) is set to zero by multiplying the j-th row by $-a_{ij}$ and add it to the i-th rows. After forward elimination, the matrix will become

$$\begin{pmatrix} 1 & a_{12} & a_{13} & b_1 \\ 0 & 1 & a_{23} & b_2 \\ 0 & 0 & 1 & b_3 \end{pmatrix} \tag{4}$$

or with rows of all zero entries in the lowest rows

$$\begin{pmatrix} 1 & a_{12} & a_{13} & b_1 \\ 0 & 1 & a_{23} & b_2 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & a_{12} & a_{13} & b_1 \\ 0 & 1 & a_{23} & b_2 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5}$$

in which cases, the program will execute the backward elimination procedure. Although in the matrix on the right in Eq.5 the system of linear equations corresponding to the matrix has no solutions, the RREF of it still exists.

If the RREF exists, the program will continue to execute 'backward elimination', where the elements above pivots are set to zero by adding the j-th row, multiplied by $-a_{ij}$, to the i-th row. The result is RREF of the input matrix, then.

2.3 Pseudocode

Algorithm 1 RREF of a given matrix

Input: an input matrix, A
Output: RREF of the input matrix
$n \leftarrow$ row number of A
for $i \leftarrow 1$ to n **do**
  if $|a_{ii}| < 10^{-10}$ **then**
    swap the i-th row with argmax$(A[i, i+1:])$-th row
    if $|a_{ii} < 10^{-10}|$ **then return** 'RREF does not exist'
    end if
  end if
  for $j \leftarrow i+1$ to n **do**
    for $k \leftarrow i$ to n **do**
      $a_{jk} \leftarrow a_{jk} - a_{ji}a_{ik}$
    end for
  end for
end for
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to $i-1$ **do**
    for $k \leftarrow i$ to n **do**
      $a_{jk} \leftarrow a_{jk} - a_{ji}a_{ik}$
    end for
  end for
end for

2.4 Input and Outputs

The matrix required in the problem description is solved. The output is

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 11.0 \\ 0.0 & 1.0 & 0.0 & -4.0 \\ -0.0 & -0.0 & 1.0 & 3.0 \end{pmatrix} \tag{6}$$

as is shown in test case 1.

Test case 2 and 3 are cases where the row of the input matrix is more than the columns. In test case 6 the entries are chosen so that the second column has no valid pivot. My program works fine in these test cases, shown in Fig.1

The source code file Q2_RREF.py can be run in the terminal and no external packages are required.

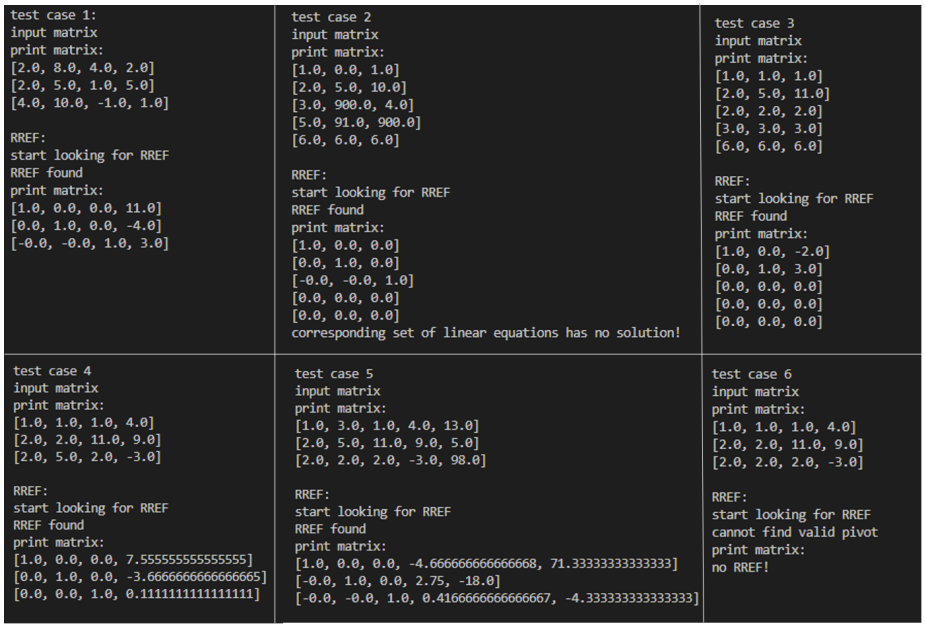

Figure 1: Screenshot of test cases in Q2

3 Solve Schrodinger Equations

3.1 Description

Solve the 1D Schrodinger equation with the potential (i) V(x)= x^2; (ii) V(x)= $x^4 - x^2$ with the variational approach using a Gaussian basis (either fixed widths or fixed centers). Consider the three lowest energy eigenstates.

3.2 Analysis and solution

Searching for a certain state is equivalent to searching for the linear combination of a set of (somewhat appropriate, if not complete) basis. Suppose the basis are ϕ_i and some state can be linearly represented by the basis $\psi = \sum_i c_i \phi_i$. We then want the expectation of the energy of the state ψ to be minimized, that is,

$$\frac{\partial \langle H \rangle}{\partial c_i} = \partial_{c_i} \frac{\sum_{ij} \langle c_i \phi_i | H c_j \phi_j \rangle}{\sum_{ij} \langle c_i \phi_i | \phi_j c_j \rangle} \tag{7}$$

denoting $\langle \phi_i H \phi_j \rangle$ as H_{ij} and $\langle \phi_i \phi_j \rangle$ as S_{ij} (the basis we chose are not necessarily orthonormal).

$$\sum_j c_j H_{ij} = \frac{\sum_{ij} c_i c_j H_{ij}}{\sum_{ij} c_i c_j S_{ij}} \sum_j c_j S_{ij} \tag{8}$$

$$H\vec{c} = ES\vec{c} \tag{9}$$

where E is $\frac{\sum_{ij} c_i c_j H_{ij}}{\sum_{ij} c_i c_j S_{ij}}$ which is, by definition, the expectation value of Hamiltonian. Meanwhile, Eq.9 is a generalized eigenvalue problem, where E is the eigenvalue. The problem can be solved by taking the square root of S and converting Eq.9 into

$$H'\vec{c'} = E\vec{c'} \tag{10}$$

where $H' = S^{-\frac{1}{2}} H S^{-\frac{1}{2}}$, and $\vec{c'} = S^{\frac{1}{2}}\vec{c}$.

In our numerical solution, the matrix calculations are relatively straightforward with the help of all sorts of available linear algebra packages, such as numpy and scipy. What we are left to do is to select a set of proper basis and then calculate H_{ij} and S_{ij}. Integrations are done numerically in my program. As for basis choices, the eigenstates in the harmonic potential Hamiltonian are Gaussian functions, so we choose them as our basis. To construct a set of basis, we need to vary some parameters in the wavefunction. Suppose a Gaussian function is expressed as

$$\phi = (\frac{2\nu}{\pi})^{1/4} e^{-\nu(x-s)^2} \tag{11}$$

where m, $\hbar$ has been set to 1 and ω set to $\sqrt{2}$ (this will keep the harmonic potential in the form of x^2). By choosing a series of ν or a series of s, we are able to produce different basis, as shown in Fig.3 and Fig.2.

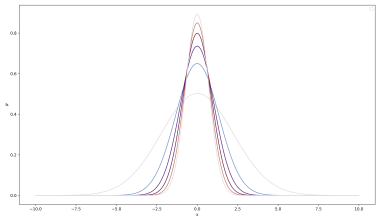

Figure 2: basis with different width

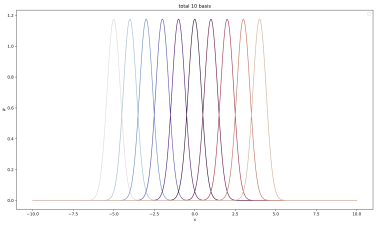

Figure 3: basis with different center position

It is also of importance how many basis to choose and how dense their intervals are. If there are not enough number of basis, the result may be inaccurate, as shown in Fig.4 (the basis have different center location varying in $[-10, 10]$). However, when the number is increased from 25 to 35, the calculated eigenfunctions are more accurate. On the other hand, there can not be too many basis, otherwise the inner product with each other will be close to the inner product with itself. This will result in larger non-diagonal entries in matrix S, and thus potentially produce imaginary eigenvalues of H, which is unphysical. (Though H itself is a symmetrical matrix, we are solving a generalized eigenvalue problem and H' is not necessarily symmetrical) Adding basis in a wider range seems to help but basis 'outside' the range of the potential well can only introduce slight improvement to the solution. It seems that a trade-off is inevitable.

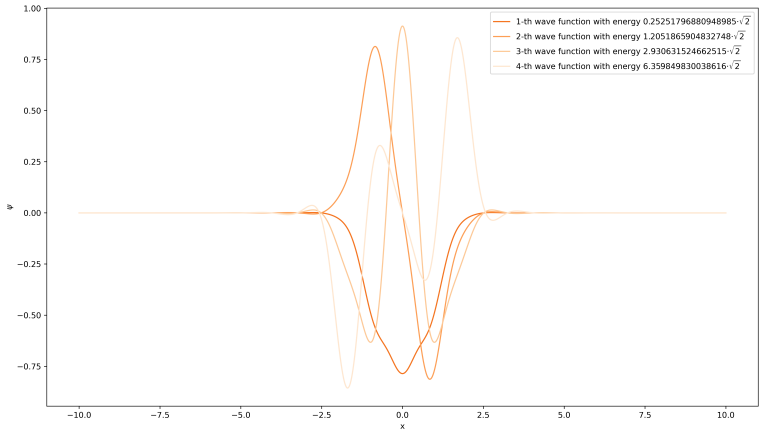

Figure 4: 25 basis, result is not ideal

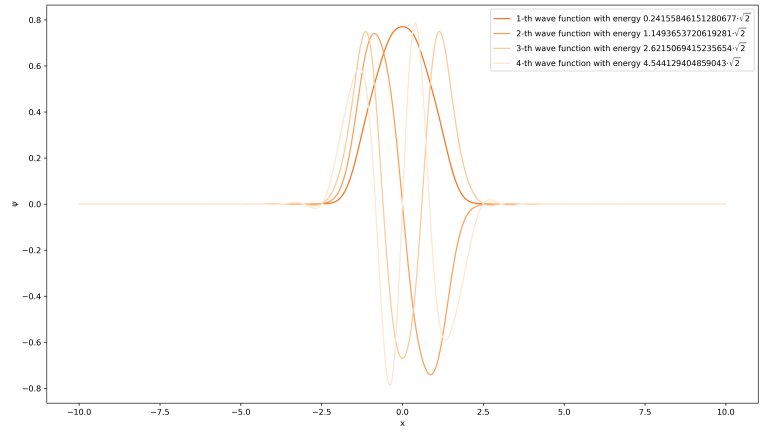

Figure 5: 35 basis, result is more accurate

Luckily, after some trials, I've found that in most cases enough basis are allowed to produce an accurate enough result before failing due to non-positive-definite S matrix. In the following sections, ν will be varied in $[0.1, 1]$ with $s = 0$ while s will be varied in $[-10, 10]$ with $\nu = 3$. Furthermore, I've found better choices of the basis can give precise results with fewer number of basis, which will be included in Sec.3.4.3.

3.3 Pseudocode

Algorithm 2 RREF of a given matrix

Input: n basis ϕ_i and potential $V(x)$
Output: eigenvalues E and eigenstatesψ of the Hamiltonian

for i in range(n) **do**
  for j in range(n) **do**
    $H_{ij} \leftarrow \int \phi_i(\partial_x^2 + V(x))\phi_j dx$
    $S_{ij} \leftarrow \int \phi_i \phi_j dx$
  end for
end for
$D, U \leftarrow$ eigenvalues and matrix of eigenvectors of S
$S^{\frac{1}{2}} \leftarrow UD^{\frac{1}{2}}$
$S^{-\frac{1}{2}} \leftarrow$ inverse of $S^{\frac{1}{2}}$
$H' \leftarrow S^{-\frac{1}{2}} H S^{-\frac{1}{2}}$
$E, C' \leftarrow$ eigenvalues and matrix of eigenvectors of H'
$C \leftarrow S^{-\frac{1}{2}} C'$
$\psi_i \leftarrow \sum_{j=1}^{n} C_{ij}\phi_j$
Sort E and ψ_i according to E

3.4 Results

3.4.1 How to run the program

The file Q3_schrodinger.py can be run from the terminal. Python packages numpy, matplotlib and tqdm are required.

3.4.2 Lowest 3 states

Solve for V_1

The first potential is $V(x) = x^2$, which is the harmonic oscillator potential. As mentioned above, the parameters are chosen so that the potential has the form $\frac{m\omega^2 x^2}{2}$. The theoretically correct eigenvalues of Hamiltonian are $(0.5 + n)\hbar\omega$, where $\hbar = 1$ and $\omega = \sqrt{2}$, respectively. As shown in table3.4.2.

It can be noticed that simply varying ν in Gaussian wave packets cannot produce the first activated state because the basis are even while those states are odd. No matter how many basis are added, as long as they are even, odd eigenstates are 'skipped', as shown in Fig.6. On the other hand, by varying the center, odd functions can be created. In order to overcome this problem, I add a set of odd basis to the even set, namely

$$\phi_i = x(\frac{2\nu}{\pi})^{1/4} e^{-\nu(x-s)^2} \tag{12}$$

It does not matter that the basis are not normalized, because the produced eigenstates will be normalized once more. The resulting eigenvalues are also listed in 3.4.2 and the wave functions are plotted in Fig.7

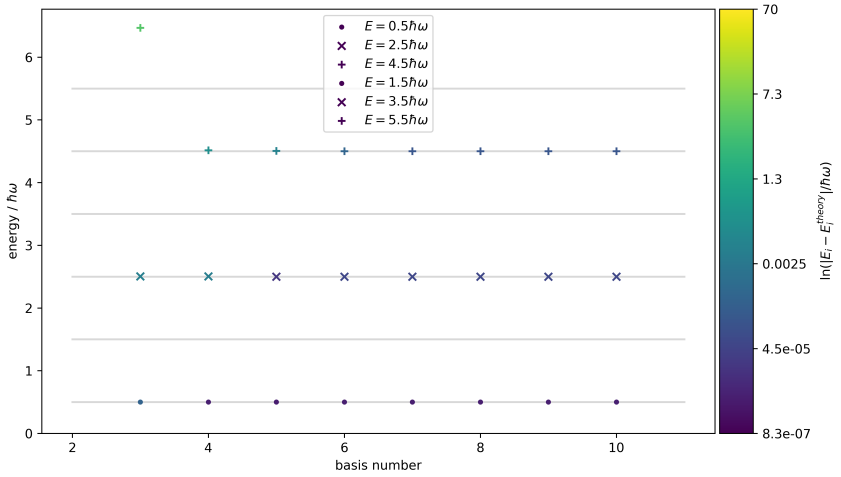

Figure 6: Basis are created by varying width of Gaussian functions. Only even eigenstates of the Hamiltonian are produced, with energy $(2n+\frac{1}{2})\hbar\omega$. The grey lines are theoretical energies and the energies of odd states are 'skipped'.

	basis num	$E_0/\hbar\omega$	$E_1\hbar\omega$	$E_2\hbar\omega$
vary centers s	40	0.50000	1.499998	2.49996
vary widths ν	10	0.50000		2.49994
vary widths ν (even and odd)	20	0.50000	1.499997	2.49999

Solve for V_2

The second potential is $V(x) = -x^2 + x^4$, or $(-(\frac{x}{x_0})^2 + (\frac{x}{x_0})^4)\frac{m\omega^2 x_0^2}{2}$ where $x_0 = 1$, $\hbar = 1$ and $\omega = \sqrt{2}$, to be exact. To find the odd states, we substituted the varying width ν of Gaussian basis with varying width ν of both Gaussian and basis in Eq.12. The varying center s basis are also used and compared with the previous basis. The calculated 3 lowest eigenvalues are shown in Fig.3.4.2, and it can be identified that the two choices of basis give very close results. As is shown in Fig.9 and Fig.8 (they are in theory identical except for an overall phase difference), the wave functions are 'fatter' compared to those in V_1 because the buttom of potential V_2 is wider.

	basis num	$E_0/\hbar\omega$	$E_1\hbar\omega$	$E_2\hbar\omega$
vary centers s	70	0.238970	1.140342	2.589210
vary widths ν (even and odd)	20	0.238984	1.140456	2.589428

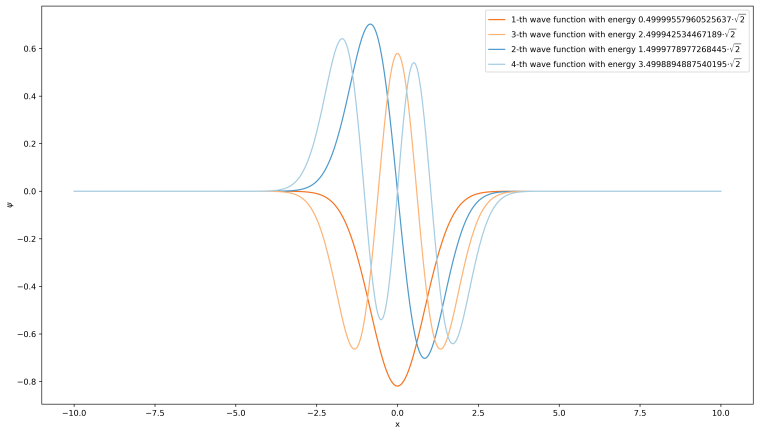

Figure 7: Eigenfunctions of V_1. The blue ones are odd states while the orange ones are even.

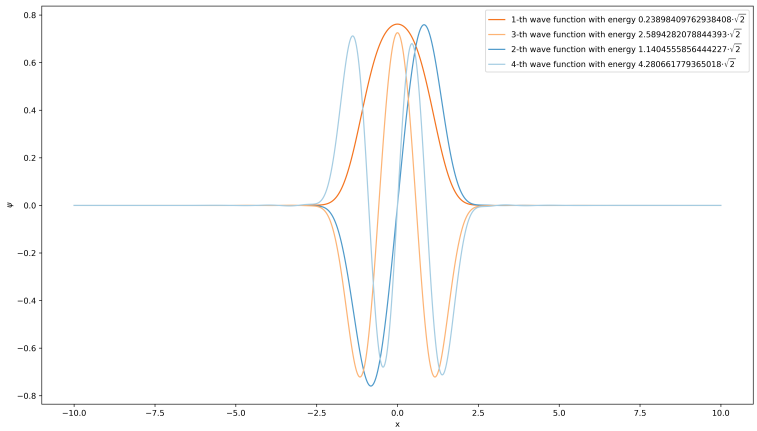

Figure 8: Eigenstates of V_2 produced with basis in Eq.12 and Gaussian functions with varying width

3.4.3 Discussions

As you may have noticed, when choosing the 'varying center Gaussian' basis, the required basis number is larger than 'varying width Gaussian and Eq.12' basis. By changing the number of basis and examining the difference between

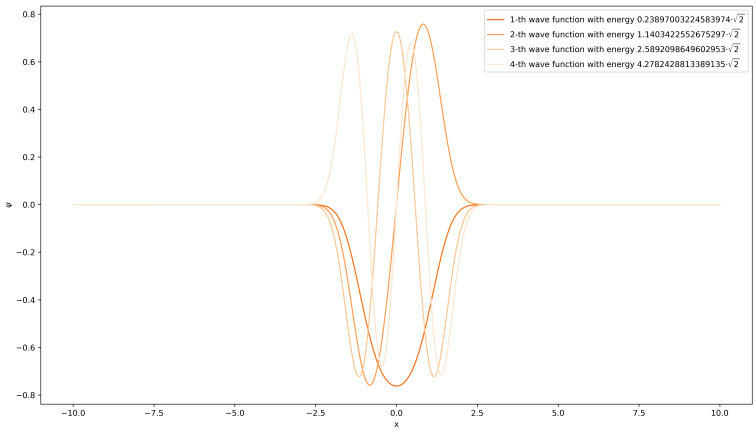

Figure 9: Eigenstates of V_2 produced by Gaussian functions with varying center

the calculated eigenvalues and theoretical values, the difference between the two types of basis can be shown clearly when comparing Fig.11 and Fig.10. I think the reason is that in a symmetrical potential like V_1 and V_2, Gaussian functions or Eq.12 centered at $x = 0$ are closer to actual eigenfunctions. The difference in required number of basis is even more drastic when it comes to V_2, where the 'varying center Gaussian' basis gives eigenvalues hundreds of times of the actual value with few basis numbers while 'varying width Gaussian and Eq.12' basis are already able to give relatively good results. These are shown in Fig.13 and Fig.12.

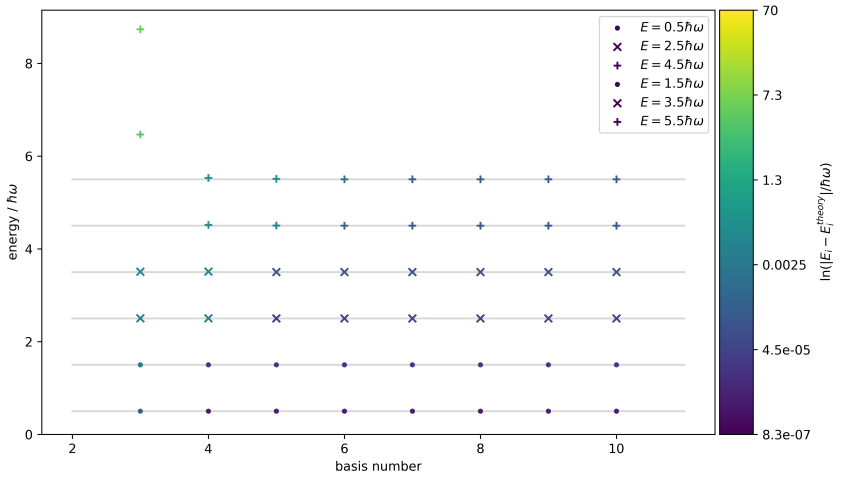

Figure 10: Few basis can produce accurate results. The color characterizes how close the numerical result is to the theoretical value.

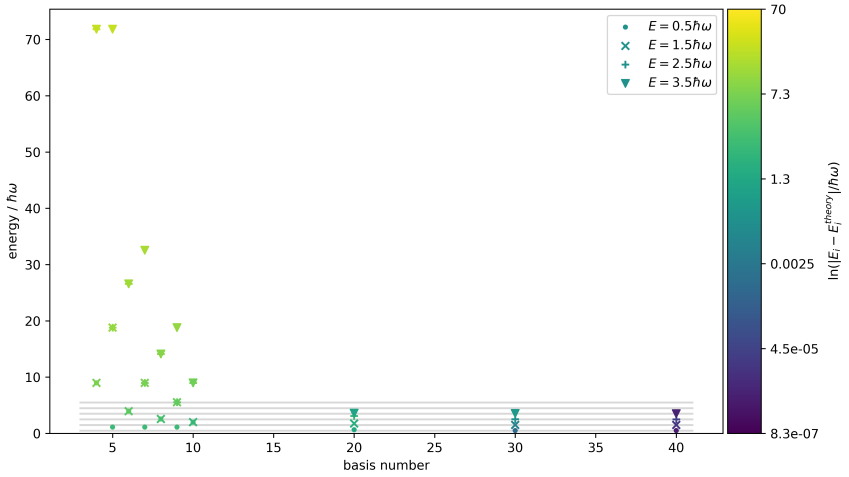

Figure 11: Many basis required to get precise result with varying center basis. The color characterizes how close the numerical result is to the theoretical value.

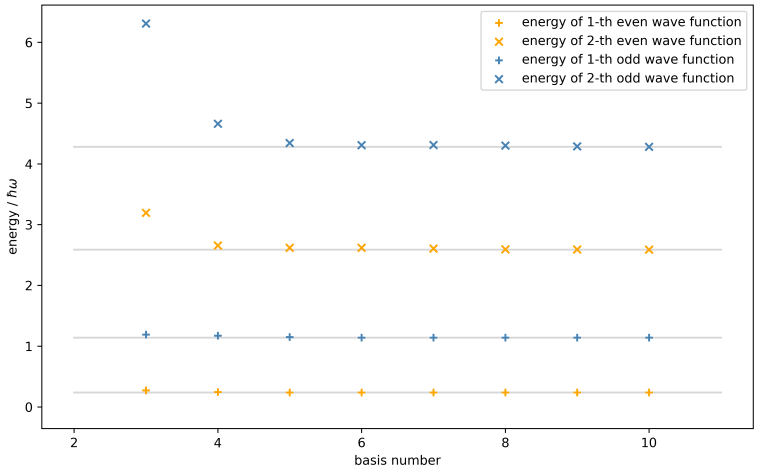

Figure 12: Many basis required to get precise result with varying center basis.

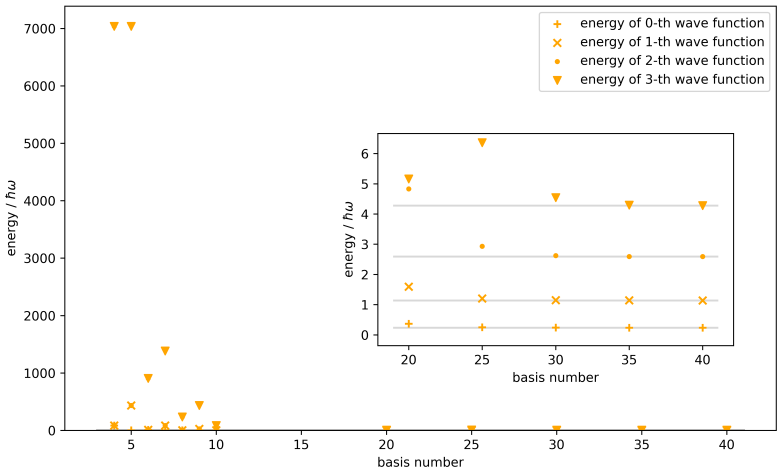

Figure 13: Few basis is required for 'varying width Gaussian and Eq.12' basis to give accurate results