Computational Physics Exam: Quantum Harmonic Oscillator

Tamilarasan Ketheeswaran 411069*

28 July 2023

Contents

1 Introduction

In this exam, we deal with the simulation of the (1D) quantum harmonic oscillator. First, we find an analytic expression for the averages and variances of the position of a wavepacket in such a potential. After that, we discuss the simulation method used to solve the Schrödinger numerically. There we decompose matrices for faster calculation and make use of the second-order-product formula.

Consequently, we present the results and compare them to the analytical expression. At last we discuss the effects on varying the different parameters of the system. Here, we investigate the different kind of states in a quantum harmonic oscillator.

For the simulation, we use the programming language `Python` and the packages `numpy`[1] and to plot our results `matplotlib`. Moreover, from `numba` we import `njit` and from `multiprocessing` `Pool` for a faster simulation.

*tamilarasan.ketheswaran@rwth-aachen.de

[1]abbreviated as `np`

2 Theoretical Framework

In this section, we govern the theory behind the system. Having a particle in a harmonic potential, we are interested in the time evolution of the particle's mean position $\langle x(t) \rangle$ and variance $\mathrm{Var}(x(t))$. Setting $m = 1$ and $\hbar = 1$, the Schrödinger equation in position space for a harmonic potential is given by

$$i\frac{\partial}{\partial t}\Phi(x,t) = \left(-\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{\Omega^2}{2}x^2\right)\Phi(x,t) \tag{1}$$

with the Hamilton operator given by

$$\hat{H} = \left(\frac{\hat{p}^2}{2} + \frac{\Omega^2}{2}\hat{x}^2\right). \tag{2}$$

Initially, at $t = 0$, the wavepacket is described via

$$\langle x|\Phi(0)\rangle = \Phi(x, t = 0) = \Phi_0 = \left(\frac{1}{\sqrt{\pi\sigma^2}}\right)^{\frac{1}{2}} e^{-\frac{(x-x_0)^2}{2\sigma^2}}, \tag{3}$$

a gaussian wavepacket of width $\frac{\sigma}{\sqrt{2}}$ centered around x_0. We focus ourselves on the calculation of $\langle x(t)\rangle$ and $\langle x^2(t)\rangle$. For that, we make use of the *Ehrenfest Theorem*:

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{O}\rangle = \frac{1}{i}\langle[\hat{O},\hat{H}]\rangle + \left\langle\frac{\partial\hat{O}}{\partial t}\right\rangle. \tag{4}$$

The position operator $\hat{x}$ and momentum operator $\hat{p}$ do not depend on time, such that the derivative on the RHS vanishes. In general, the average of an operator $\langle O\rangle$ can be calculated using

$$\langle O\rangle = \langle\Phi(t)|\hat{O}|\Phi(t)\rangle = \int_{-\infty}^{\infty}\Phi^*(x,t)\langle x|O|\Phi(t)\rangle\,\mathrm{d}x. \tag{5}$$

In the following derivation, we make use of the four commutator relations

$$[\hat{x},\hat{p}] = i, \qquad [\hat{x},\hat{p}^2] = 2i\hat{p}, \qquad [\hat{x}^2,\hat{p}] = 2i\hat{x}, \qquad [\hat{x}^2,\hat{p}^2] = 4i\hat{x}\hat{p} + 2. \tag{6}$$

Also, we come across different integrals, which can be narrowed down to the calculation of the following:

$$\begin{aligned}\int_{-\infty}^{\infty} e^{-x^2}dx &= \sqrt{\pi}\\ \int_{-\infty}^{\infty} xe^{-x^2}dx &= 0\\ \int_{-\infty}^{\infty} x^2e^{-x^2}dx &= \frac{\sqrt{\pi}}{2}.\end{aligned} \tag{7}$$

Starting with the Ehrenfest theorem for both the position and momentum operator, with our definition for $\hat{H}$ from eq.(2), we obtain

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{x}\rangle = \frac{1}{i}\langle[\hat{x},\hat{H}]\rangle = \frac{1}{2i}\langle[\hat{x},\hat{p}^2]\rangle = \langle\hat{p}\rangle \tag{8}$$

and

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{p}\rangle = \frac{1}{i}\langle[\hat{p},\hat{H}]\rangle = \frac{\Omega^2}{2i}\langle[\hat{p},\hat{x}^2]\rangle = -\Omega^2\langle\hat{x}\rangle. \tag{9}$$

Taking another time derivative of $\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{x}\rangle$, we have

$$\frac{\mathrm{d}^2}{\mathrm{dt}^2}\langle\hat{x}\rangle = \frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{p}\rangle \qquad \Rightarrow \qquad \frac{\mathrm{d}^2}{\mathrm{dt}^2}\langle\hat{x}\rangle = -\Omega^2\langle\hat{x}\rangle\,. \tag{10}$$

The second-order differential equation above can be solved with the ansatz

$$\langle x(t)\rangle = +A\cos(\Omega t) + B\sin(\Omega t) \tag{11}$$
$$\text{from eq.(8)} \Rightarrow \langle p(t)\rangle = -A\Omega\sin(\Omega t) + B\Omega\cos(\Omega t). \tag{12}$$

Here, the constants can be solved using the initial wave packet in eq.(5) and the integrals in eq.(7) such that

$$A = \langle x(t)\rangle\Big|_{t=0} = \int_{-\infty}^{\infty} x|\Phi_0|^2\mathrm{d}x = x_0 \qquad \text{and}$$
$$B\Omega = \langle p(t)\rangle\Big|_{t=0} = \int_{-\infty}^{\infty}\Phi_0^*\frac{1}{i}\frac{\partial}{\partial x}\Phi_0\mathrm{d}x = 0.$$

Therefore

$$\langle x(t)\rangle = x_0\cos(\Omega t). \tag{13}$$

Now, we dedicate ourselves to the calculation of $\langle x^2(t)\rangle$. Again, from the Ehrenfest theorem in eq.(4) we have

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{x}^2\rangle = \frac{1}{i}\langle[\hat{x}^2,\hat{H}]\rangle = \frac{1}{2i}\langle[\hat{x}^2,\hat{p}^2]\rangle = \langle\hat{x}\hat{p}+\hat{p}\hat{x}\rangle = 2\langle\hat{x}\hat{p}\rangle - i \tag{14}$$
$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{x}\hat{p}\rangle = \frac{1}{i}\langle[\hat{x}\hat{p},\hat{H}]\rangle = \frac{1}{i}\langle\hat{x}[\hat{p},\hat{H}] + [\hat{x},\hat{H}]\hat{p}\rangle = \langle\hat{p}^2\rangle - \Omega^2\langle\hat{x}^2\rangle\,. \tag{15}$$

To obtain a term for $\langle\hat{p}^2\rangle$, we use

$$\text{from eq.(2)} \Rightarrow \langle\hat{H}\rangle = \frac{\langle\hat{p}^2\rangle}{2} + \frac{\Omega^2}{2}\langle\hat{x}^2\rangle$$
$$\Leftrightarrow \langle\hat{p}^2\rangle = 2\langle\hat{H}\rangle - \Omega^2\langle\hat{x}^2\rangle\,,$$

such that eq.(15) rearranges to

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{x}\hat{p}\rangle = 2\langle\hat{H}\rangle - 2\Omega^2\langle\hat{x}^2\rangle\,. \tag{16}$$

After taking another time derivative of the term in eq.(14), we insert eq.(16), such that

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{dt}^2}\langle\hat{x}^2\rangle &= 4\langle\hat{H}\rangle - 4\Omega^2\langle\hat{x}^2\rangle\\ &= 4\Omega^2\left(\frac{\langle\hat{H}\rangle}{\Omega^2} - \langle\hat{x}^2\rangle\right).\end{aligned} \tag{17}$$

Using the Ehrenfest theorem for the Hamilton operator H, we have

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle\hat{H}\rangle = \frac{1}{i}\langle[\hat{H},\hat{H}]\rangle + \left\langle\frac{\partial\hat{H}}{\partial t}\right\rangle = 0. \tag{18}$$

With the time derivative of $\langle H(t)\rangle$ being zero, we deduct the total conservation of the energy of the system, which is why $\langle\hat{H}\rangle = \langle H(t=0)\rangle$. Hence, eq.(17) equals a second-order differential equation with a constant for which we assume the solution to be[2]

$$\langle x^2(t)\rangle = A\cos(2\Omega t) + B\sin(2\Omega t) + C. \tag{19}$$

The factor 2Ω comes from the $4\Omega^2 = (2\Omega)^2$ in eq.(17). Inserting our ansatz into eq.(17) and keeping in mind the conservation of energy, we have

$$\begin{aligned} C &= \frac{\langle H(t=0)\rangle}{\Omega^2} \\ &= \frac{1}{2\Omega^2}\left(\langle p^2(0)\rangle + \Omega^2\langle x^2(0)\rangle\right). \end{aligned} \tag{20}$$

Again, for the calculation of the constants A, B and C, we need to calculate the following initial values according to eq.(5)

$$\langle x^2(t)\rangle|_{t=0} = \int_{-\infty}^{\infty} x^2|\Phi_0|^2\mathrm{d}x \qquad = \frac{1}{2}\left(\sigma^2 + 2x_0^2\right) \tag{21}$$

$$\langle p^2(t)\rangle|_{t=0} = \int_{-\infty}^{\infty} \Phi_0^*\frac{1}{i^2}\frac{\partial^2}{\partial x^2}\Phi_0\mathrm{d}x = \frac{1}{2\sigma^2} \tag{22}$$

$$\langle (xp)(t)\rangle|_{t=0} = \int_{-\infty}^{\infty} x\Phi_0^*\frac{1}{i}\frac{\partial}{\partial x}\Phi_0\mathrm{d}x = \frac{i}{2}, \tag{23}$$

where we have used the integrals from eq.(7). With eq.(23), eq.(20) rewrites as

$$C = \frac{1}{4\Omega^2\sigma^2} + \frac{1}{4}\left(\sigma^2 + 2x_0^2\right). \tag{24}$$

With the initial condition for $\langle x^2(t)\rangle|_{t=0}$, inserted into eq.(19), we have

$$\begin{aligned} \langle x^2(t)\rangle|_{t=0} = \frac{1}{4\Omega^2\sigma^2} + \frac{1}{4}\left(\sigma^2 + 2x_0^2\right) + A &= \frac{1}{2}\left(\sigma^2 + 2x_0^2\right) \\ A = -\frac{1}{4\Omega^2\sigma^2} + \frac{1}{4}\left(\sigma^2 + 2x_0^2\right)&. \end{aligned} \tag{25}$$

Lastly, eq.(23) can be inserted into eq.(14), such that

$$\frac{\mathrm{d}}{\mathrm{dt}}\langle x^2\rangle\Big|_{t=0} = 2\langle xp\rangle\Big|_{t=0} - i = 0.$$

And thus

$$0 = 2\Omega B \qquad \Rightarrow \qquad B = 0. \tag{26}$$

Summarizing the results for A, B and C, we have

$$\langle x^2(t)\rangle = \left(\frac{1}{4\Omega^2\sigma^2} + \frac{1}{4}\left(\sigma^2 + 2x_0^2\right)\right) + \left(-\frac{1}{4\Omega^2\sigma^2} + \frac{1}{4}\left(\sigma^2 + 2x_0^2\right)\right)\cos(2\Omega t),$$

or rearranging the terms and using trigonometric identities, we are left with

$$\langle x^2(t)\rangle = \frac{1}{2\Omega^2\sigma^2}\sin^2(\Omega t) + \frac{1}{2}\left(\sigma^2 + 2x_0^2\right)\cos^2(\Omega t). \tag{27}$$

From the latter expression, and from eq.(13), the variance can be calculated:

$$\mathrm{Var}(x) = \langle x^2(t)\rangle - \langle x(t)\rangle^2 \tag{28}$$

$$= \frac{1}{2\Omega^2\sigma^2}\sin^2(\Omega t) + \frac{1}{2}\sigma^2\cos^2(\Omega t). \tag{29}$$

[2]note that these constants A and B are not the same as in eq.(11)

3 Simulation model and method

As seen in the previous section, initially there is a Gaussian wavepacket (see eq.(3)) centered at x_0. To solve eq.(1) efficiently, we constrain the boundary of the system to the left and right such that $-15 \leq x \leq 15$. Within these boundaries, the position can be discretized according to

$$x_i = -\Delta \left(\frac{L-1}{2} - i \right), \quad \text{with} \quad i = 0, 1, 2, ..., L-1. \tag{30}$$

In that case, we have L discrete grid points and the spatial resolution is given by Δ. In `python`, we implement this discretization of position using

```
x = np.linspace(-15,15,L).
```
[3]

With this discretization, the total wave function at any time t is given by

$$\Phi(t) = \begin{pmatrix} \Phi_0(t) \\ \Phi_1(t) \\ \vdots \\ \Phi_{L-1}(t) \end{pmatrix}, \tag{31}$$

where $\Phi_i(t)$ is the value of the wave function at position x_i and time t, $\Phi(x_i, t)$. Regarding the implementation into `python`, we use the function for the initial state eq.(3) and calculate all the values for the positions defined in `x`. By doing so, we are left with an array of length L containing the values of the wavefunctions. In addition to the discretization of x, the time is also discretized according to

$$t = n \cdot \tau \tag{32}$$

with n ranging from 0 to m. The idea of the algorithm, we discuss now, is to update the array $\Phi(t)$, in each iteration corresponding to a time evolution of τ. This has the advantage to that we don't have to store the wavefunction at all times individually[4].

With the discretization of space, for the right-hand-side (RHS) of eq.(1) it follows

$$\frac{\partial^2}{\partial x^2}\Phi(x,t) \longrightarrow \frac{\Phi(x_i+\Delta,t) - 2\Phi(x_i,t) + \Phi(x_i-\Delta,t)}{\Delta^2} \tag{33}$$

and

$$V(x)\Phi(x,t) \longrightarrow V(x_i)\Phi(x_i,t) \quad \text{with} \quad V(x) = \frac{\Omega^2}{2}x^2. \tag{34}$$

[3]this gives us an array with each point that we have discretized in position

[4]Saving the wave function at each time step results in an enormous runtime and use of computing resources.

Employing the discretizations into the RHS of the Schrödinger equation and using eq.(31), the differential equation can be written in matrix notation as

$$i\frac{\partial}{\partial t}\begin{pmatrix} \Phi_0(t) \\ \Phi_1(t) \\ \Phi_2(t) \\ \vdots \\ \vdots \\ \Phi_{L-1}(t) \end{pmatrix} = \Delta^{-2} \underbrace{\begin{pmatrix} 1+\Delta^2 V_0 & -1/2 & 0 & & & 0 \\ -1/2 & 1+\Delta^2 V_1 & -1/2 & & & \\ 0 & -1/2 & 1+\Delta^2 V_2 & & & \\ & & & \ddots & & 0 \\ & & & & 1+\Delta^2 V_{L-2} & -1/2 \\ 0 & & & 0 & -1/2 & 1+\Delta^2 V_{L-1} \end{pmatrix}}_{H} \begin{pmatrix} \Phi_0(t) \\ \Phi_1(t) \\ \Phi_2(t) \\ \vdots \\ \vdots \\ \Phi_{L-1}(t) \end{pmatrix}$$

source: Computational Physics - Lecture 17: Time-dependent Schrödinger equation I (page 39)(edited)

with $V_i = V(x_i)$.
The general solution for the Schrödinger equation is given by

$$\Phi(t) = e^{-itH}\Phi(0). \tag{35}$$

Taking the exponential of H is cumbersome, which is why we resort to the use of second-order product formula approximation. For that, we need to decompose H, such that undertaking matrix exponentials is much easier. We chose the decomposition of H into the three different matrices V, K_1, K_2, such that

$$H = V + K_1 + K_2 = \Delta^{-2}\begin{pmatrix} 1+\Delta^2 V_0 & 0 & 0 & & & 0 \\ 0 & 1+\Delta^2 V_1 & 0 & & & \\ 0 & 0 & 1+\Delta^2 V_2 & & & \\ & & & \ddots & & 0 \\ & & & & 1+\Delta^2 V_{L-2} & 0 \\ 0 & & & 0 & 0 & 1+\Delta^2 V_{L-1} \end{pmatrix}$$

$$+\Delta^{-2}\begin{pmatrix} \boxed{\begin{matrix} 0 & -1/2 \\ -1/2 & 0 \end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} & & \begin{matrix} 0 \\ \end{matrix} \\ \begin{matrix} 0 & \quad 0 \end{matrix} & 0 & & \\ & & \boxed{\begin{matrix} \ddots & -1/2 \\ -1/2 & 0 \end{matrix}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & & \begin{matrix} 0 & \quad 0 \end{matrix} & 0 \end{pmatrix} + \Delta^{-2}\begin{pmatrix} 0 & \begin{matrix} 0 & \quad 0 \end{matrix} & & 0 \\ \begin{matrix} 0 \\ 0 \end{matrix} & \boxed{\begin{matrix} 0 & -1/2 \\ -1/2 & 0 \end{matrix}} & & \\ & & \ddots & 0 \\ \begin{matrix} \\ 0 \end{matrix} & & \begin{matrix} \\ 0 \end{matrix} & \boxed{\begin{matrix} 0 & -1/2 \\ -1/2 & 0 \end{matrix}} \end{pmatrix}$$

source: Computational Physics - Lecture 17: Time-dependent Schrödinger equation I (page 49)(edited)

With the chosen decomposition, the second-order product formula equals

$$e^{-itH} = \left(e^{-i\tau H}\right)^n \approx \left(e^{-i\tau\frac{K_1}{2}} e^{-i\tau\frac{K_2}{2}} e^{-i\tau V} e^{-i\tau\frac{K_2}{2}} e^{-i\tau\frac{K_1}{2}}\right)^n. \tag{36}$$

With eq.(35), and the decomposition discussed, we have

$$\Phi(t) \approx \left(e^{-i\tau\frac{K_1}{2}} e^{-i\tau\frac{K_2}{2}} e^{-i\tau V} e^{-i\tau\frac{K_2}{2}} e^{-i\tau\frac{K_1}{2}}\right)^n \Phi(0). \tag{37}$$

In this calculation our approach is to solve from right to left. Therefore we solve the product between the matrixproduct between our $\Phi(0)$ and the matrix left to it. In that case we have to seperate between three cases, multiplying $e^{-i\tau V}$ or $e^{-i\frac{\tau}{2}K_{1/2}}$ with Ψ.[5,6]

Here, the advantages of our decomposition are made clear. The computation of $e^{-i\tau V}$, in particular, is very easy as it is a diagonal matrix. The matrix product in that case is given by

$$e^{-i\tau V}\Psi(t) = \mathrm{diag}(e^{-i\tau V_{00}}, e^{-i\tau V_{11}}, e^{-i\tau V_{22}}, ..., e^{-i\tau V_{(L-1)(L-1)}})\Psi(t) \tag{38}$$

such that for each element of $\Psi(t)$

$$e^{-i\tau V_{ii}}\Psi_i(t) = \exp\left(-i\tau \underbrace{\frac{1}{\Delta^2}(1+\Delta^2 V(x_i))}_{V_{ii}}\right)\Psi_i(t), \tag{39}$$

is calculated. This elementwise multiplication of two arrays, is implemented into `python` by creating an array consisting $\exp(-i\tau V_{ii})$ at all positions x_i and multiplying this with our array for $\Psi_i(t)$.

For

$$e^{-i\frac{\tau}{2}K_{1/2}}\Psi(x,t), \tag{40}$$

a different approach is chosen, due to the more complicated form of $K_{1/2}$. Fortunately, both K_1 and K_2 are composed of 2x2 matrices along the diagonal. Additionally, for K_1 the last entry is equal to zero, and for K_2, the first entry. To this end, multiplying the matrixexponentials with $\Psi(t)$, we have to differentiate between the two cases

$$\exp\left(+i\frac{\tau}{4\Delta^2}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right)\begin{pmatrix}\Psi_k(t)\\ \Psi_{k+1}(t)\end{pmatrix} \tag{41}$$

and

$$\exp\left(+i\frac{\tau}{4\Delta^2}\cdot 0\right)\Psi_k(t) = \Psi_k(t). \tag{42}$$

It is easy to show that eq.(41) can be rewritten into

$$\exp\left(+i\frac{\tau}{4\Delta^2}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right)\begin{pmatrix}\Psi_k(t)\\ \Psi_{k+1}(t)\end{pmatrix} = \begin{pmatrix}\cos(a) & i\sin(a)\\ i\sin(a) & \cos(a)\end{pmatrix}\begin{pmatrix}\Psi_k(t)\\ \Psi_{k+1}(t)\end{pmatrix}, \tag{43}$$

with $a = \frac{\tau}{4\Delta^2}$. With the last entry of K_1 being the zero entry, k in eq.(43) must take all even numbers from 0 to $L-2$. For K_2, the opposite is true, such that we have to take all the odd numbers from 1 to $L-1$.[7,8] The multiplication of a matrix with a vector is realized in `python` using `np.dot(a,b)`, which outputs the product of `a` and `b`.
The index k does not take all the numbers, but only the even ones for K_1 and the odd ones for K_2. To this end, choosing `for k in range(0,L-1,2)`, the counter `k` gets incremented by 2 in each step, such that with k we have the even numbers and with $k+1$ the odd.

[5] Ψ is an arbitrary array with the same dimensions as Φ

[6] $K_{1/2}$, which means either K_1 or K_2.

[7] here, with $K_{1/2}$ we are referring to $e^{-i\tau\frac{K_{1/2}}{2}}$

[8] There is no need for an explicit implementation of eq.(42), as doing no operation on an entry is the same as multiplying it by 1.

In the code snippet below, we show the implementation of the function `dt(phi)`. Said function takes in a $\Phi(x,t)$ as an input and returns $\Phi(x,t+\tau)$, which is the wavefunction's evolution for one time step τ. The iteration is done using a `for-loop` where the index n goes from 0 to m. The values of interest at specific times $t_k = k\tau$, are saved separately after iteration k.

```python
1  M    = np.array([[np.cos(a), 1j*np.sin(a)], [1j*np.sin(a), np.cos(a)]], dtype=np.
       complex128)
2
3  def dt(phi):
4      for k in range(0,len(phi)-1,2):               #K1
5          phi[k:k+2] = np.dot(M,phi[k:k+2])
6      for k in range(0,len(phi)-1,2):               #K2
7          phi[k+1:k+3] = np.dot(M,phi[k+1:k+3])
8      phi = Vx*phi                                  #V part
9      for k in range(0,len(phi)-1,2):               #K2
10         phi[k+1:k+3] = np.dot(M,phi[k+1:k+3])
11     for k in range(0,len(phi)-1,2):               #K1
12         phi[k:k+2] = np.dot(M,phi[k:k+2])
13     return phi
```

Using `phi[k:k+2]` (line 5 and 12 in code snippet), the vector (Φ_k, Φ_{k+1}) is sliced from the total array.[9]

Apart from the wave function, we are interested in the probability, defined as

$$P(x,t) = |\Phi(x,t)|^2\Delta \quad \Rightarrow \quad P(x_i,t) = |\Phi_i(t)|^2\Delta.^{10} \tag{44}$$

Next, we dedicate ourselves to the calculation of the mean position and variance. To this end, we have to discretize eq.(5) such that

$$\langle x_i(t)\rangle = \sum_{i=0}^{L-1} x_i P(x_i,t) \quad \text{and} \quad \langle x_i^2(t)\rangle = \sum_{i=0}^{L-1} x_i^2 P(x_i,t). \tag{45}$$

These two sums, can be implemented into `python` using

$$\langle x_i(t)\rangle = \texttt{np.sum(x*P*Delta)} \qquad \text{and} \qquad \langle x_i^2(t)\rangle = \texttt{np.sum(x**2*P*Delta)},$$

with $\texttt{P} = P(x_i,t)/\Delta$. From there the variance can be calculated according to eq.(28):

$$\mathrm{Var}(x_i) = \langle x_i^2(t)\rangle - \langle x_i(t)\rangle^2 .$$

A remark about the use of `multiprocessing` can be found in the appendix.

[9] similar for the odd case (line 7 and 10 in code snippet)

[10] The absolute value can be calculated using `np.absolute()`.

4 Simulation results

In this section, we present the solutions we obtained for the algorithm. To this end, we vary the parameters that affect the simulation. We study five different combinations of the parameters

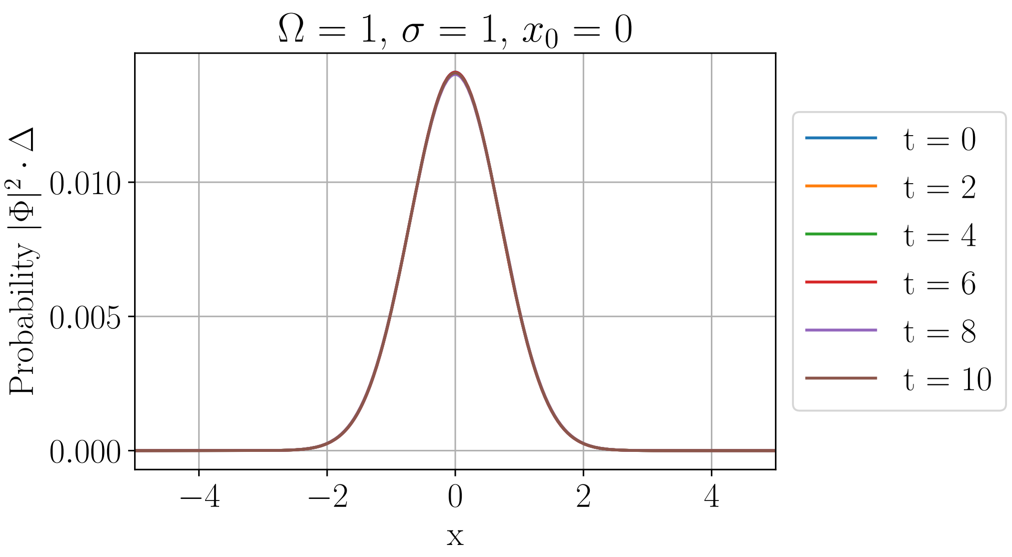
$$(\Omega, \sigma, x_0) = \underbrace{(1,1,0)}_{(a)},\ \underbrace{(1,1,1)}_{(b)},\ \underbrace{(1,2,0)}_{(c)},\ \underbrace{(2,1,1)}_{(d)},\ \underbrace{(2,2,2)}_{(e)}. \tag{46}$$

For the spatial discretization, we chose $\Delta = 0.025$ and $L = 1201$. For the discretization in time $\tau = 0.00025$ and $m = 40000$, such that the algorithm stops at $t = m\tau = 10$.

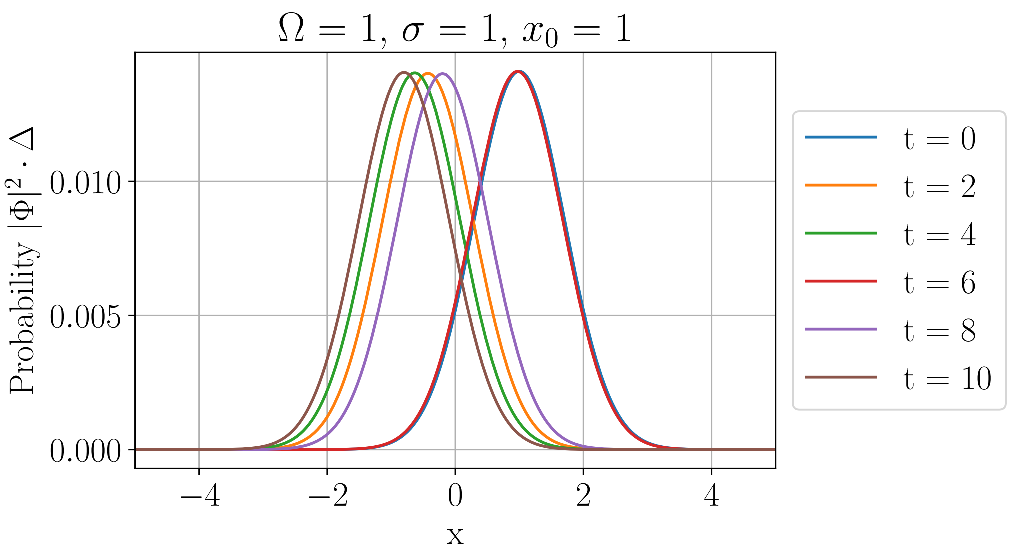
In fig.(1), the probability $P(x,t) = |\Phi(x,t)|^2 \cdot \Delta$, are plotted for $t \in \{0, 2, 4, 6, 8, 10\}$. As the Gaussian packet is localized in the vicinity of zero, we chose to plot x from -5 to 5 for better visualization. In the appendix fig.(4), the plots from -15 to 15 can be found.

For the positions' averages and variances, we plotted the analytical expectation according to eq.(13) and eq.(29) in the same plot fig.(2).

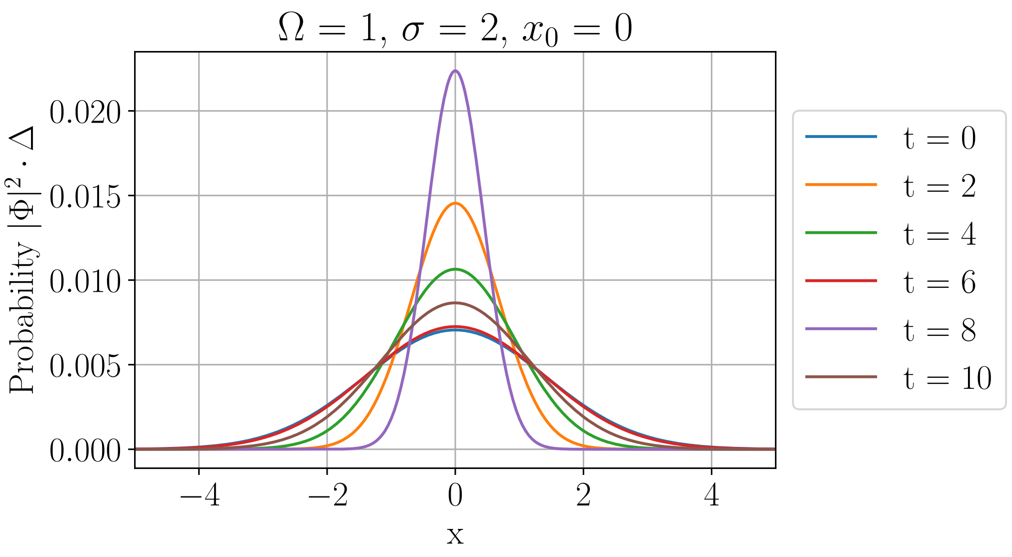
Additionally to compare the theory and simulation, in fig.(3), the differences between the theory and simulation can be found.


$\Omega = 1,\, \sigma = 1,\, x_0 = 0$
Probability $|\Phi|^2 \cdot \Delta$
x
t = 0
t = 2
t = 4
t = 6
t = 8
t = 10

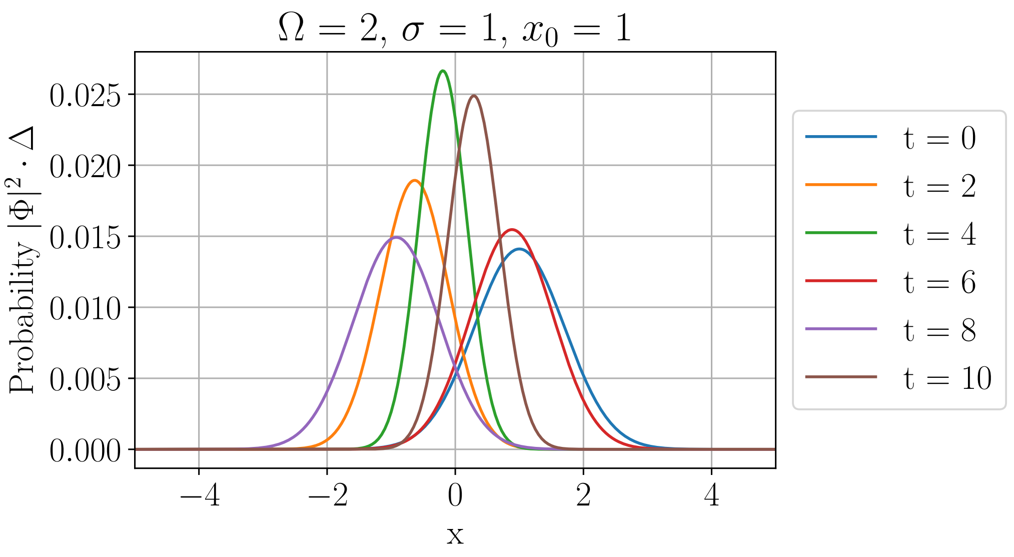
(a)


$\Omega = 1,\, \sigma = 1,\, x_0 = 1$
Probability $|\Phi|^2 \cdot \Delta$
x
t = 0
t = 2
t = 4
t = 6
t = 8
t = 10

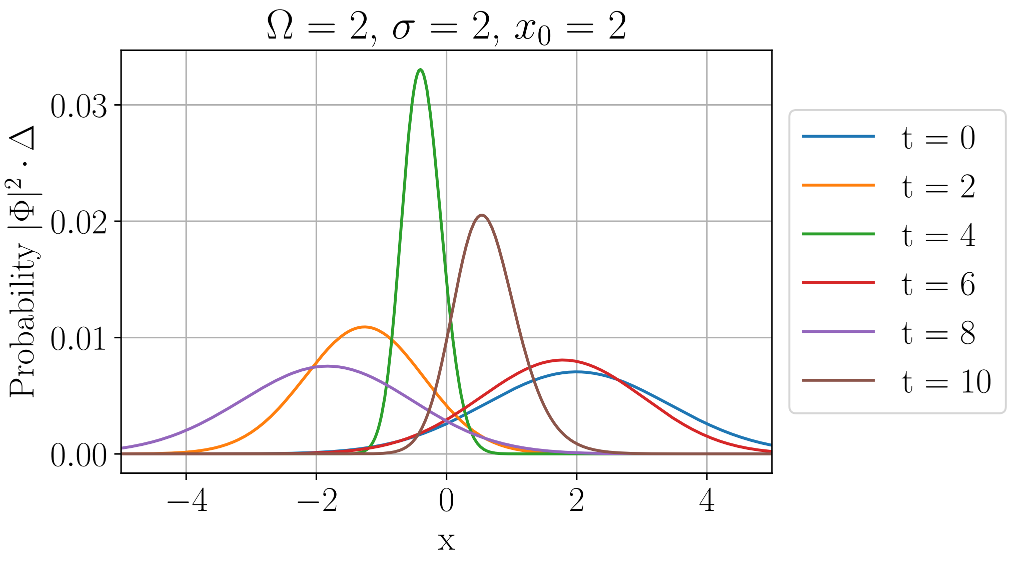
(b)


$\Omega = 1,\, \sigma = 2,\, x_0 = 0$
Probability $|\Phi|^2 \cdot \Delta$
x
t = 0
t = 2
t = 4
t = 6
t = 8
t = 10

(c)

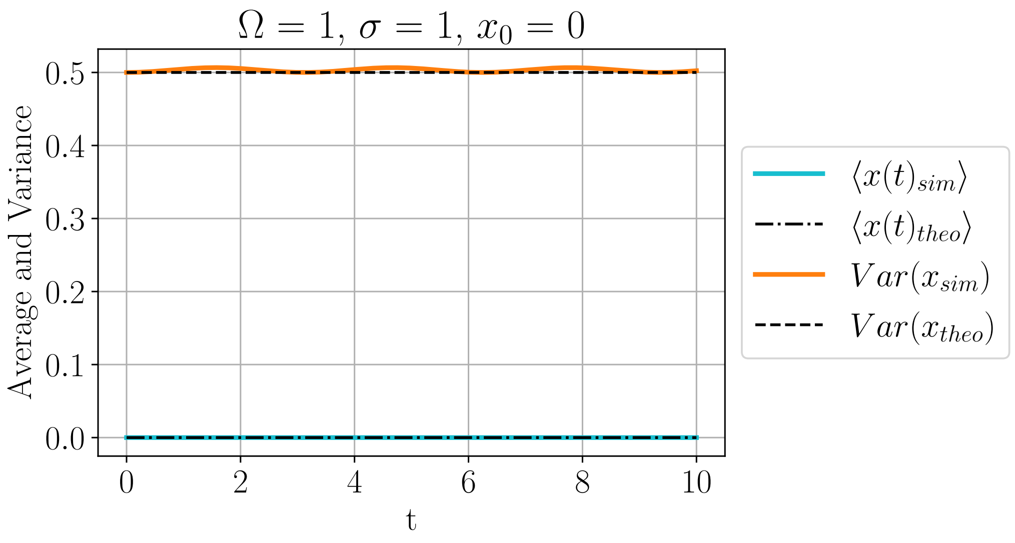


(d)



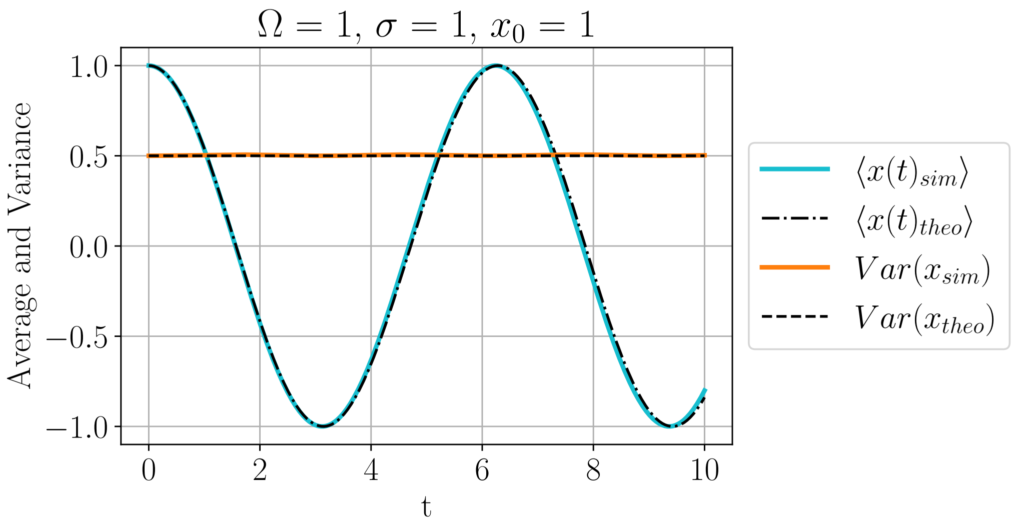
(e)

Figure 1: Simulation results for the probability $P(x,t) = |\Phi(x,t)|^2 \cdot \Delta$. In the title of each plot, the parameters Ω, σ, and x_0 used are specified. The functions are plotted for the six different times $t = 0, 2, 4, 6, 8, 10$. The system is bounded to $-15 \leq x \leq 15$, but only -5 to 5 is shown here. Note that in (a) all the probabilities coincide, which is why it looks like only plotting $t = 10$



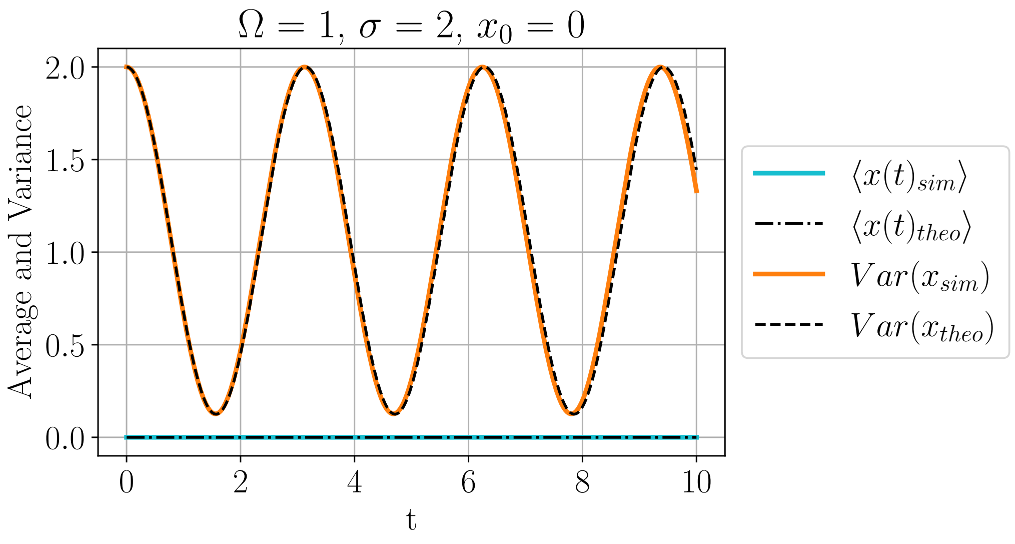
$\Omega = 1$, $\sigma = 1$, $x_0 = 0$
Average and Variance
t
$\langle x(t)_{sim} \rangle$
$\langle x(t)_{theo} \rangle$
$Var(x_{sim})$
$Var(x_{theo})$

(a)



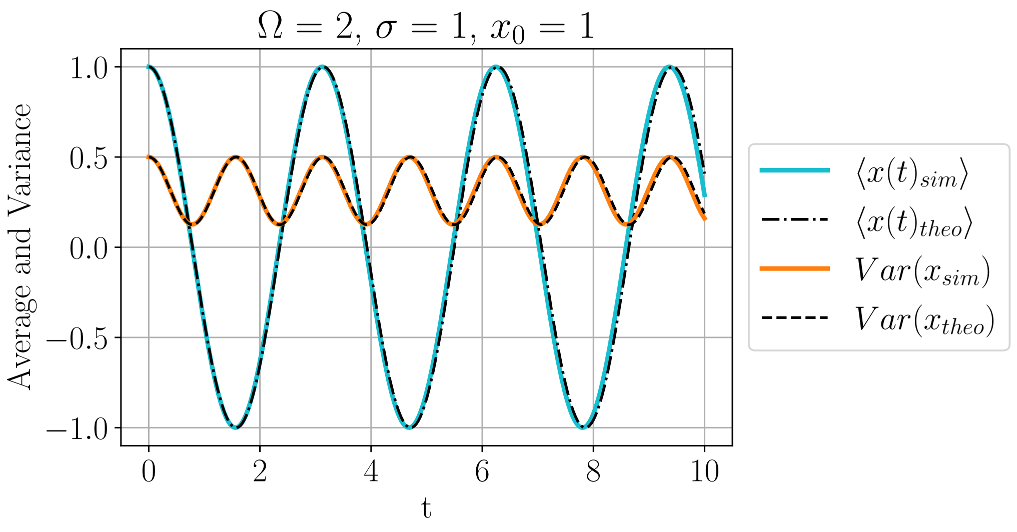
$\Omega = 1$, $\sigma = 1$, $x_0 = 1$
Average and Variance
t
$\langle x(t)_{sim} \rangle$
$\langle x(t)_{theo} \rangle$
$Var(x_{sim})$
$Var(x_{theo})$

(b)

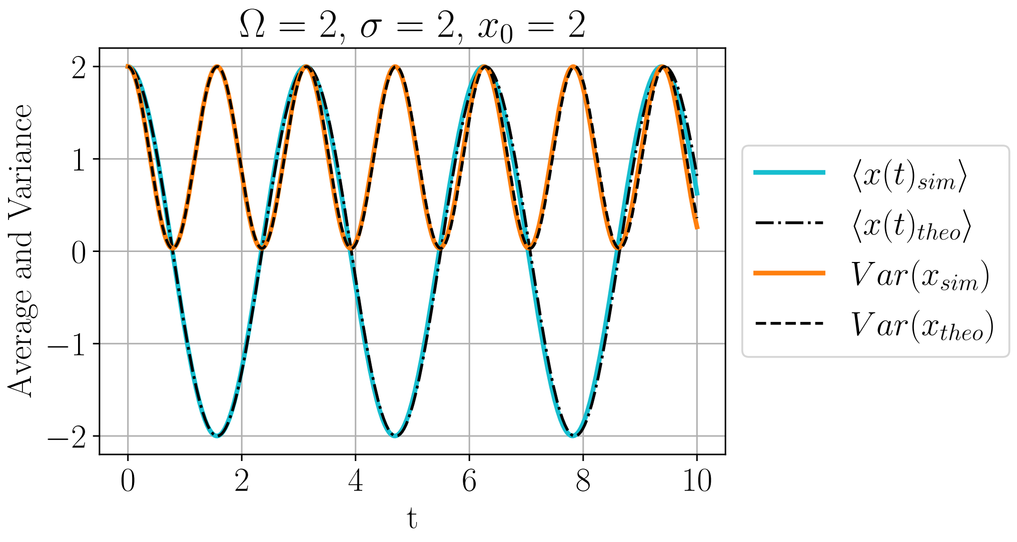


$\Omega = 1$, $\sigma = 2$, $x_0 = 0$
Average and Variance
t
$\langle x(t)_{sim} \rangle$
$\langle x(t)_{theo} \rangle$
$Var(x_{sim})$
$Var(x_{theo})$

(c)

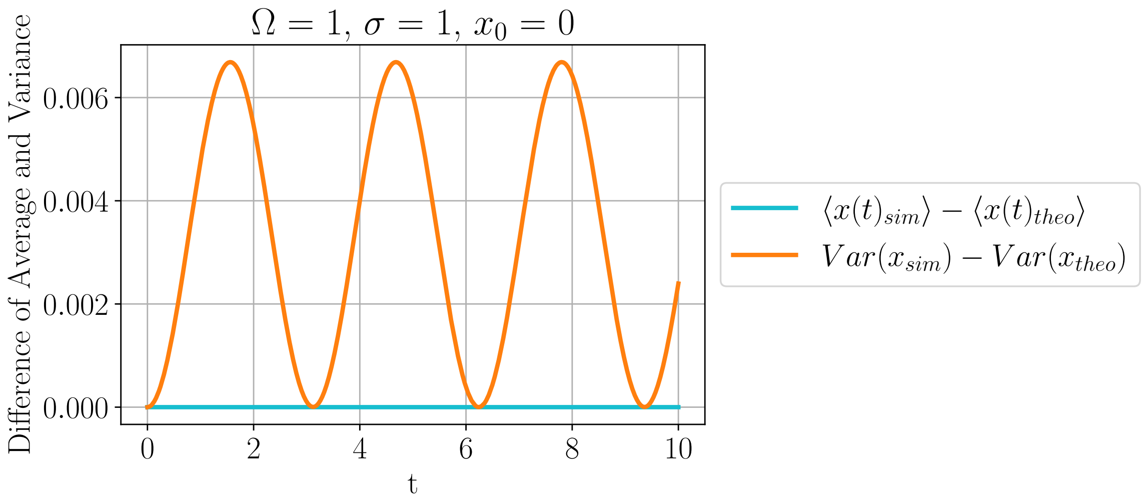


(d)



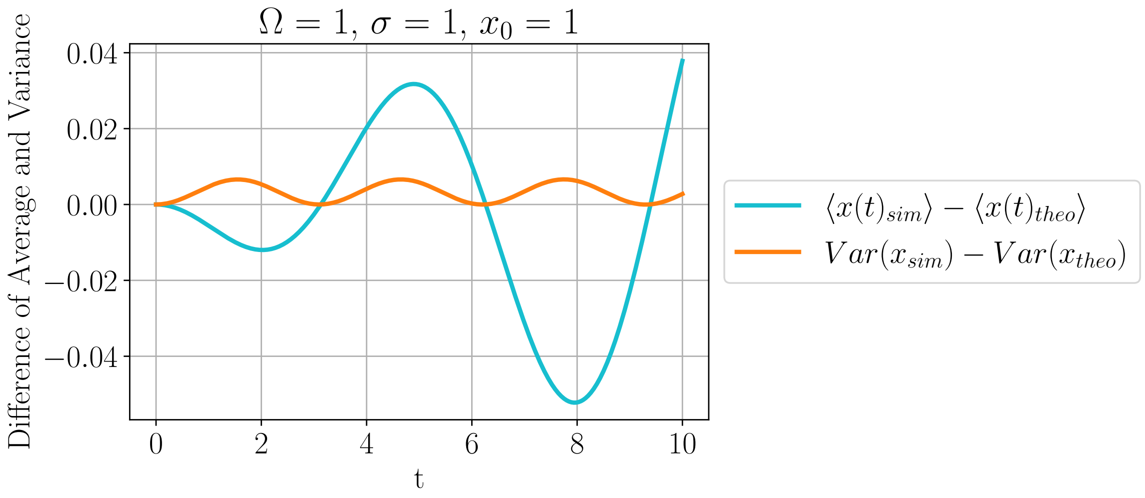
(e)

Figure 2: Simulation results for the average and variances for the position. In the title of each plot, the parameters Ω, σ, and x_0 used are specified. The solid lines show the results according to the simulation model and the dashed line the theoretical expectation according to eq.(13) and eq.(29). *theo*, corresponds to the theory and *sim* to the simulation results



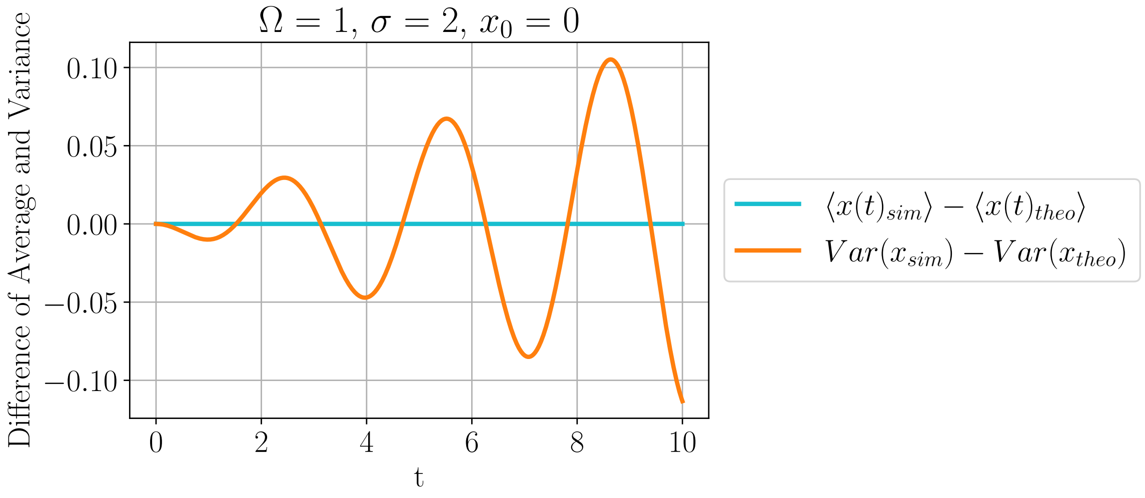
$\Omega = 1$, $\sigma = 1$, $x_0 = 0$
Difference of Average and Variance
0.006
0.004
0.002
0.000
0
2
4
6
8
10
t
$\langle x(t)_{sim} \rangle - \langle x(t)_{theo} \rangle$
$Var(x_{sim}) - Var(x_{theo})$

(a)



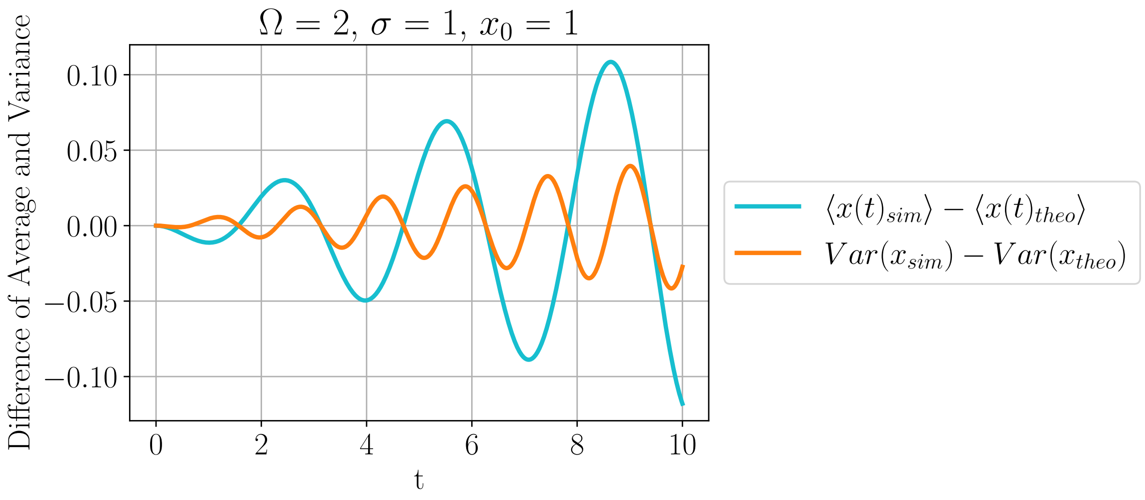
$\Omega = 1$, $\sigma = 1$, $x_0 = 1$
Difference of Average and Variance
0.04
0.02
0.00
−0.02
−0.04
0
2
4
6
8
10
t
$\langle x(t)_{sim} \rangle - \langle x(t)_{theo} \rangle$
$Var(x_{sim}) - Var(x_{theo})$

(b)

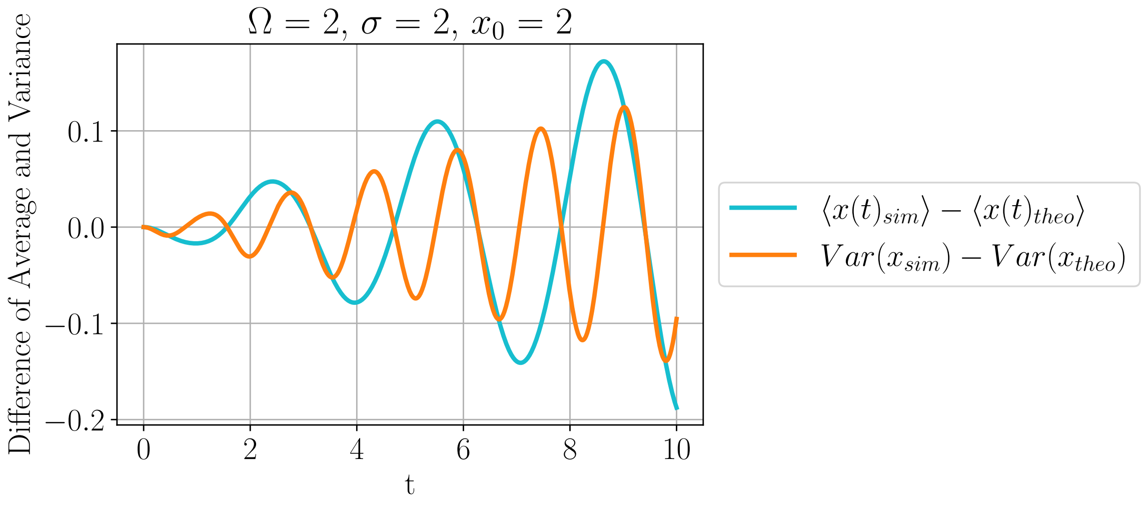


$\Omega = 1$, $\sigma = 2$, $x_0 = 0$
Difference of Average and Variance
0.10
0.05
0.00
−0.05
−0.10
0
2
4
6
8
10
t
$\langle x(t)_{sim} \rangle - \langle x(t)_{theo} \rangle$
$Var(x_{sim}) - Var(x_{theo})$

(c)



(d)



(e)

Figure 3: Difference between simulation results and theoretical expectation for the average and variances for the position. In the title of each plot, the parameters Ω, σ, and x_0 used are specified. *theo*, corresponds to the theory and *sim* to the simulation results

5 Discussion

We start the discussion by describing fig.(1). To this end, we point out the main similarities and differences. (a) and (b) share the property that the probability function does not change its shape with time, i.e. the variance stays constant. In fig.(2) this becomes more obvious as the Variance in (a) and (b) stay constant at 1/2 over time. However, here, it needs to be noted that the simulation does not exactly match with our theory, this is not only restricted to (a) and (b) but to all the results. This, we investigate later. The reason why (a) and (b) are so different from the other cases is due to the fact that (a) is an energy eigenstate and (b) a coherent state. Also (a) can be viewed as a coherent state, due to the fact that the vacuum itself is also a coherent one. Case (b), describes a coherent state with non-zero α.[11] The interesting property of coherent states is that they resemble classical behavior. Imagine a ball in a harmonic potential, such that the system is a classical harmonic oscillator. If the initial position of the ball is at the minimum, with the initial momentum equal to 0, we do not expect anything to happen with the ball. This is analogous to our case (a). If, however, the ball is slightly displaced from the minimum, the ball starts to oscillate to the left and right, similar to our case (b). Additionally, the fact that the wave function is preserving its shape, is because coherent states stay coherent in the quantum harmonic oscillator. This in particular is very interesting, as these states are not eigenstates of our Hamiltonian.

In (c), the behavior is striking, as it is the only case, where the average position stays constant, that is at 0, but the variance oscillates over time. The frequency at which the variance oscillates is 2Ω. Also, the difference between (a) and (c) is only the σ, but contrary to (a) the variance oscillates. This is due to eq.(29), where for $\Omega, \sigma = 1$, the variance is equal to a constant. But there is an underlying relation between Ω and σ, that governs the cases when the variance does not change over time. This relation can be derived from eq.(29) when the constants in front of the $\sin^2(\Omega t)$ and $\cos^2(\Omega t)$ are the same, or simply put

$$\frac{1}{\sigma^2\Omega^2} = \sigma^2 \quad \rightarrow \quad \Omega\sigma^2 = \pm 1.^{12} \tag{47}$$

We can see that our initial conditions for (a) and (b), satisfy this relation.

Considering (d) and (e), we see they are very similar. In both cases, we have a wave packet that is initially displaced from 0, which is why it oscillates back and forth. While (d) starts at $x_0 = 1$, (e) starts at $x_0 = 2$. Comparing the average positions in fig.(2) for both (d) and (e), one sees that in both cases the frequency is the same. This was also shown in the theoretical derivation (see eq.(13)), that the frequency of the average does only depend on Ω. This behavior matches our expectations from classical mechanics. Independent of the initial position of the ball, the oscillation frequency is the same. The solution for the position of the ball equals the solution for our mean position. Looking at the variances of (d) and (e), one can see that the variance oscillates in these cases too. For (d) and (e) Ω was chosen twice as large as in (c), which is why the variance oscillates with double the frequency of (c). The difference between (d) and (e) is, that for choosing a different σ, the amplitude of the variance oscillation changes.

Lastly, when comparing the oscillation of the variance with the oscillation of the mean position in (d) and (e), the variance has double the frequency. This implies in (d) and (e) that the wavepacket contracts when moving to the minimum and expands moving away from it. Thus at $\pm x_0$, we have the highest variance for the wavepacket.

[11] α defines the amplitude and phase of a coherent state

[12] -1 can be ignored, as a negative frequency does not make sense

Additonally, as the energy eigenstates form a complete basis, (b), (c), (d), and (e) are superpositions of the energy eigenstates. As (a) is an eigenstate itself, it is stationary.

After having discussed the simulation results and comparing them with each other, we turn to the theoretical expectation. In fig.(2), additionally to our simulation the theoretical results derived in section 2 are plotted as the dashed line. In (a), we can see how the theoretical expectation for $\langle x(t)\rangle$ matches the results very well. However, for the variance, it seems like our simulation oscillates above the expectation. These are probably artifacts due to discretizations and approximations, as we have seen similar behavior in the exercise about "Molecular Dynamics Simulation". For further investigation, fig.(3) shows the differences between the simulation and theoretical result. Here, our hypothesis of oscillation above the theoretical expectation is proved to be true. In (b), similar to (a), the variance oscillates although again the theoretical expectation should be constant. But in both (a) and (b), it seems that the amplitude of the oscillation does not change over time. For (c), (d), and (e), the difference between simulation and theory, oscillates, but the amplitude increases gradually. For the mean position, we have similar behavior. In (b), (d), and (e), the difference oscillates, and at the same time the amplitude increases. Only for (a) and (c), the difference stays constant and is equal to 0. But these are the cases in which the position does not change at all, hence it seems that $x_0 = 0$ cancels the error that is responsible for the difference. As the theory predicts $x_0 \cos(\Omega t)$, we expect that the error lies in the frequency Ω such that our simulation does not compute the exact $\cos(\Omega t)$, but a small deviation from that. The simulation seems to osicillate with higher frequency than the theory, which is best seen for greater values of t. It has probably to do with the second-order product formula only being an approximation and due to discretizations of the space.

Overall one can summarize that the simulation method using the second-order product formula and discretizing the system yields results that are in good reasoning with the theory. However, it needs to be pointed out that any approximation is only an approximation. Additionally, apart from approximations numerical uncertainties are also a source of errors. Nevertheless, the method used is very viable to produce good results. The advantage of this simulation is how versatile it is, changing the initial wavepacket or parameters can be done easily, whereas doing it analytically would take much more time.

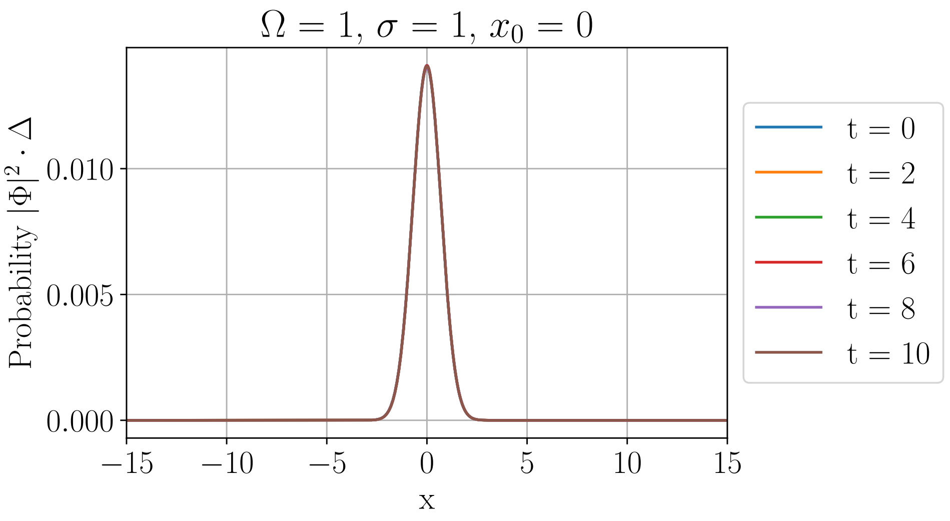
Appendix A Code

In the section about the simulation model, the use of multiprocessing was not discussed. As this only makes the code more efficient, it was of no avail to explain it there. The basic idea is to calculate the same function with different arguments parallel. This has the advantage that the runtime is reduced drastically for five different inputs.

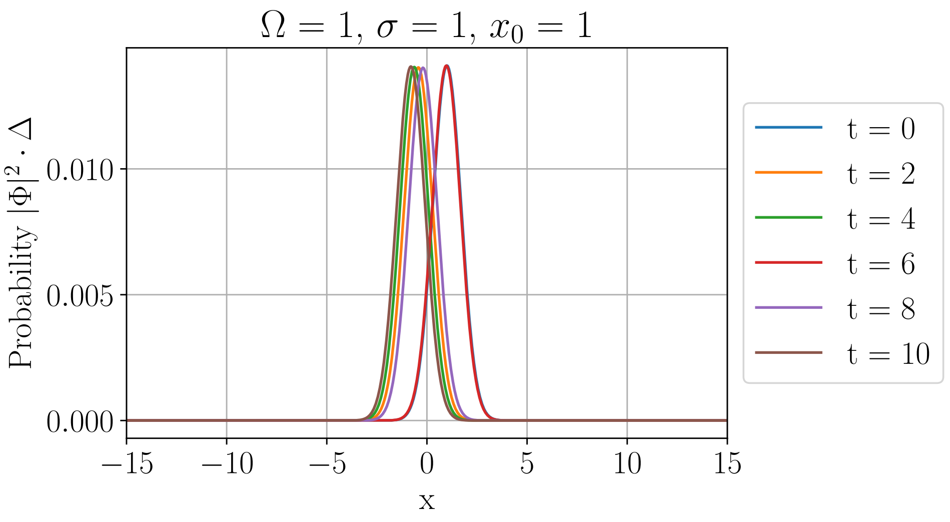
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 14 15:46:58 2023

@author: tamilarasan
"""
#import important modules
import numpy as np
import matplotlib.pyplot as plt
from numba import njit, vectorize, float64

from multiprocessing import Pool

#some plotting arguments for better visualization
import os
os.environ["PATH"] += os.pathsep + '/Library/TeX/texbin'

plt.rcParams["text.usetex"] = True
plt.rcParams["font.family"] = "times new roman"
plt.rcParams["font.size"] = "18"

pi      = np.pi


#parameters for the discretizations
Delta = 0.025
L = 1201
tau = 0.00025
m = 40000


#function that solves the system Input is an array consisting of [Omega, sigma, x0]
def solve(Input):
    Omega, sigma, x0 = Input

    #Matrix M as explained in simulation model
    c = np.cos(tau/(4*Delta**2))
    s = 1j*np.sin(tau/(4*Delta**2))
    M = np.array([[c,s],[s,c]], dtype=np.complex128)

    #discretization of space
    x = np.linspace(-15,15,L)

    #function that calculates initial wavepacket
    @njit
    def Phi0(x):
        return (pi*sigma**2)**(-1/4)*np.exp(-(x-x0)**2/(2*sigma**2))

    #the initial wavepacket
    Phi = np.array(Phi0(x)).astype(np.complex128)

    #function for the the harmonic potential
    #function is vectorized to input array
    @njit
    def V(x):
        return 1/2*Omega**2*x**2
    Vvec= np.vectorize(V)
    Vx = np.exp(-1j*tau*(Delta**(-2)+Vvec(x)))

    #function that calculates one time step tau evolution
    #input is a phi(t) output is phi(t+tau)
    @njit
    def dt(phi):
        for k in range(0,len(phi)-1,2):                #K1
            phi[k:k+2] = np.dot(M,phi[k:k+2])
        for k in range(0,len(phi)-1,2):                #K2
            phi[k+1:k+3] = np.dot(M,phi[k+1:k+3])
        phi = Vx*phi                                   #V part
        for k in range(0,len(phi)-1,2):                #K2
            phi[k+1:k+3] = np.dot(M,phi[k+1:k+3])
        for k in range(0,len(phi)-1,2):                #K1
            phi[k:k+2] = np.dot(M,phi[k:k+2])
        return phi

    #array that will consist of mean values of x and x^2
    X = np.zeros(m+1)
    Xsq = np.zeros(m+1)
    #time at which the plots are plotted
    tprint = [0,2,4,6,8,10]
    #array in which the probabilities at tprint will be saved
    Pt = []
    #doing m iterations
    for i in range(m):
        P = np.abs(Phi)**2
        if round(i*tau,5) in tprint: #if statement to get the probabilities at times
    defined in tprint
            Pt+= [P]
        X[i] = np.sum(x*P*Delta)
        Xsq[i] = np.sum(x**2*P*Delta)
        Phi = dt(Phi)
    #after last iteration the last values (t=10) have to be saved seperately
    P = np.abs(Phi)**2
    Pt += [P]
    X[m] = np.sum(x*P*Delta)
    Xsq[m] = np.sum(x**2*P*Delta)
    return X,Xsq,Pt



#function that plots the values obtained
def plot(task,arg):
    Omega, sigma, x0 = arg
    legend = [[1,1,0],[1,1,1],[1,2,0],[2,1,1],[2,2,2]]        #for which parameters
    the plot get legends
    X,Xsq,P = task                    #solutions from function solve(arg)
    t = np.linspace(0,m*tau,m+1)      #timearray
    x = np.linspace(-15,15,L)         #position discretization

    #theoretical expectation of x and x^2
    xth = x0*np.cos(Omega*t)
    xsqth = 1/(2*Omega**2*sigma**2)*(np.sin(Omega*t))**2+ 1/2*(sigma**2+2*x0**2)*(np.
    cos(Omega*t))**2
    tprint = [0,2,4,6,8,10] #time at which the probabilities are plotted

    #plotting the averages (both theory and simulation)
    plt.grid()
    plt.plot(t, X, label =r" $\langle x(t)_{sim} \rangle$", color ="tab:cyan", lw = 2.5)
    plt.plot(t, xth, label = r"$\langle x(t)_{theo} \rangle$", color = "black", ls = "-.", lw = 1.5)
    plt.plot(t, Xsq-X**2,label =r"$Var(x_{sim})$", color ="tab:orange", lw = 2.5)
    plt.plot(t, xsqth-xth**2, label = r"$Var(x_{theo})$", color = "black",ls = "--", lw = 1.5)
    plt.xlabel("t")
    plt.ylabel("Average and Variance")
    plt.title(r"$\Omega$ = "+str(Omega)+r", $\sigma$ = "+str(sigma)+r", $x_0$ = "+str(x0))
    if arg in legend:
        plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.savefig(f"plot/Omega{Omega}_sigma{sigma}_x0{x0}_Averages.pdf",bbox_inches='tight')
    plt.show()

    #plotting the differences between theory and simulation
    plt.grid()
    plt.plot(t, X-xth, label = r"$\langle x(t)_{sim} \rangle - \langle x(t)_{theo} \rangle$", color = "tab:cyan", lw = 2.5)
    plt.plot(t, Xsq-X**2 - (xsqth-xth**2),label =r"$Var(x_{sim}) -Var(x_{theo})$", color ="tab:orange", lw = 2.5)
    plt.xlabel("t")
    plt.ylabel("Difference of Average and Variance")
    plt.title(r"$\Omega$ = "+str(Omega)+r", $\sigma$ = "+str(sigma)+r", $x_0$ = "+str(x0))
    if arg in legend:
        plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.savefig(f"plot/Omega{Omega}_sigma{sigma}_x0{x0}_Averages_expect.pdf", bbox_inches='tight')
    plt.show()

    #plotting the probability from -5 to 5
    for i in range(6):
        plt.plot(x,P[i]*Delta, label = "t = "+str(tprint[i]))
    plt.title(r"$\Omega$ = "+str(Omega)+r", $\sigma$ = "+str(sigma)+r", $x_0$ = "+str(x0))
    plt.xlabel("x")
    plt.xlim(-5,5)
    plt.grid()
    plt.ylabel(r"Probability $|\Phi|^2\cdot \Delta$")
    if arg in legend:
        plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.savefig(f"plot/Omega{Omega}_sigma{sigma}_x0{x0}_Probabilities.pdf", bbox_inches='tight')
    plt.show()

    #plotting the probability from -15 to 15
    for i in range(6):
        plt.plot(x,P[i]*Delta, label = "t = "+str(tprint[i]))
    plt.title(r"$\Omega$ = "+str(Omega)+r", $\sigma$ = "+str(sigma)+r", $x_0$ = "+str(x0))
    plt.xlabel("x")
    plt.xlim(-15,15)
    plt.grid()
    plt.ylabel(r"Probability $|\Phi|^2\cdot \Delta$")
    if arg in legend:
        plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.savefig(f"plot/Omega{Omega}_sigma{sigma}_x0{x0}_Probabilities_full.pdf", bbox_inches='tight')
    plt.show()
```

```python
    return 0

#multiprocessing to calculate all the initial values at the same time
if __name__ == '__main__':
    pool = Pool()
    args = [[1,1,0],[1,1,1],[1,2,0],[2,1,1],[2,2,2]]
    Sol = pool.map(solve, args)

    for i in range(5):
        plot(Sol[i],args[i])
```

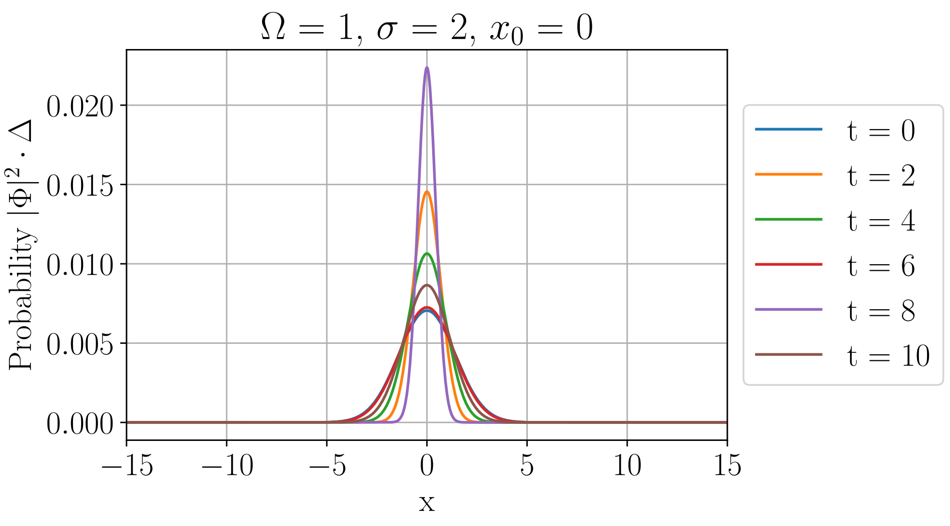
Appendix B Probability $-15 \leq x \leq 15$



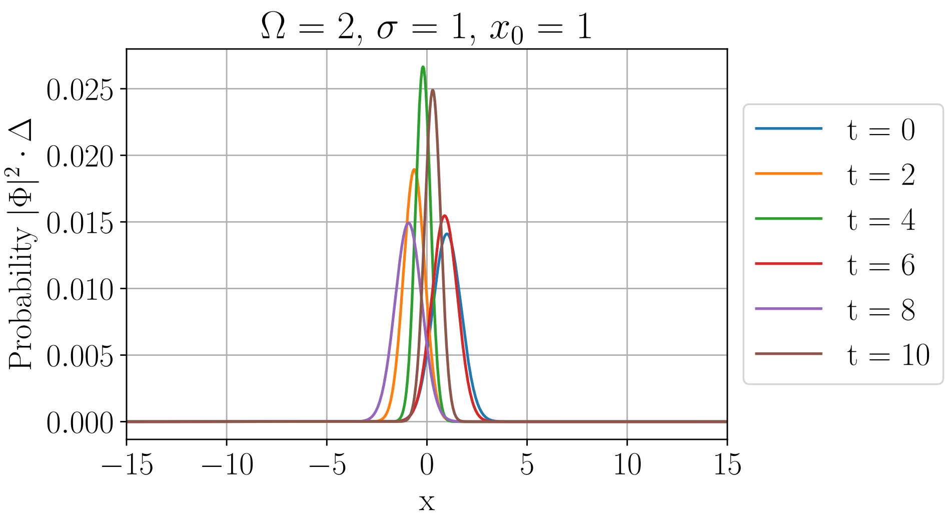
(a)



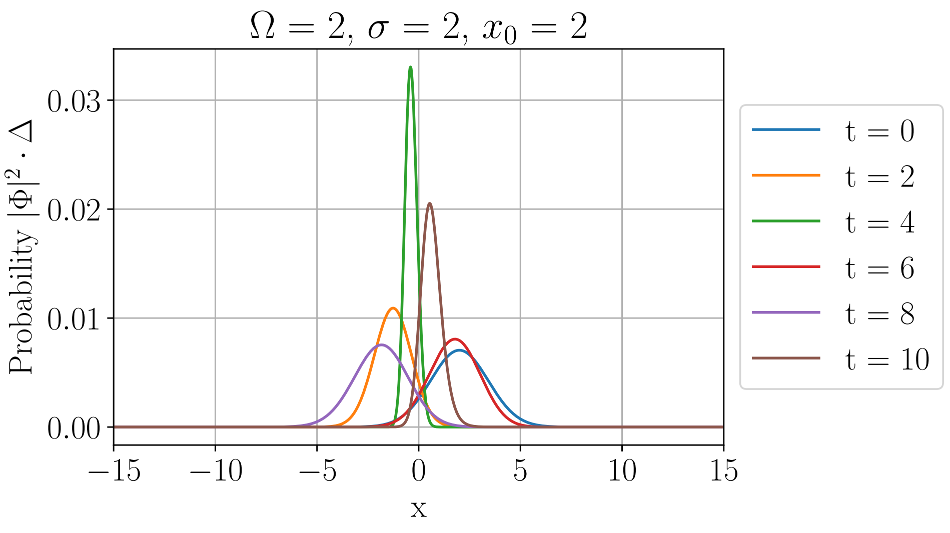
(b)



(c)



(d)



(e)

Figure 4: Simulation results for the probability $|\Phi(x,t)|^2$. In the title of each plot, the parameters Ω, σ, and x_0 used are specified. The functions are plotted for the six different times $t = 0, 2, 4, 6, 8, 10$. The system is bounded to $-15 \leq x \leq 15$.