Quantum information and computing: Report 6

Nicola Lonigro
University of Padua
(Dated: November 16, 2020)

The time independent Shrodinger's equation is solved using the corresponding eigenvalue equation for an harmonic oscillator. The validity of the parameters are measured by comparing the relative errors on the first eigenvalues.

I. THEORY

The quantum harmonic oscillator is the quantum-mechanical analog of the classical harmonic oscillator. Because an arbitrary smooth potential can usually be approximated as a harmonic potential at the vicinity of a stable equilibrium point, it is one of the most important model systems in quantum mechanics. Furthermore, it is one of the few quantum-mechanical systems for which an exact, analytical solution is known.

For a single particle the Hamiltonian is

$$H = \frac{p^2}{2m} + \frac{1}{2}\omega^2 x^2$$

In the following we will use m = 1 and $\hbar = 1$. In the static case the solution involves finding the eigenvalues of

$$H\,|\psi\rangle = E\,|psi\rangle$$

By using the representation of the momentum operator in real space and discretizing the corresponding second derivative

$$p = \frac{2}{dx^2}\psi - \frac{1}{dx^2}\left(\psi_{i-1} + \psi_{i+1}\right)$$

the energies of the stationary states can be found from the eigenvalues of the triangular matrix with elements

$$H_{ij} = \delta_{ij}(\frac{2}{dx^2} + \omega^2 x^2) + (\delta_{i+1j} + \delta_{ij+1})(-\frac{1}{dx^2})$$

The energies can be calculated analitically and they are equal to

$$E_n = \left(n + \frac{1}{2}\right)\omega$$

II. CODE DEVELOPMENT

The previous code takes the same parameters as before as input: the input file, the root of the output file and an optional DEBUG flag. In this case the structure of the input file was altered to reflect the parameters of the problem and in particular the input file must now include 4 parameters: the number of points used in the discretization, the mass of the particle (left equal to 1 during the whole exercise but left for generalization), the ω parameter and a length parameter. Note that the code actually uses an interval going from -L to L. The output of the program will be written on three different files: one with all the eigenvales, one with eigenvectors and one that will contain the sum of the relative error on the first eigenvalues.

```
CALL GET_COMMAND_ARGUMENT(1,input_file)
open (unit = 5, file = input_file)
CALL FATAL(  LEN_TRIM(input_file) == 0,      "
  main","Missing input file name" )

CALL GET_COMMAND_ARGUMENT(2,output_rad)
CALL FATAL(  LEN_TRIM(output_rad) == 0,      "
  main","Missing output file name" )
open (unit = 10,Access = 'append', file = TRIM
  (output_rad)//"_egvals.txt")
open (unit = 11,Access = 'append', file = TRIM
  (output_rad)//"_egvects.txt")
open (unit = 12,Access = 'append', file = TRIM
  (output_rad)//"_err.txt")

CALL GET_COMMAND_ARGUMENT(3,call_flags)
IF (call_flags == "DEBUG")     D_FLAG = .TRUE.

read (5,*) n_eig,mass,omega,L

CALL FATAL(   n_eig  <= 0 ,      "main","
  Number of eigenvalues must be positive" )
CALL FATAL(   mass   <= 0 ,      "main","Mass
  must be positive" )
CALL FATAL(   L      <= 0 ,      "main","
  Length must be positive" )
```

The matrix is then initialized with the the values for the elements described in section I.

```
DO i = 1,n_eig
  matrix_1(i,i) = 2.0/dx/dx + (omega*(-1.0*L +
   (i-1)*dx))**2
  IF (i < n_eig) THEN
     matrix_1(i,i+1) = -1.0/dx/dx
     matrix_1(i+1,i) = -1.0/dx/dx
  END IF
END DO
CALL CHECKPOINT(D_FLAG,"After Initialization",
   n_eig,matrix_1,omega,mass,L,N,dx)
```

The eigenvalues and eigenvectors are then computed and the results are printed to the first two files

```
CALL DSYEV('V','L',n_eig,matrix_1,n_eig,egvals
  ,WORK,LWORK,INFO)
CALL FATAL(INFO /= 0, "main", "Eigenvalues not
   found")
```

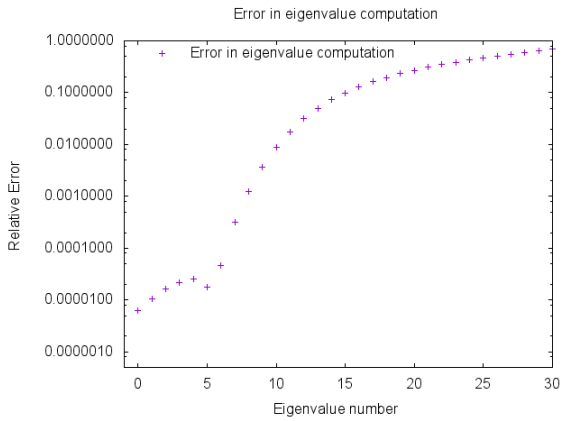

FIG. 1. Relative error for the eigenvalues

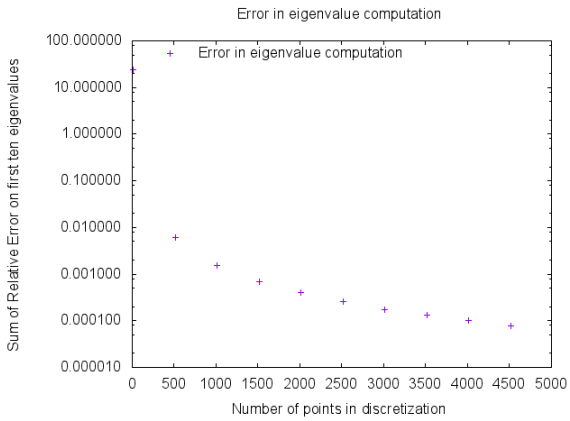

FIG. 2. Relative error for the eigenvalues when changing discretization

```
3
4   write(10,'(E15.5)')egvals
5   write(11,'(E15.5)')matrix_1
6   CALL CHECKPOINT(D_FLAG,"After Computation",
      n_eig,matrix_1,omega,mass,L,N,dx,egvals)
```

Finally the errors in respect to the expected values are computed and the result is printed on the third file. Note the factor two on the expected energies as the hamiltonian requested for the exercise did not have the typical factor $\frac{1}{2}$ present in the definition of the harmonic oscillator.

```
1   DO i = 1,n_eig
2     errs(i) = ((i-0.5)*2*omega - egvals(i))/((i
      -0.5)*2*omega )
3   END DO
4
5   !write(12,'(E15.5)') ABS(errs)
6   write(12,*) n_eig,mass,omega,L,SUM(ABS(errs
      (1:10)))
7   CALL CHECKPOINT(D_FLAG,"Final", n_eig,matrix_1
      ,omega,mass,L,N,dx,egvals)
8 END PROGRAM Exercise
```

To test the code with different values for the parameters, the python code was modified to work with scans over different parameters.

III. RESULTS

The code has been tested with reasonable values for the parameters and the errors on the eigenvalues in ascending order are shown in Fig.1. In particular a value of 1 was used for the ω and a value of 5 for the L, keeping in mind that L scales as $\frac{1}{\omega}$ and the system is in natural units. Different values for N were initially tested and the proposed example has a value of 1000. It is clear that the error is very low for the first seven errors and then it starts increasing rapidly and they are reasonable until the tenth eigenvalue.

Starting from these reasonable values the system was scanned by changing one parameter at at time and checking the sum of the relative errors over the first ten eigenvalues. In Fig. 2 the results when changin the discretization is shown. Even if the error keeps going down when increasing the size it is clear a discretization of 500 is more than sufficient for reasonable results.

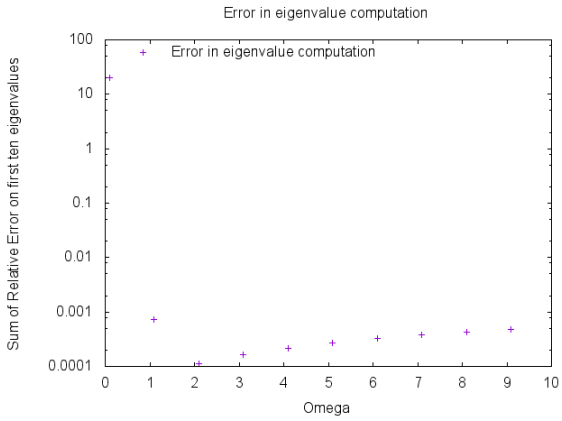

FIG. 3. Relative error for the eigenvalues when changing omega

In Fig. 3 the comparison using a different value of ω is shown. Notice how it is not monotone and there is an interval of maximum efficiency around omega equal to 2. Indeed, at lower values the scaling condition $L \simeq \frac{1}{\omega}$ becomes less true while at higher values an higher discretization is required to account for the stronger oscillations near the center. Finally in Fig.4 a similar behaviour is shown for the L parameter.

IV. SELF-EVALUATION

The correctness of the code has been studied by changing the parameters and so it is possible to optimize the resources used in respect to the precision required from the computation. The main possible problem to the stability

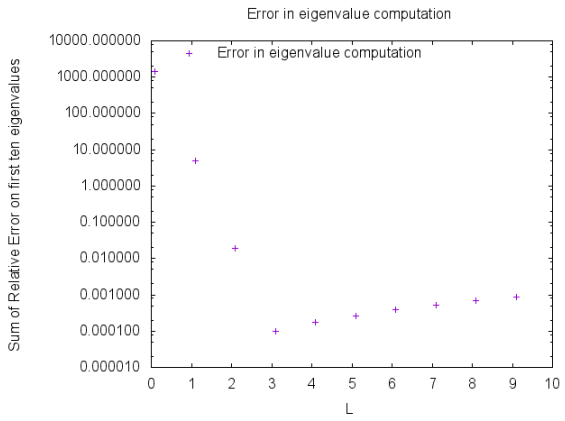

FIG. 4. Relative error for the eigenvalues when changing Length

of the code is the division by dx^2 that may give wrong results for very low values of dx. This could be checked for by a test, requiring the discretization to not be too small. The code has been implemented with very easy ways to check different parameters and avoiding hard coding parameters into the code. This could be improved by also allowing for non natural units and by also asking for an interval to find the solution over, instead of using [-L:L]. The code has been made efficient by the use of the very fast LAPACK library, a possible improvement could be obtained by using the symmetric nature of the matrix to write only one of the two equal sections of the matrix.