Quantum Information and Computing 2020/21
Week 6 report

Rocco Ardino

(Dated: Monday 16th November, 2020)

In this work we provide a numerical solution to the time independent Schrödinger equation for a monodimensional harmonic oscillator in a limited space interval. The method of finite differences is employed for the purpose and the results found for the first k eigenvalues are compared with the theory expectation. Lastly, the first eigenfunctions are plotted in order to check the correctness of the numerical simulation.

1 Theory

Schrödinger equation for a monodimensional quantum harmonic oscillator and for $\hbar = 1$ reads:

$$H\psi_n = \left(-\frac{1}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{1}{2}m\omega^2 x^2\right)\psi_n = E_n\psi_n \quad , \tag{1}$$

being E_n and ψ_n respectively the n^{th} eigenvalue and eigenfunction of the Hamiltonian H. In particular, from theory we know that the eigenvalues read:

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

with $n \geq 0$, and the eigenfunctions are linked to the Hermite functions $H_n(x)$. In x representation:

$$\psi_n(x) = \frac{1}{\sqrt{2^n n!}} \cdot \left(\frac{m\omega}{\pi}\right)^{\frac{1}{4}} \cdot e^{-\frac{m\omega x^2}{2}} \cdot H_n\left(\sqrt{m\omega}x\right) \quad . \tag{3}$$

Now, we focus on the technique to solve numerically this problem. For simplicity, let us limit on a finite and symmetric space interval $[-a, a]$. Then, we discretize it into N smaller intervals of width $\Delta x = \frac{2a}{N}$, which define the points $x_i = -a + i\Delta x$, $i = 0, \dots, N$. With such choice, the finite differences method applied to the second derivative of ψ evaluated in the point x_i returns:

$$\frac{\mathrm{d}^2\psi}{\mathrm{d}x^2}(x_i) = \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})}{(\Delta x)^2} \quad . \tag{4}$$

These tricks translate our problem into a linear algebra one, namely finding the eigenvalues and eigenvectors of the following tridiagonal matrix, representing the discretized Hamiltonian:

$$\begin{bmatrix} \frac{2}{2m(\Delta x)^2} + \frac{1}{2}m\omega^2 x_0^2 & -\frac{1}{2m(\Delta x)^2} & 0 & \cdots & 0 \\ -\frac{1}{2m(\Delta x)^2} & \frac{2}{2m(\Delta x)^2} + \frac{1}{2}m\omega^2 x_1^2 & -\frac{1}{2m(\Delta x)^2} & \cdots & 0 \\ 0 & -\frac{1}{2m(\Delta x)^2} & \ddots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \frac{2}{2m(\Delta x)^2} + \frac{1}{2}m\omega^2 x_N^2 \end{bmatrix} \quad . \tag{5}$$

2 Code Development

For this work, the new module **sch_eq_utils** is implemented, containing several functions and subroutines for multiple purposes:

- **init_ham(N,L,m,omega)**, for instantiating the tridiagonal $(N+1)\times(N+1)$ matrix expressed in Eq. 5, considering a symmteric space interval of length `L` and with `m` and `omega` parameters;
- **diag_ham(ham,eigs)**, for diagonalizing the tridiagonal matrix in Eq. 5 and returning the eigenvalues and eigenfunctions inside the appropriate input arguments, which are modified during the execution;
- **print_eigfc(xs,ys,filename,unit)**, for printing on a certain file the discretized x_i and the corresponding $\psi(x_i)$;
- **print_eigvl(xs,es,eigs,filename,unit)**, for printing on a certain file the eigenvalues E_i, computed from both numerical solution and theory, and the corresponding index i.

In particular, we explain the core functionalities and the workflow in the following subsections.

2.1 Discretized Hamiltonian initialization

The code for the initialization of the discretized Hamiltonian matrix of Eq. 5 is showed in Listing 1. Note that the input parameters are the number of discrete intervals N, the size L of the symmetric space interval $[-a,a]$ and the parameters m and ω.

```
function init_ham(N, L, m, omega) result(ham)
    ! input arguments
    integer(4) :: N     ! hamiltonian dimesnions
    real(8)    :: L     ! width of the box
    real(8)    :: m     ! mass
    real(8)    :: omega ! omega

    real(8), dimension(N+1,N+1) :: ham
    real(8) :: DL
    integer(4) :: ii, jj

    DL = L / N
    do ii=1,N+1
        do jj=1,N+1
            if (ABS(ii-jj)==1) then
                ham(ii,jj) = - (1.0d0/(2.0d0*m)) * (1.0d0/(DL**2))
            else if (ii==jj) then
                ham(ii,jj) =   (1.0d0/(2.0d0*m)) * (2.0d0/(DL**2)) &
                             + (0.5d0*m*omega**2) * (-L/2 + DL*(ii-1))**2
            else
                ham(ii,jj) =    0.0d0
            end if
        end do
    end do
end function init_ham
```

Listing 1. Implementation of the initialization of discretized Hamiltonian tridiagonal matrix.

2.2 Eigenvalues and eigenvectors numerical computation

For the calculation of the eigenvalues and eigenvectors of the discretized Hamiltonian, the Lapack function **dstev** is employed since the matrix we are working with is tridiagonal and this choice optimizes the execution time. Therefore, the subroutine **diag_ham** is built as a wrapper of the Lapack function, as it is showed in Listing 2. Note that the input arguments are changed after the execution of this subroutine, storing then the j^{th} eigenfunction in the j^{th} column of `ham` and the $N+1$ eigenvalues in `eigs`.

```
subroutine diag_ham(ham, eigs)
    ! input arguments
    real(8), dimension(:,:) :: ham
    real(8), dimension(:)   :: eigs

    ! vectors storing diagonal and upper/lower diagonal
    real(8), dimension(size(ham,1))  :: dd
    real(8), dimension(size(dd,1)-1) :: sd

    real(8), allocatable :: work(:)
    integer(4) :: N, lwork, info
    integer(4) :: ii

    N = size(ham, 1)

    ! fill diagonal
    do ii=1,N
        dd(ii) = ham(ii,ii)
    end do
    ! fill upper/lower diagonal
    do ii=1,N-1
        sd(ii) = ham(ii,ii+1)
    end do

    N = size(dd, 1)
    lwork = max(1,2*N-2)

    allocate(work(lwork))
    call dstev('V', N, dd, sd, ham, N, work, info)
    deallocate(work)

    eigs = dd
end subroutine diag_ham
```

Listing 2. Implementation of the wrapper of `dstev`.

2.3 Main program

Lastly, all the functions described before are employed in a test program, which takes multiple command-line input arguments, listed in Table 1, and compiled using the flags `-Wall` and `-ffree-line-length-0`. Given in order the number of discretization intervals N, the length L of the space interval, the number k of eigenfunctions to store and the parameters m and ω, the program:

- initializes the discretized Hamiltonian;
- diagonalizes the ladder through the Lapack function wrapper previously described;
- prints on separate files the first k eigenfunctions points $(x_i, \psi_k(x_i))$ in a two column format;

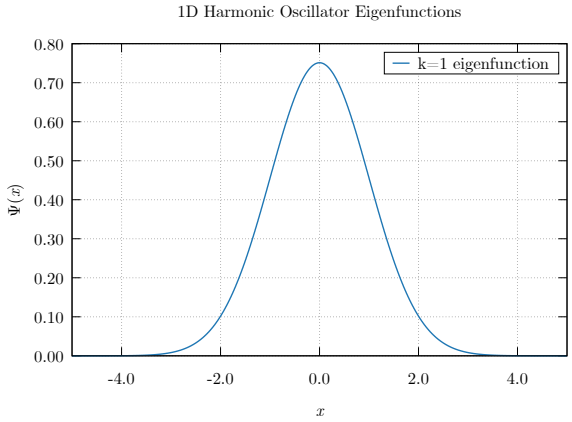
- prints on file the $N+1$ eigenvalues obtained from numerical simulation, the corresponding $N+1$ eigenvalues obtained from theory and the index n associated to the $(n+1)^{\text{th}}$ eigenvalue.

Arg name	Arg number	Arg meaning
`N`	1	Number of discretization intervals
`L`	2	Length of spatial interval of simulation
`K`	3	Number of eigenfunctions to save
`M`	4	Mass parameter of the Hamiltonian
`O`	5	ω parameter of the Hamiltonian

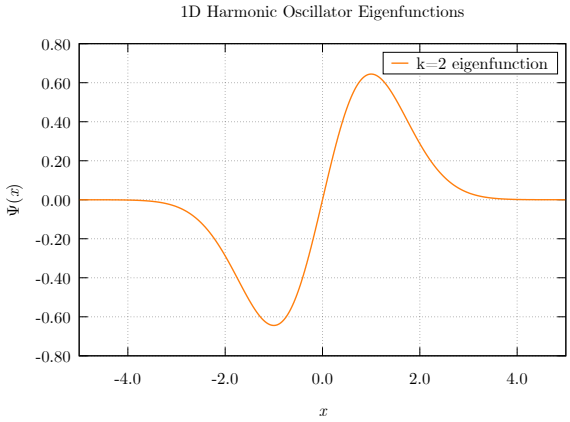
Table 1. Command-line arguments of the main program.

3 Results

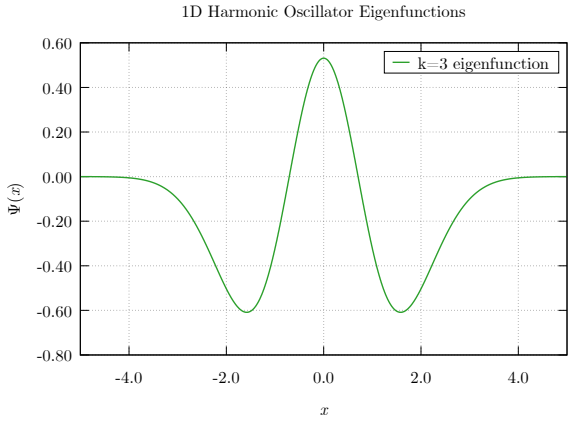
The results obtained from the main program, for $2m = 1$, $\frac{1}{2}m\omega^2 = 1$, $N = 2500$ and $L = 10$, are plotted using Gnuplot. The first 4 eigenfunctions are reported in Figures 1a, 1b, 1c and 1d.



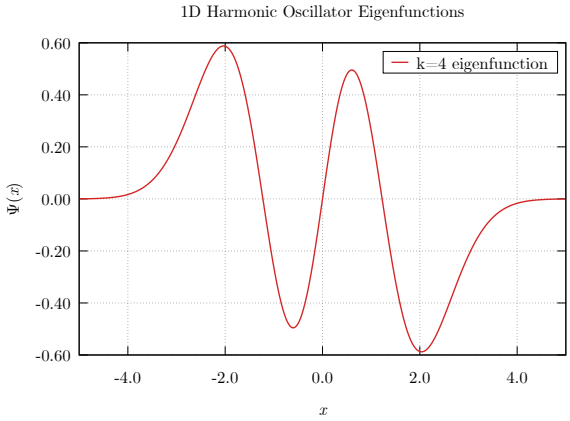
(a) $k = 1$



(b) $k = 2$



(c) $k = 3$



(d) $k = 4$

Figure 1. First $k = 4$ normalized eigenfunctions obtained from numerical simulation.

The first $(N+1)$ numerical eigenvalues, with $N = 2500$, are computed and compared with the theoretical expectation in Eq. 2. The relative error between numerical and theoretical computations is showed in Figure 2, where it is plotted for every E_k, with $0 < k \leq 2500$. We can see how the numerical solution through the finite differences method is approximately correct up to $k \sim 10$, then the $E_{k,\text{num}}$s start to explode.

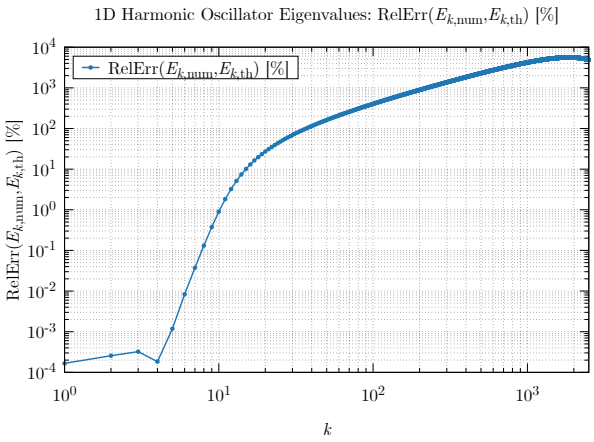


Figure 2. Relative error $\frac{|E_{k,\text{num}} - E_{k,\text{th}}|}{E_{k,\text{th}}}$ for $0 < k \leq 2500$.

4 Self-evaluation

In this work we managed to solve numerically the time independent Schrödinger equation for a monodimensional harmonic oscillator in a limited space interval. This task is accomplished using a finite differences method and Lapack functions for the code implementation of the diagonalization of a tridiagonal matrix, representing the discretized Hamiltonian.

Concerning the priorities for a good scientific software, the code implemented has the following features:

- **correctness**: the numerical solutions for the first k eigenvalues and eigenfunctions, with $k \sim 10$, is in agreement with theory with good precision;
- **numerical stability**: small variations in the input arguments of the main program do not result in catastrophic variations in the output;
- **accurate discretization**: the step size can be set to a sufficiently small size to capture the physics of the problem, but this is possible only for the first $k \sim 10$ eigenvalues, since the method employed has a finite order of precision;
- **flexibility**: the code implementation allows to run the numerical simulation for different values of m, ω parameters, to control the discretization and the size of the space interval in which the simulation is run. The next step to further improve this feature is adding an argument parser;
- **efficiency**: state-of-the-art tools like Lapack functions are employed for the most critical steps of the simulation. However, a deeper analysis on the optimal work parameters of the diagonalization functions should be done.