

# **Numerical method to find the ground state of an Hamiltonian**

**Technical report for fortran numerical project**

Paradis Enzo

Student at the university of Bourgogne Franche-Comté

Master CompuPhys - 1<sup>st</sup> year

**Contents**

**I Introduction** 2

**II Functional requirement of the program** 2

    II.1 Hamiltonians . . . . . 2

**III Internal structure of the program** 4

## I Introduction

In quantum mechanics the eigenstates of an Hamiltonian are really important, especially the ground state. This state is the one that is the most populated and it's the starting point for the study of light-matter interaction. This program goals to find this ground state for any Hamiltonian by partially diagonalise the Hamiltonian. We will try to use the power iterative method and the Lanczos algorithm. This program was written in Fortran 95 under manjaro. There are 3 f95 files. The first one is `main.f95`, which is the main file. If you want to add a new Hamiltonian or choose a method to diagonalise one, it's in this file. The second file is `state.f95`, which contains all the code for the power iterative method (in the subroutine `subroutine state`). You don't have to modify this file. And the third one is `lanczos.f95`, which contains the code for the Lanczos algorithm (in the subroutine `subroutine lanczos`). You don't have to modify this file but this part of the code don't work so you can try to improve it if you want. There is a file named `main` which is the compile form of the program. You can find all these files on [github](#).

## II Functional requirement of the program

In the main file, you can find the `program pool`. In this program there is the creation of Hamiltonians that we want to diagonalize and the called of subroutines for each method. There are already 3 functions for Hamiltonians. There is also an "Hamiltonian" named `test`, which looks like an Hamiltonian for tests. So after creating the Hamiltonian wanted you can use the `subroutine main`, which create initial data and call the `subroutine state`, or the `subroutine lanczos` or both.

### II.1 Hamiltonians

For this program we had created 3 functions for Hamiltonian. There are tables to describe them :

<code>function H_2lvl(omega, phi, delta)</code>		
Description	Input	Output
This function retrieve a Hamiltonian for a 2 level atom interacting with laser fields.	<ul style="list-style-type: none"><li>• <code>omega</code> : the laser amplitude multiplied by the atomic dipolar moment amplitude.</li><li>• <code>phi</code> : the laser phase.</li><li>• <code>delta</code> : the detuning.</li></ul>	This function will retrieve a 2x2 matrix.

Table 1: `function H_2lvl`

After using the `function` `molecular_H` you have to add to it the wanted potential to the kinetic Hamiltonian

<code>function</code> <code>H_3lvl(omega_p, phi_p, delta_p, omega_s, phi_s, delta_s)</code>		
Description	Input	Output
This function retrieve a Hamiltonian for a 3 level atom interacting with laser fields.	<p>The "p" is for pump mode and "s" for Stokes mode.</p> <ul style="list-style-type: none"> <li>• <code>omega_p/omega_s</code> : the laser amplitude multiplied by the atomic dipolar moment amplitude.</li> <li>• <code>phi_p/phi_s</code> : the laser phase.</li> <li>• <code>delta_p/delta_s</code> : the detuning.</li> </ul>	This function will retrieve a 3x3 matrix.

Table 2: `function` `H_2lvl`

<code>function</code> <code>molecular_H(N, L, m)</code>		
Description	Input	Output
This function retrieve a molecular kinetic Hamiltonian.	<ul style="list-style-type: none"> <li>• <code>N</code> : number of particle (size of the Hamiltonian)</li> <li>• <code>L</code> : size of the molecule</li> <li>• <code>m</code> : mass of the molecule</li> </ul>	This function will retrieve a NxN matrix, wich is diagonal.

Table 3: `function` `H_2lvl`

(all of them are commented under the example call). There are three different potatentials.

- Particle in a box (`function` `box_potential(N, L)`)

$$V(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{if } x \in ]0, L[ \\ +\infty & \text{if } x = L \end{cases} \quad (\text{II.1})$$

- Harmonic oscillator (`function` `ho_potential(N, L, k)`)

$$V(x) = \frac{1}{2}k\left(x - \frac{L}{2}\right)^2 \quad (\text{II.2})$$

- Vibration of the  $H_2^+$  molecule (`function` `molecular_potential(N, L, V0, a)`)

$$V(x) = V_0 \left( e^{-2a(x-x_0)} - 2e^{a(x-x_0)} \right) \quad (\text{II.3})$$

### **III Internal structure of the program**