

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 - 1st year

Contents

I	Introduction	2
II	Functional requirement of the program	2
II.1	Hamiltonians	2
III	Internal structure of the program	4
III.1	Description of the diagonalization algorithms	4
III.1.1	The power method algorithm	4
III.1.2	The Lanczos algorithm	5
IV	Appendix	6
IV.1	Proof of the relation for the characteristic polynomial	6

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">• <code>omega</code> : the laser amplitude multiplied by the atomic dipolar moment amplitude.• <code>phi</code> : the laser phase.• <code>delta</code> : the detuning.	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"> • <code>omega_p/omega_s</code> : the laser amplitude multiplied by the atomic dipolar moment amplitude. • <code>phi_p/phi_s</code> : the laser phase. • <code>delta_p/delta_s</code> : the detuning. 	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"> • <code>N</code> : number of particle (size of the Hamiltonian) • <code>L</code> : size of the molecule • <code>m</code> : mass of the molecule 	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

III.1 Description of the diagonalization algorithms

III.1.1 The power method algorithm

The power method algorithm consist to apply a matrix to a vector an infinite number of time. So to find the eigenvector we have :

$$|\phi_0\rangle = H^\infty |\psi\rangle \quad (\text{III.1})$$

With H the matrix we want to diagonalize, $|\phi_0\rangle$ the eigenvector and $|\psi\rangle$ a random vector. So the associated eigenvalue, λ_i , is $\lambda_i = \langle \phi_0 | M | \phi_0 \rangle$.

But we can't apply an infinite number of time a matrix to a vector. So we will transform a little the [Equation III.1](#) and for the numerical calculation add a condition to stop before the infinity. We rewrite the [Equation III.1](#) as :

$$\lim_{k \rightarrow +\infty} \frac{H^k |\psi\rangle}{\|H^k |\psi\rangle\|} = |\phi_0\rangle \quad (\text{III.2})$$

In our case H is a Hamiltonian, so it is a hermitian matrix with a negative spectrum. And we want to find the ground state which is the lowest eigenvalue of H . The lowest eigenvalue is $\lambda_0 = \langle \phi_0 | H | \phi_0 \rangle$. So we will stop the computation when :

$$\|H|\phi_0\rangle - \langle \phi_0 | H | \phi_0 \rangle |\phi_0\rangle\| > \epsilon \quad (\text{III.3})$$

It is when the lowest eigenvalue it is a solution of the eigenequation for the ground state. With that we can write a pseudo-code to find the ground state.

```
take a random vector  $\phi_0$ 
normalize  $\phi_0$ 
 $H \leftarrow H - shift * id$ 
while  $\|H\phi_0 - \langle \phi_0 | H | \phi_0 \rangle \phi_0\| > \epsilon$  and  $k \leq k_{max}$  do
 $\phi_0 \leftarrow H\phi_0$ 
normalize  $\phi_0$ 
 $k \leftarrow k + 1$ 
end while
 $H \leftarrow H + shift * id$ 
 $\lambda_0 \leftarrow \langle \phi_0 | H | \phi_0 \rangle$ 
```

shift is a shift value to ensure a negative spectrum.

By using this method we can also find the first excited state by projecting onto an orthogonal component to avoid non-zero component from ϕ_0 appearing into ϕ_1 . So we can write the following pseudo-code :

```
take a random vector  $\phi_1$ 
 $\phi_1 \leftarrow \phi_1 - \langle \phi_0 | \phi_1 \rangle \phi_0$ 
normalize  $\phi_1$ 
 $H \leftarrow H - shift * id$ 
while  $\|H\phi_1 - \langle \phi_1 | H | \phi_1 \rangle \phi_1\| > \epsilon$  and  $k \leq k_{max}$  do
 $\phi_1 \leftarrow H\phi_1$ 
 $\phi_1 \leftarrow \phi_1 - \langle \phi_0 | \phi_1 \rangle \phi_0$ 
normalize  $\phi_1$ 
 $k \leftarrow k + 1$ 
end while
 $H \leftarrow H + shift * id$ 
 $\lambda_1 \leftarrow \langle \phi_1 | H | \phi_1 \rangle$ 
```

After that ϕ_0 is the ground state and ϕ_1 is the first excited state, λ_0 and λ_1 are the associated eigenvalues.

III.1.2 The Lanczos algorithm

The Lanczos algorithm works onto tridiagonal matrices so we project the Hamiltonian onto subspaces generated by the power iteration method. These subspaces are called Krylov subspaces :

$$\mathcal{K}^n(H, \psi_0) = \text{Lin}(\psi_0, H\psi_0, H^2\psi_0, \dots, H^{n-1}\psi_0)$$

We define the projection of H into the Krylov subspace in the basis \mathcal{B}^n (an orthonormal basis of $\mathcal{K}^n(H, \psi_0)$), as :

$$H_{B^n} = \begin{pmatrix} \alpha_0 & \tilde{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 & \ddots & \\ & \ddots & \ddots & \tilde{\beta}_{n-1} \\ 0 & & \beta_{n-1} & \alpha_{n-1} \end{pmatrix} \quad (\text{III.4})$$

With $\alpha_i = \langle \psi_i | H | \psi_i \rangle$ and $\beta_i = \langle \psi_{i+1} | H | \psi_i \rangle$. The orthonormal basis \mathcal{B}^n is defined, using a modified Gram-Schmidt orthonormalization algorithm, as :

$$\begin{aligned} \psi_1 &= \frac{\psi'_1}{\|\psi'_1\|} & \text{with } \psi'_1 &= H\psi_0 - \langle \psi_0 | H | \psi_0 \rangle \psi_0 \\ \psi_i &= \frac{\psi'_i}{\|\psi'_i\|} & \text{with } \psi'_i &= H\psi_{i-1} - \sum_{k=0}^{i-1} \langle \psi_k | H | \psi_{i-1} \rangle \psi_k \end{aligned} \quad (\text{III.5})$$

After the projection we just have to use a direct diagonalization of H_{B^n} . For a direct diagonalization we need to compute the characteristic polynomial of H_{B^n} . To compute this polynomial we will use the following recurrence relation :

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= \alpha_0 - \lambda \\ p_k(\lambda) &= (\alpha_{k-1} - \lambda)p_{k-1}(\lambda) - |\beta_{k-1}|^2 p_{k-2}(\lambda) \quad \forall k \in \{2, \dots, n\} \end{aligned} \quad (\text{III.6})$$

You can find the proof of this relation in [subsection IV.1](#). To find the lowest eigenvalue we will use a dichotomy algorithm that minimizes the number of eigenvalues in an interval. So the goal is to reduce this interval by reducing the higher value in the interval and tighten it around the eigenvalue. There is the pseudo-code of this dichotomy :

```

b ← 0
while ω(b) - ω(a) ≠ 1 do b ← (a + b)/2
while (b - a) > ε do
if ω((b + a)/2) - ω(a) = 1
b ← (a + b)/2
else
a ← (a + b)/2
end if
end while
λ0(n) ← (a + b)/2

```

The function $\omega(\lambda)$ is the number of sign changes in the sequence $(p_k(\lambda))_{k \in \{0, \dots, n\}}$. The number of sign changes give the number of eigenvalues between 0 and λ , because λ is an eigenvalue if one element of the sequence $(p_k(\lambda))_{k \in \{0, \dots, n\}}$ is null, so when there is a sign change between two consecutive elements of this sequence that means that there is an eigenvalue between them. So $\omega(b) - \omega(a)$ give the number of eigenvalues into $[a, b]$.

So the previous algorithm try to find the eigenvalue by tightening $[a, b]$ around it and keep only one eigenvalue. b is null because we want a negative eigenvalue because we diagonalize a Hamiltonian. So a as to be enough small.

IV Appendix

IV.1 Proof of the relation for the characteristic polynomial

This proof will be done by induction.

We have :

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= \alpha_0 - \lambda_0 \end{aligned} \quad (IV.1)$$

And we want to prove the statement, $S(k)$:

$$p_k(\lambda) = (\alpha_{k-1} - \lambda)p_{k-1}(\lambda) - |\beta_{k-1}|^2 p_{k-2}(\lambda) \quad \forall k \in \{2, \dots, n\} \quad (IV.2)$$

Base case :

For $k = 2$:

$$\begin{aligned} p_2(\lambda) &= \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 \\ \beta_1 & \alpha_1 - \lambda \end{vmatrix} = (\alpha_0 - \lambda)(\alpha_1 - \lambda) - |\beta_{k-2}|^2 \\ &= (\alpha_0 - \lambda)(\alpha_1 - \lambda) - |\beta_{k-2}|^2 \times 1 \\ &= (\alpha_0 - \lambda)p_1(\lambda) - |\beta_{k-2}|^2 p_0(\lambda) \end{aligned} \quad (IV.3)$$

So the statement is true for $k = 2$.

Inductive step :

We suppose that, for $k > 2$, $S(k)$ and $S(k - 1)$ are true.

$$\begin{aligned} p_{k+1}(\lambda) &= \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 - \lambda & \ddots & \\ & \ddots & \ddots & \bar{\beta}_k \\ 0 & & \beta_k & \alpha_k - \lambda \end{vmatrix} \\ &= (-1)^{2k+1} \beta_k \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 - \lambda & \ddots & \\ & \ddots & \ddots & \bar{\beta}_k \\ 0 & & \beta_{k-1} & \bar{\beta}_k \end{vmatrix} + (-1)^{2k+2} (\alpha_k - \lambda) \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 - \lambda & \ddots & \\ & \ddots & \ddots & \bar{\beta}_{k-1} \\ 0 & & \beta_{k-1} & \alpha_{k-1} - \lambda \end{vmatrix} \\ &= (-1)^{4k+1} |\beta_k|^2 \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 - \lambda & \ddots & \\ & \ddots & \ddots & \bar{\beta}_{k-2} \\ 0 & & \beta_{k-2} & \alpha_{k-2} - \lambda \end{vmatrix} + (-1)^{2k+2} (\alpha_k - \lambda) \begin{vmatrix} \alpha_0 - \lambda & \bar{\beta}_1 & & 0 \\ \beta_1 & \alpha_1 - \lambda & \ddots & \\ & \ddots & \ddots & \bar{\beta}_{k-1} \\ 0 & & \beta_{k-1} & \alpha_{k-1} - \lambda \end{vmatrix} \\ &= (\alpha_k - \lambda)p_k(\lambda) - |\beta_k|^2 p_{k-1}(\lambda) \end{aligned} \quad (IV.4)$$

That is, the statement $S(k + 1)$ also holds true, establishing the inductive step.

Conclusion

Since both the base case and the inductive step have been proved as true, by mathematical induction the statement $S(k)$ holds for every natural number k .