LAB II
Quantum wave packet dynamics

Jakub Tworzydło

Institute of Theoretical Physics
Jakub.Tworzydlo@fuw.edu.pl

7/03 and 8/03/2023 ul. Pasteura 5, Warszawa

Plan

1. Quantum evolution

2. Phase space portraits

Plan

1. Quantum evolution

2. Phase space portraits

Our task 1

We first need to implement the basic properties of wave function.

The position is defined on an interval $x \in [0, 2\pi)$ with the descrete values $x_n = \frac{2\pi n}{M}$, $n = 0, \dots, M-1$ (let $M = 100$ and $\hbar = 2 * \pi / M$).

- Write a Python function, which prepares a normalized Gaussian wave packet $\psi_G(x_n)$ centered at (x_0, p_0).
- Plot the probability amplitude $|\psi_G|^2$, mark the value x_0.
- Perform the fast Fourier transform (FFT) on the vector $\psi_G(x_n)$. Plot the resulting wave function amplitude, check normalisation, mark the value p_0.

Note that we can represent the wave function as a discrete vector $w_n = \psi_G(x_n)$. Implement this vector as a numpy array, use `fft` from `scipy`.

Hints for task 1

```
n = np.arange(M)              # numpy vector of indexes
                              # n = 0,..., M-1
x = 2*pi*n/M

psiG = np.zeros(M, dtype=complex)      # complex vector

psiG = np.exp( 1j*p0*x/hbar ) * ...    # element-wise
                                       # x -- vector

psiG = psiG/np.linalg.norm(psiG)       # normalization
```

One step evolution

We calculate the quantum dynamics of our standard map.

Let's look at some technicalities. The phase vectors

$$V_n = \exp\left(-\frac{i}{\hbar} K \cos x_n\right), \; P_m = \exp\left(-\frac{i}{2\hbar} p_m^2\right)$$

can be easily implemented with NumPy arrays `V` and `P` respectively.

One step evolution (from the lecture) in a simplified notation reads

$$\bar{\psi}_{n'} = \frac{1}{M} \sum_{m,n} e^{i\frac{2\pi}{M} n' m} P_m e^{-i\frac{2\pi}{M} mn} V_n \psi_n.$$

To implement this evolution we need two element-wise vector multiplications: first one by `V`, then next one by `P`. After the first multiplication we perform `fft` and after the next one the inverse `ifft`. Check the documentation to adjust the proper normalization.

Our task 2

- Implement one step of our evolution.
- Test it: start with a Gaussian wave packet and evolve it for a few initial steps of free evolution ($K = 0$). Plot the wave function amplitude $|\psi(x_n)|^2$ at every step, use some small initial p_0.
- Plot the same few step evolution, but now e.g. for $K = 1.1$, take $M = 1000$. Mark the position of a classical particle, following the standard map. Use the initial (x_0, p_0) in the vicinity of a stability island (form the previous Lab I).

Hints for task 2

This command allows to make a sequence of snapshots labeled by the time step `t`:

```
plt.savefig("wave_packet" + str(t).zfill(3) + ".png")
plt.close()
```

Animation can be produced with a command (in a terminal)

```
convert -delay 20 wave_packet*png animation.gif
```

or

```
ffmpeg -f image2 -framerate 10
               -i "wave_packet\%03d.png" animation.avi
```

Animation in Colab –> last slide

Extra task: this should be fun!

We want to reproduce some quantum phase space portraits.

- Calculate the Husimi distribution $Q(x, p)$ and plot the amplitude $|Q|$ using a color map; test by plotting Q-distribution for an initial Gaussian wave packet.
- Plot the Husimi distribution $|Q|$ of wave function for a number of steps (up to $10 - 20$). Present the sequences for different $K = 0.6, 2.1, 5.2$, start with the initial Gaussian at $(x_0, p_0) = (1.5, 0.3)$.
- *Idea: mark also the position of classical particle. Find and illustrate a case when the quantum wave packet follows (for some time) the classical trajectory.*

Hints

Let:

`Qdist = np.zeros((M,M), dtype=complex)`

represents $Q(x_0, p_0)$.

`psiG` – Gaussian wave packet centered at the postition `x0=2*pi*n0/M` with momentum `p0=0`.

Calculate for `n0=0,...,M-1` (updating `psiG` inside the loop)

```
Qdist[:,n0] = fft(np.conj(psiG)*psi)
```

Color plot

```
n = np.arange(M)
X, Y = np.meshgrid(n,n)  # some convenient grid
plt.pcolor( X,Y, np.abs(Qdist), cmap='hot')
plt.colorbar()               # color scale
```

Animations in Colab

```
# mount drive
from google.colab import drive
from google.colab import files  # (optional)
drive.mount('/content/gdrive')
images_dir = '/content/gdrive/My Drive/'
```

then we can save the snapshots

```
plt.savefig(images_dir+'wave_packet{:03d}.png'.format(t))
# # (optionally) one can download the resulting plots
# files.download(
#           images_dir+'wave_packet{:03d}.png'.format(t))
```