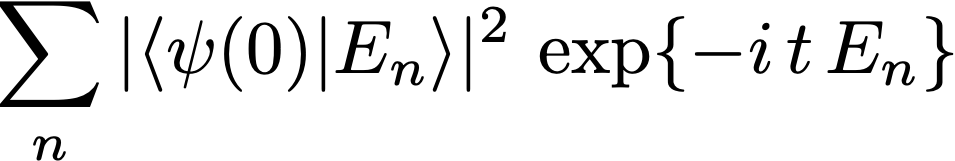
Teaching wave mechanics with Matlab

To avoid any misunderstanding, let’s make it clear that this is not a place where one can learn about professional software devoted to a numerical approach in atomic or molecular physics. For this aim you should look elsewhere for implementations of quantum MonteCarlo, like *Car-Parrinello*. Here instead you will find ideas and techniques useful for teaching elementary wave mechanics to undergraduates. This doesn’t mean that our methods cannot lead to original research, they actually originated from research activity, specifically looking for the impact of background geometry on quantum spectra[[1]](#footnote-2), but they are mostly useful for pedagogical applications. You’ll find here a basic technique, essentially Feynman Path integral, which allows to study the evolution of wave packets according to Schroedinger’s equation, in any given energy potential and display the effect of diffraction, interference, energy barrier penetration (tunnel effect). The calculation of energy spectra for charged particles in external magnetic fields, the nice phenomenon of Landau Levels and Quantum Hall Effect can be approached with our techniques.

The basic idea, known as “*splitting method*”or “*Leapfrog algorithm”,* goes back to Isaac Newton! His proof of the Law of Areas, aka angular momentum conservation, introduces a trick: imagine that the planet evolves inertially for an infinitesimal time as a free particle - hence no attraction from the Sun, hence for another interval the position stays fixed but momentum gets an impulse , where F is directed towards the Sun according to His Law. If the interval is small enough the result is equivalent to a correct solution of the equation of motion. See Newton’s Principia[[2]](#footnote-3)! Now, the same idea can be exported to quantum mechanics, This translates to the well known Trotter’s formula (originally invented by Sophus Lie in the context of continuous groups), namely the quantum evolution given by the unitary operator can be approximated with arbitrary accuracy choosing an infinitesimal interval of time *t/n* and computing . The advantage is twofold, with respect to other numerical recipes: each exponential can be computed *exactly (i.e. to machine accuracy)* and the result is by construction a *unitary operator,* hence there is no loss of probability along the way. Energy is conserved only approximately, namely *n*  , but this can be improved at the expense of speed. The crucial ingredient which allows to compute very efficiently the first exponential is represented by the fast Fourier transform. We go to Fourier space by Fourier, then the free evolution driven by is simply a multiplication by a phase factor and the inverse Fourier takes us back to position space, where the exponential is also a pure phase factor. The idea is due to M.D. Feit, J.A.Fleck and A. Steiger[[3]](#footnote-4).

On your personal computer the method can be implemented using Matlab to study quantum dynamics for scalar particles in one-two and three dimensions. You will find Matlab codes in this repository for a suite of problems, but the good idea is to grasp the essential idea and to try to build yourselves the program, why, you could do better than us!

The technique can also be used to estimate the energy spectrum: if you compute the time evolution then the spectrum is encoded in the wave function. You can extract the spectrum by Fourier analysing the overlap which is given by

This was the proposal in the original reference (3). At present this is not the best choice, since we can compute the spectrum by other “sparse” methods much more efficient (Arnoldi method), as you may find elsewhere in this blog.

## W

## e suggest to start with two examples - *“doublewell.m”* and “*wms2.m*”. The first program displays the evolution of a quantum particle in a potential exhibiting two classical vacua, namely . Let the particle be located as a Gaussian packet around *x=a* at *t=0;* quantum evolution makes it evolve oscillating around *x=a* and we see its leaking to the left region around *x* *= -a even if its energy is less than* due to the tunnel effect. This is a simplified model of what really happens in the ammonia molecule .

Now let’s look at the code in detail. You may conveniently keep the code under your sight with an editor, here we give some essential comments. The function interface is very concise…

function setup=doublewell(setup)

The interaction with the user is through a setup file. If you start from scratch this can be empty and a default will be provided; it is then saved with your choices in order to be able to restart with the same initial setup or a variation thereof. The setub file contains all physical parameters *mass, momentum, width of the wave packet,*  and other “technical” parameters which define the computing environment *box size, grid size, evolution time, etc.*. Then come the definition of the position grid (here 1-dimensional)

dx = 2\*L/N

x = -L\*(1-1/N) : dx : L\*(1-1/N);

and the momentum space grid

k = fftshift(pi\*(-N/2:(N/2-1))/L);

Then we define the potential energy

V = V0 \* (1-(x/omega).^2).^2;

The spitting algorithm requires the two phase factors and , thus

U = exp(-i\*tau\*V); U2 = exp(-0.5i\*tau\*V);

W = exp(-0.5i\*k2/mass\*tau);

U2, a half step evolution, is used to build a symmetric algorithm, as we can see in the full listing. The initial wave packet is built as a normalized Gaussian, centered around one minimum of the potential

psi = exp(-(x-x0).^2/4/sigma+i\*mom\*x)/2\*pi\*sigma).^(1/4);

The evolution is given by a loop over the number of time steps, each consisting of the application of the phase factors to the wave function alternatevily in position and in momentum space, just one line

psi = U.\*ifft(fft(psi).\*W);

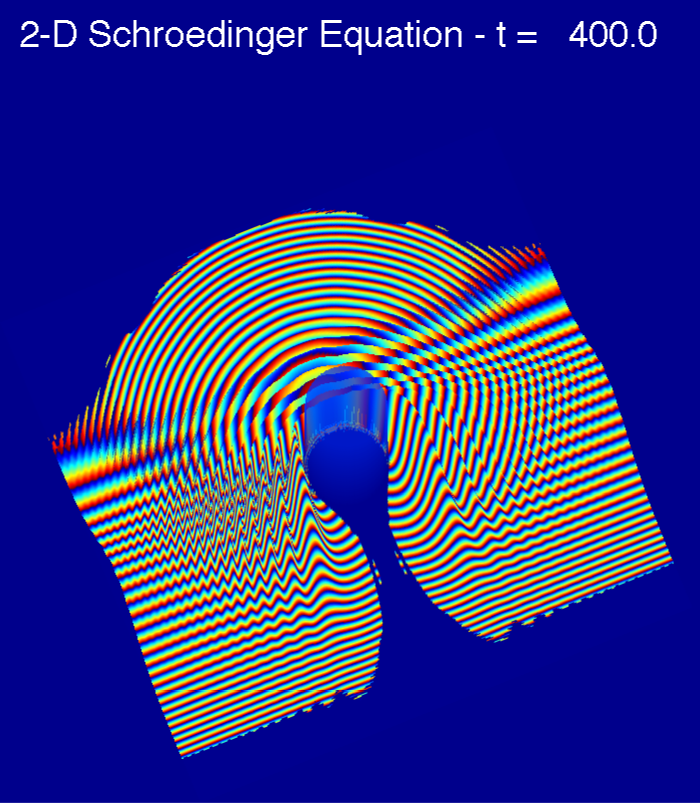
but it may be clearer by unfolding it to

phi = fft(psi); % *go to momentum space*

phi = W.\*phi; % *apply W=exp(-i dt p^2)*

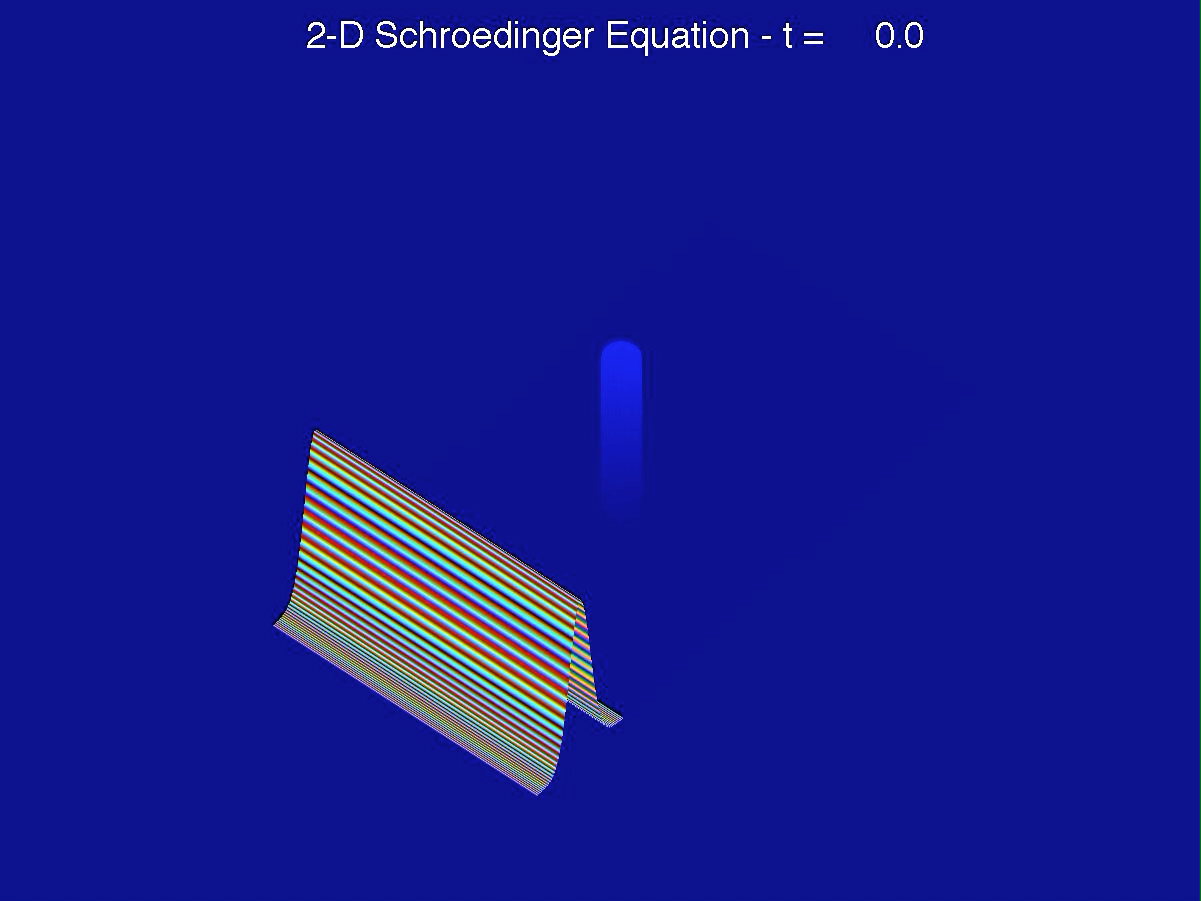
psi = ifft(phi);% *back to position space*

psi = U.\*psi; % *apply U = exp(-i dt V(x))*

In this loop we spend most of the computing time. The rest of the code is devoted to the graphical display of the probability density . An option lets us to save the snapshots in order to produce a stand alone video clip. Let’s skip the details.

## T

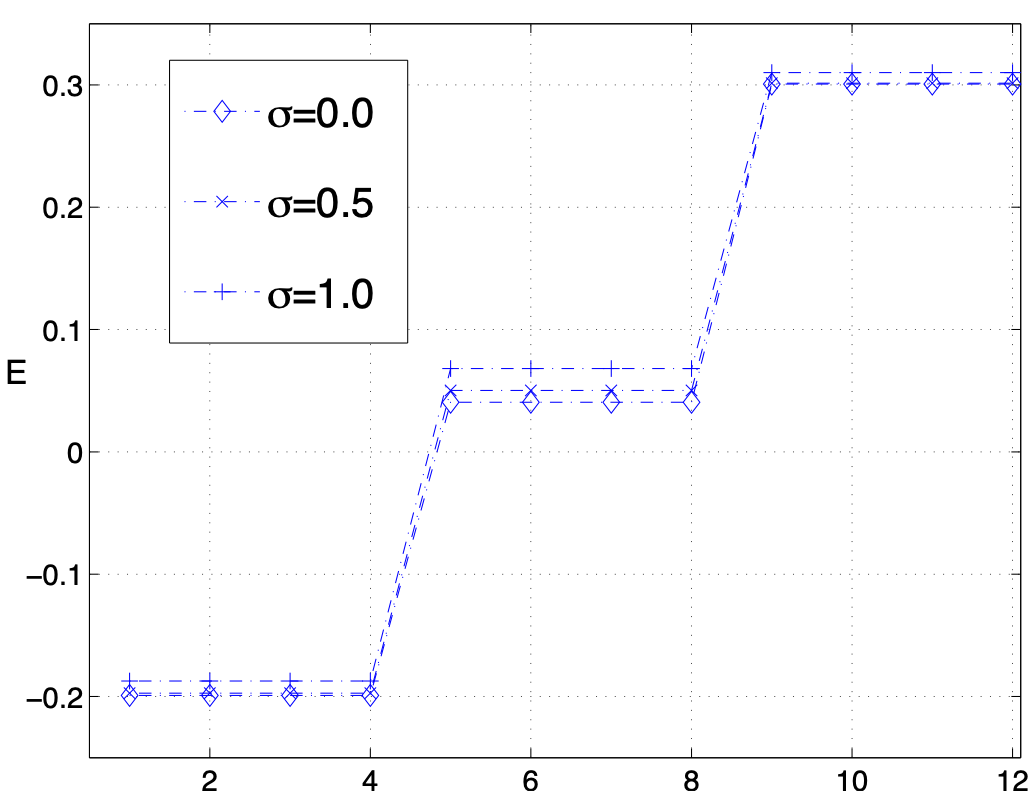
## he second example consists in the evolution of a wave packet in two-dimensions and it can be used to display the phenomena of diffraction and interference typical of wave mechanics. In the multidimensional case we have to work a bit more in defining the position grid and momentum grid. This can be easily done using Matlab’s built-in routines *meshgrid* or *ndgrid.* The core of the program is essentially identical, given we employ the two-dimensional version of the fast Fourier tranform *fft2* and *ifft2.* The display is organised to give both the probability density and the phase: this latter is rendered by a color code which is linked to the phase. In this way one can check the difference between *phase velocity* and *group velocity!* Here we have a plane wave packet scattering around a potential barrier and we can see the back scattered circular wave and the main packet continuing in the original direction.

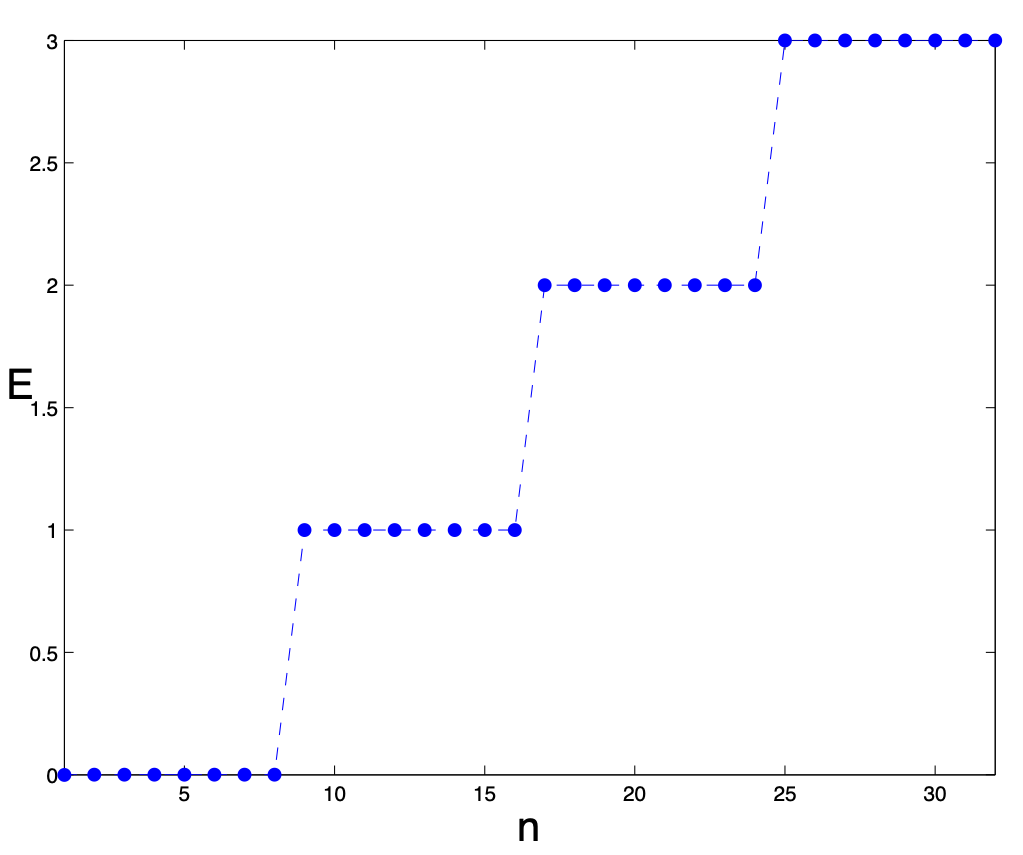
A video clip with about this problem can be found as “*thebigone.mov”* in the file depository.

Landau levels

N

ow we introduce another elementary application of the splitting technique which has a distinguished character: Landau Levels on the computer. The subject has been for decades a topic in Mathematical Physics, since the 1930 paper by Landau, but in recent years it has been reconsidered or its phenomenological relevance[[4]](#footnote-5). The problem consists in computing the energy spectrum for a charged particle in two dimensions subject to a transverse uniform magnetic field. The problem allows for a complete analytic solution (Landau), but the availability of a precise numerical computation make it possible to estimate the effect of various perturbation, which is not always possible analytically. Our method consists in representing momenta using the trick to go in momentum space via *fft,* however this implies that we are working with periodic boundary conditions hence the particle lives on a torus*.* If the size of the torus is large enough this has little impact on the physics of the problem, but the interesting features of the problem manifest themselves in the opposite case, when we insist on having to do with a particle on a torus and a transversal magnetic field; since a strictly solenoidal field must have zero flux this means that we are actually studying a field generated by a suitable magnetic charge and this must be in accord with Dirac’s quantization condition - hence the flux is quantized. The energy levels turn out to have a degeneracy exactly given by the quantized flux! We can then experiment what happens in presence of various perturbations. Details can be found in the two references; see also Ref.([[5]](#footnote-6)). Matlab code can be found in the blog repository under the name *LL2.m.*

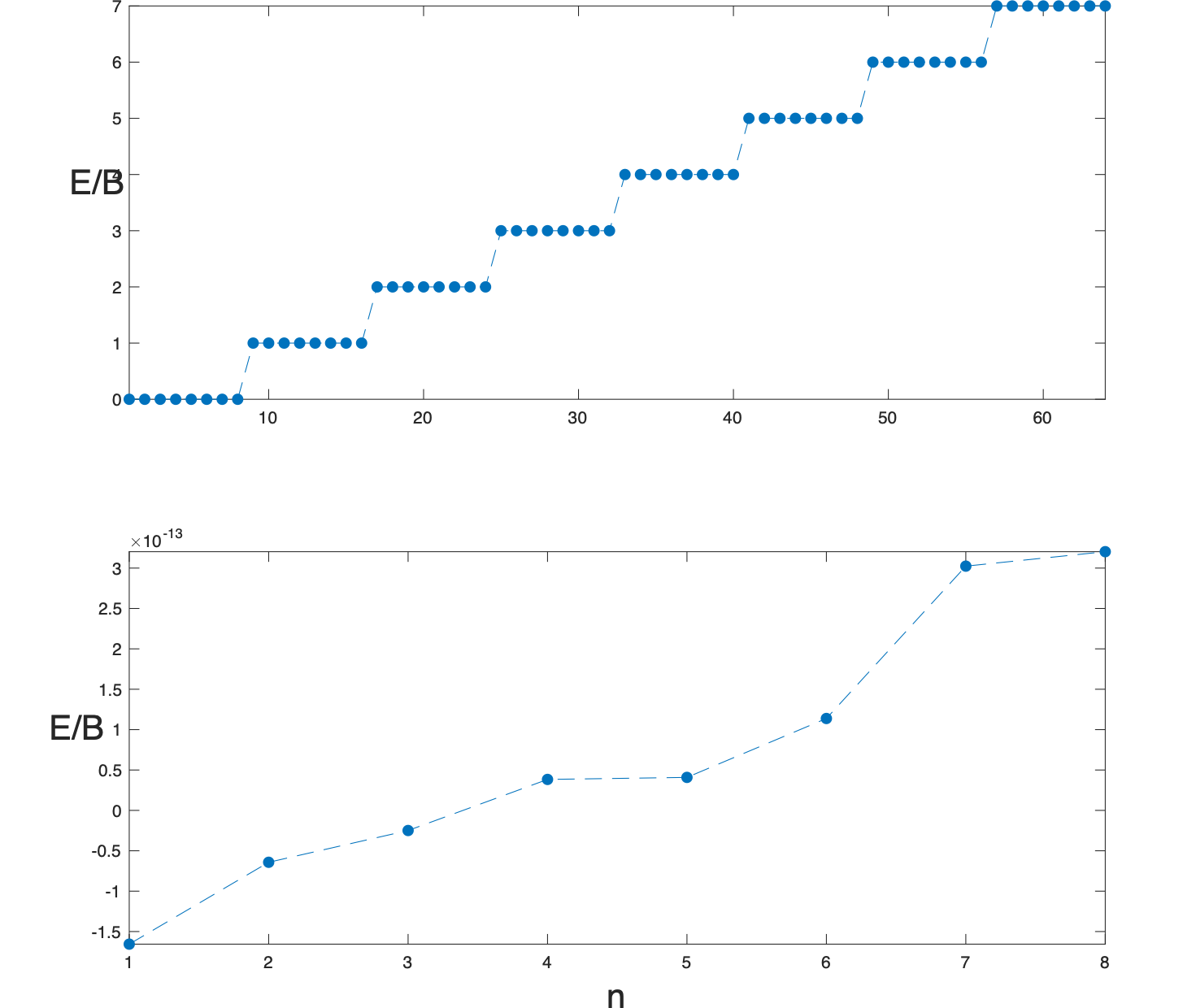
*An example of Landau spectrum with monopole charge 8.*

The previous plot shows that u*nder a periodic perturbation the Landau levels are rather stable,* which is important since in real materials impurities are always present*.* This appears to be hard to prove analytically.

The essential features of the program are the following:

1. we do not use a discretisation to represent the momentum operators, rather we resort to the spectral representation based on the fast Fourier transform.
2. we take into account that the wave function for a charged particle on a compact manifold (here the torus) subject to a transversal magnetic field is not just a periodic function on the torus, rather it is a section of the line bundle built according to the rules of quantum gauge theories. This obliges us to introduce a correction with a phase factor *before* taking the Fourier transform. Details can be found in the first references.
3. The spectrum is computed using the matlab routine ***eigs*** which implements the *Krylov-Schur* algorithm[[6]](#footnote-7): it allows to define the Hamiltonian operator through a subroutine instead of giving explicitly its matrix representation. The accuracy of the method is remarkable. See the following plot where the degeneracy of the levels is reproduced to the required accuracy .

The relevant code, to be found in the archive, is called *LandauMathieu.m*.

**

1. C. Destri, P. Maraner, and E. Onofri, *Nuovo Cimento A*, **107**, 237 (1994). P.Maraner, ,E.Onofri and [G.P. Tecchiolli](https://inspirehep.net/literature?q=a%20G.Tecchiolli.1), *Spectral methods in computational quantum mechanics, J.Comput.Appl.Math., 37, (1991) 209-219.*  [↑](#footnote-ref-2)
2. For instance on gallica.bnf.fr [↑](#footnote-ref-3)
3. M.D. Feit, J.A. Fleck Jr and A. Steiger, J. Comput. Phys. 47 (1982) 412-433. [↑](#footnote-ref-4)
4. Int.J.Theoret.Phys. 40(2) 537, 2001; Int.J.ModernPhys.C, 2008. [↑](#footnote-ref-5)
5. P.Maraner, G.P.Tecchiolli and E.O., *Spectral methods in computational quantum mechanics,* J. Comput. Appl. Math., 37 (1991). [↑](#footnote-ref-6)
6. Stewart, G.W. "*A Krylov-Schur Algorithm for Large Eigenproblems.*" SIAM Journal of Matrix Analysis and Applications. Vol. 23, Issue 3, 2001, pp. 601–614. [↑](#footnote-ref-7)