ASSIGNMENT 5

Physics of Data – Quantum Information and Computing

A.Y. 2024/2025

Lorenzo Calandra Buonaura

02/12/2024

THEORY (1)

- **Time-dependent quantum harmonic oscillator**

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{\omega^2}{2m}(\hat{q} - q_0(t)) \qquad \text{with} \;\; q_0(t) = \frac{t}{T} \qquad (t \in [0, T])$$

- **Time evolution in real time:**
  - Schrödinger equation:

$$i\hbar \frac{\partial \psi(x,t)}{\partial t} = \hat{H}\, \psi(x,t)$$

  - Wavefunction evolution:

$$\psi(x,t) = \psi(x,0)\, e^{-\frac{i\hat{H}t}{\hbar}} = \sum_n c_n(0)\, e^{-\frac{iE_n t}{\hbar}} \psi_n(x)$$

  - **Expected behaviour:**

Dynamic evolution of the states, expected oscillations at frequencies proportional to $E_n/\hbar$.

- **Time evolution in imaginary time**
  - Change of variable

$$\tau = it$$

  - Schrödinger equation

$$-\hbar \frac{\partial \psi(x,\tau)}{\partial \tau} = \hat{H}\, \psi(x,\tau)$$

  - Wavefunction evolution

$$\psi(x,\tau) = \psi(x,0)\, e^{-\frac{\hat{H}\tau}{\hbar}} = \sum_n c_n(0)\, e^{-\frac{E_n \tau}{\hbar}} \psi_n(x)$$

  - **Expected behaviour:**

Eigenstates decay exponentially at rates proportional to $E_n/\hbar$ (for large τ the ground state is projected out).

THEORY (2)

- **Split-operator method**
  - Numerical approach used to solve the time-dependent Schrödinger equation, used for systems with Hamiltonians of the form (where $T(p)$ is the kinetic energy term and $\mathrm{V}(x)$ is the potential energy term):

$$\hat{\mathrm{H}} = T(p) + V(x)$$

  - Leverages **Baker-Campbell-Hausdorff** formula for approximating the time evolution operator using operator splitting (accurate at second order in Δt).

$$e^{-\frac{i\hat{\mathrm{H}}\Delta t}{\hbar}} \approx e^{-\frac{i\hat{T}\Delta t}{2\hbar}} e^{-\frac{i\hat{V}\Delta t}{\hbar}} e^{-\frac{i\hat{T}\Delta t}{2\hbar}}$$

  - Computationally efficient: requires only Fourier transforms and pointwise operations.

  - Works only for small Δt: using large time steps can lead to errors.

CODE DEVELOPMENT (1)

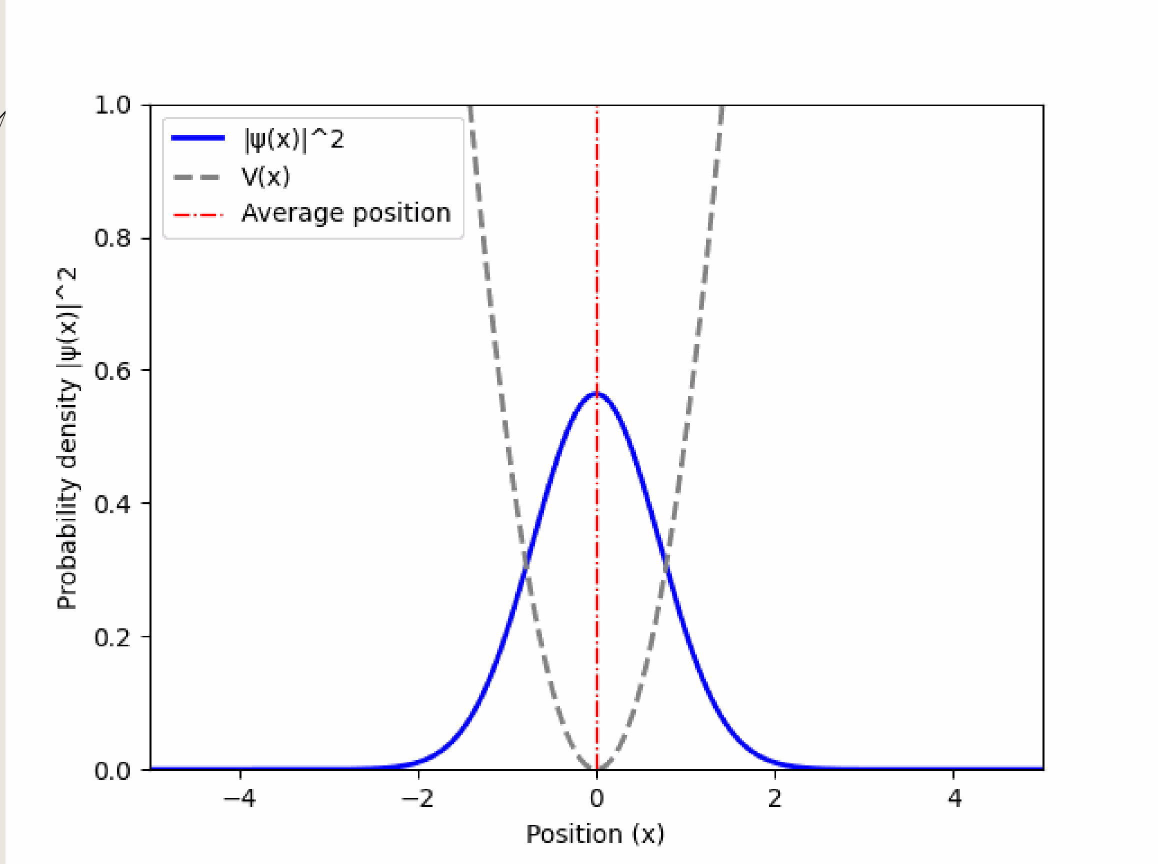
```
class Param:
  """
  Param:
    Container for holding all simulation
    parameters.
  """
  def __init__(self,
               x_min: float,
               x_max: float,
               num_x: int,
               tsim: float,
               num_t: int,
               im_time: bool = False) -> None:
        …

  def _validate(self) -> None:
    """
    _validate :
      Check for common errors in parameter
      initialization.
    """
        …
```

```
class Operators:
  """
  Container for holding operators and
  wavefunction coefficients.
  """
  def __init__(self,
               res: int,
               voffset: float = 0,
               wfcoffset: float = 0,
               omega: float = 1.0,
               order: int = 2,
               n: int = 0,
               q0_func=None,
               par: Param = None) -> None:
        …

  def _initialize_operators(self, par: Param,
    voffset: float, wfcoffset: float,
    order: int, n: int) -> None:
    """
    _initialize_operators:
      Initialize operators and wavefunction
      based on the provided parameters.
        …

   def calculate_energy(self, par: Param) -> float:
    """
    calculate_energy:
      Calculate the energy <Psi|H|Psi>.
```

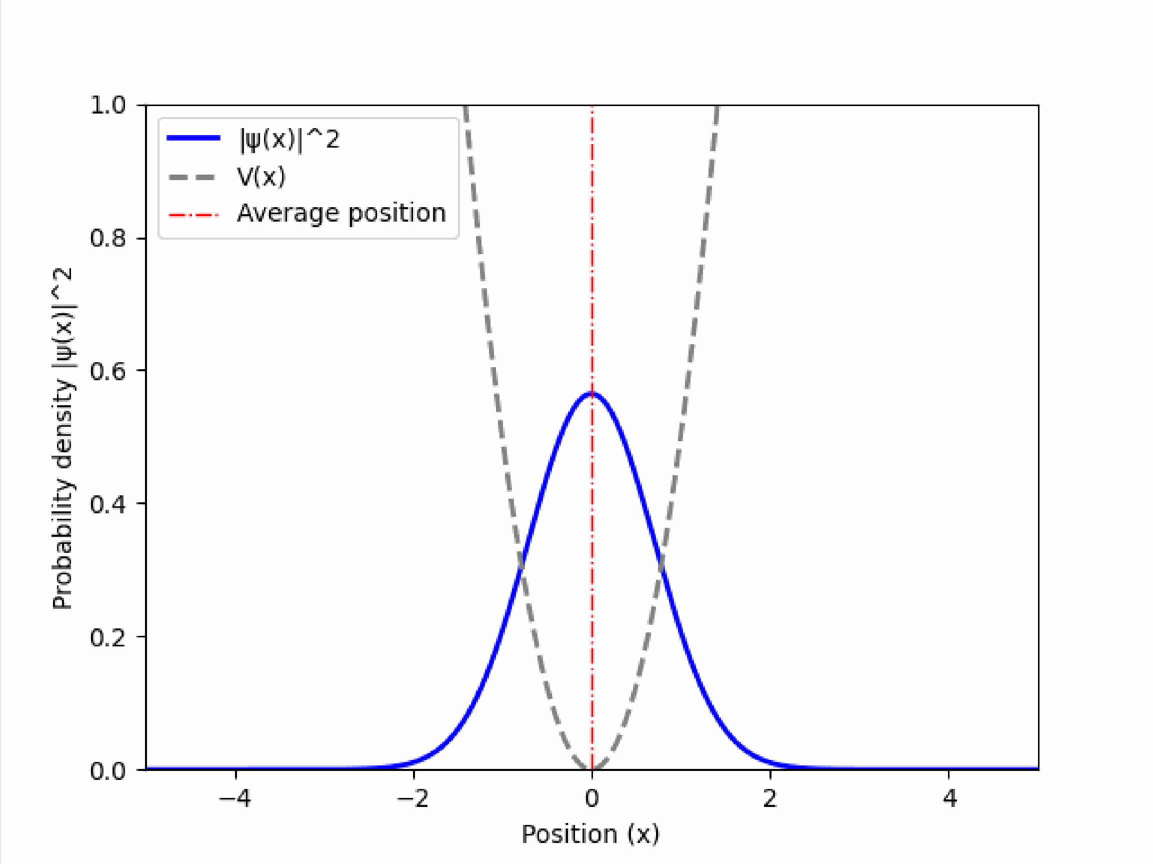
CODE DEVELOPMENT (2)

```
def split_op(par: Param, opr: Operators) -> None:
  for i in range(par.num_t):
    q0 = opr.q0_func(i * par.dt)
    opr.V = 0.5 * (par.x - q0) ** 2 * opr.omega ** 2


    # Evolution
    coeff = 1 if par.im_time else 1j
    opr.R = np.exp(-0.5 * opr.V * par.dt * coeff)
    opr.wfc *= opr.R
    opr.wfc = np.fft.fft(opr.wfc)
    opr.wfc *= opr.K
    opr.wfc = np.fft.ifft(opr.wfc)
    opr.wfc *= opr.R

    # Density for plotting and potential
    density = np.abs(opr.wfc) ** 2

    # Renormalization
    if par.im_time:
      renorm_factor = np.sum(density * par.dx)
      if renorm_factor != 0.0:
        opr.wfc /= np.sqrt(renorm_factor)
        density = np.abs(opr.wfc) ** 2
      else:
        db.checkpoint(debug=True, msg1=f"RENORMALIZATION WARNING! …", stop=False)

    # Saving for visualization (100 snapshots)
        …
```

RESULTS (1)

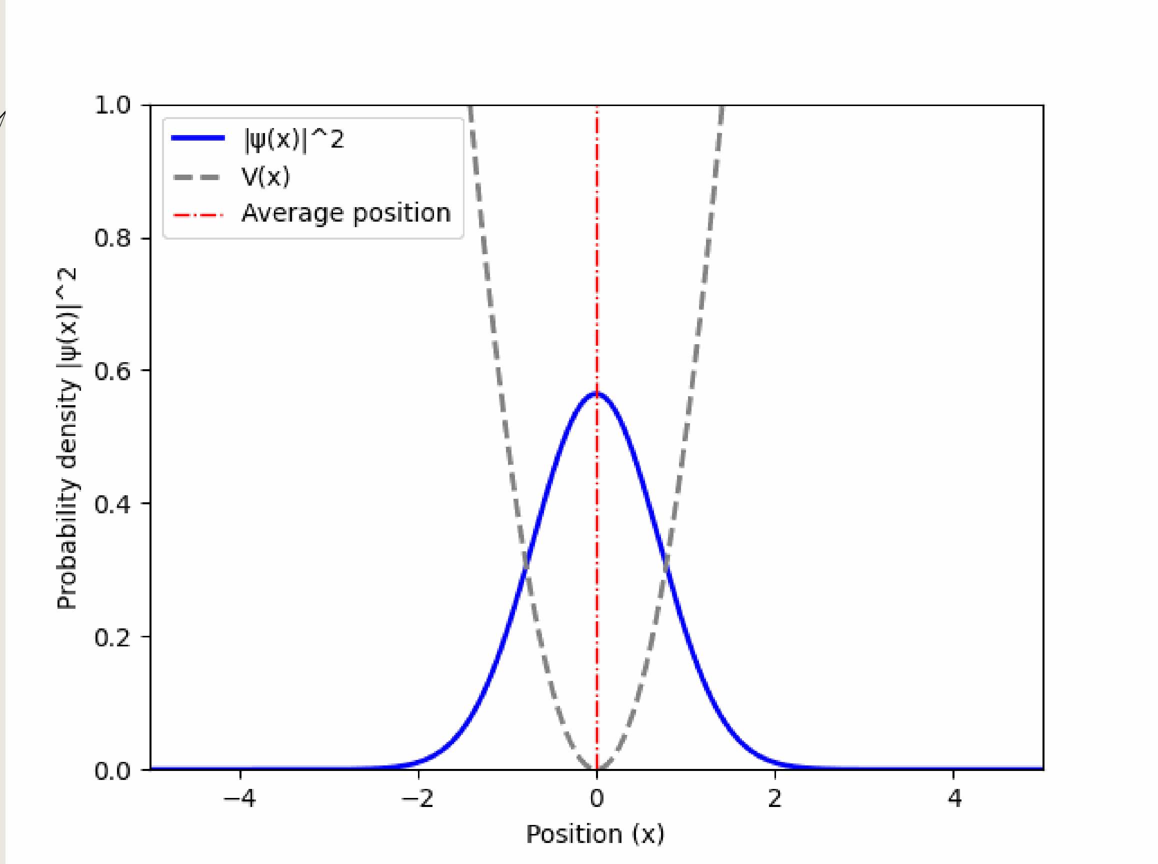


Real time evolution (T = 6.44)

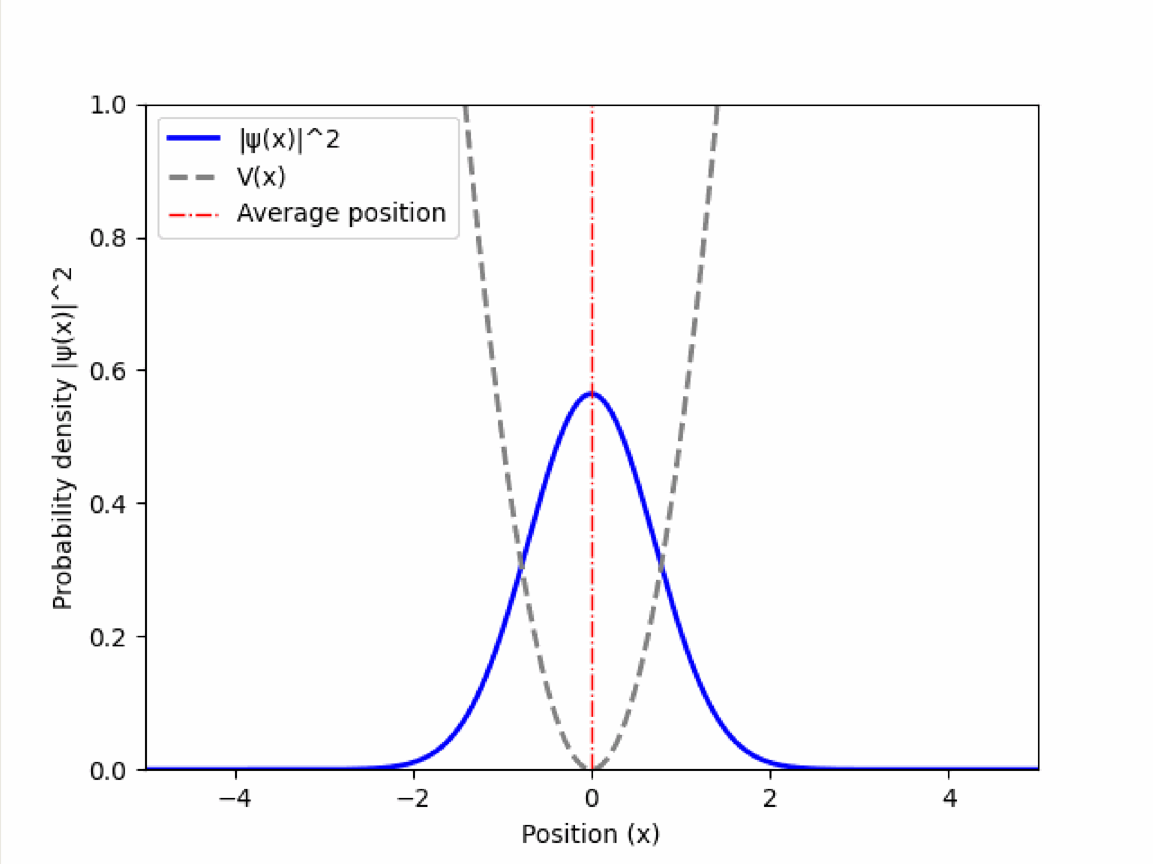


Imaginary time evolution (T = 6.44)

RESULTS (2)

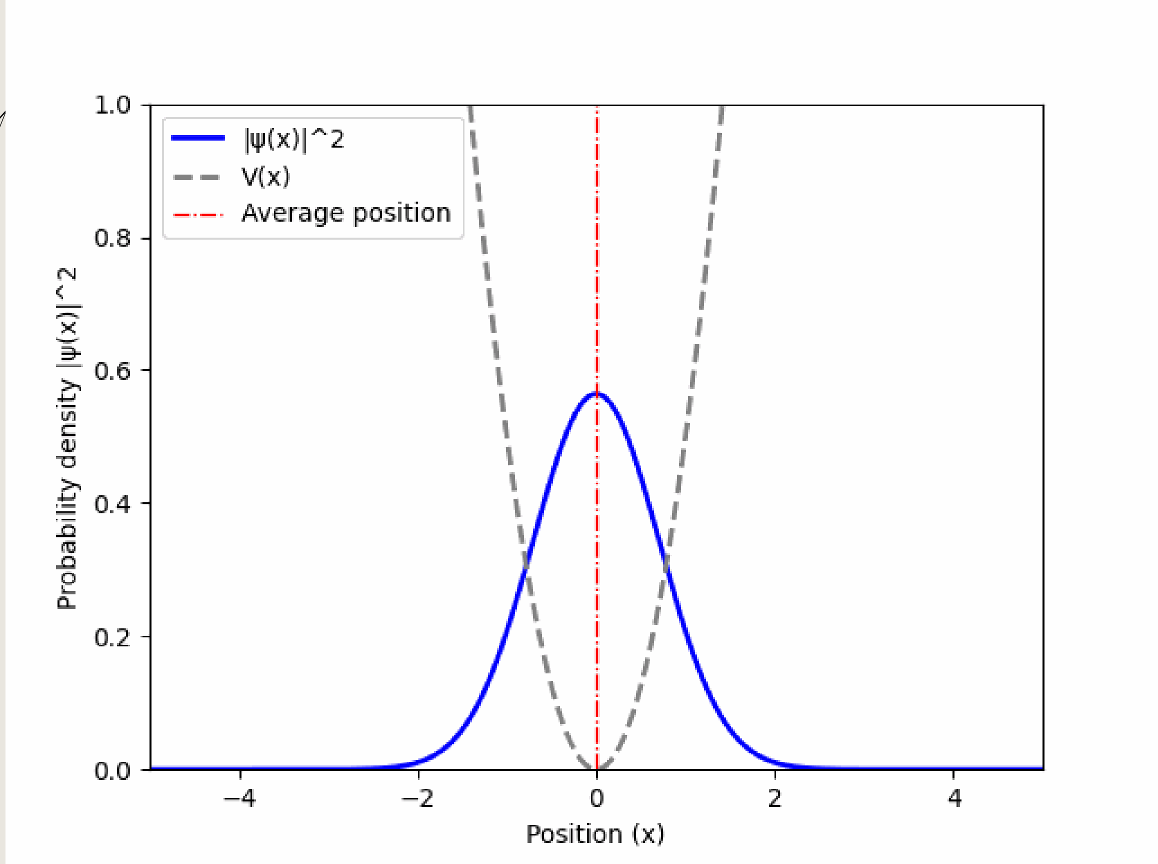


Real time evolution (T = 17.33)

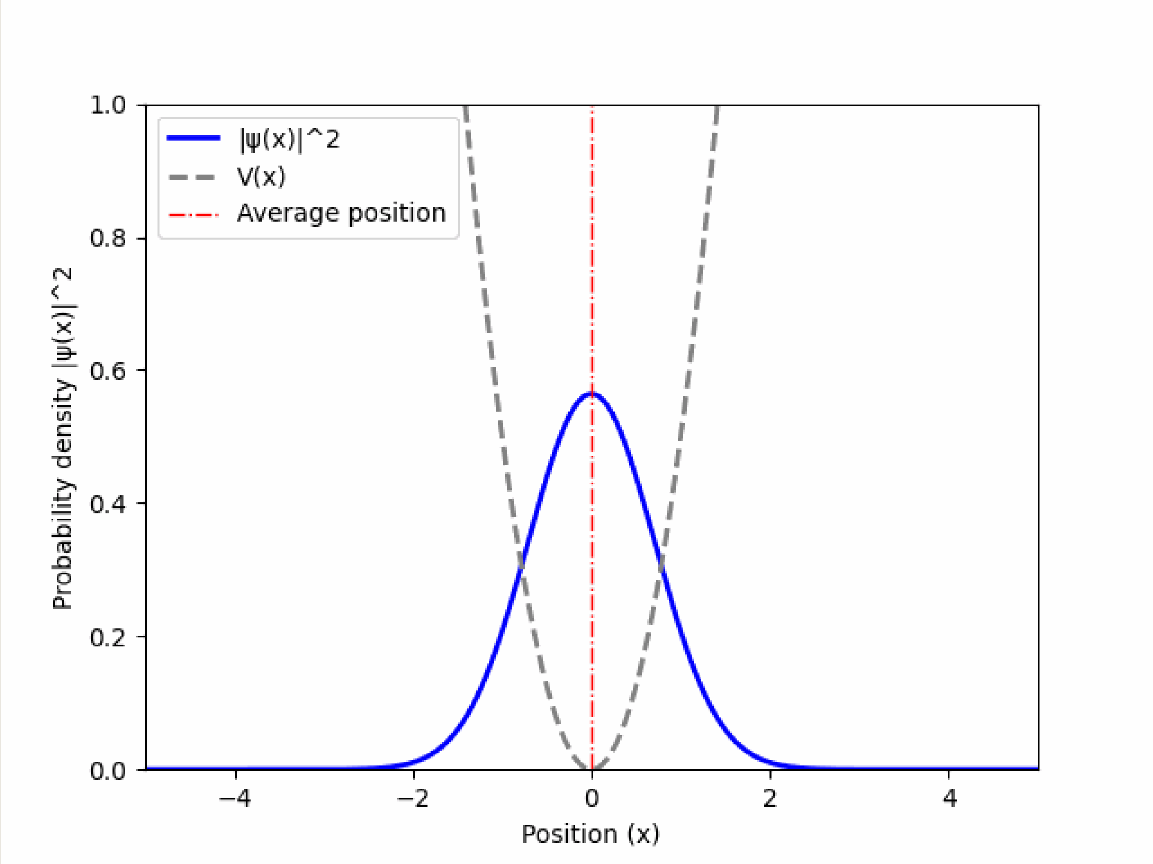


Imaginary time evolution (T = 17.33)

RESULTS (3)

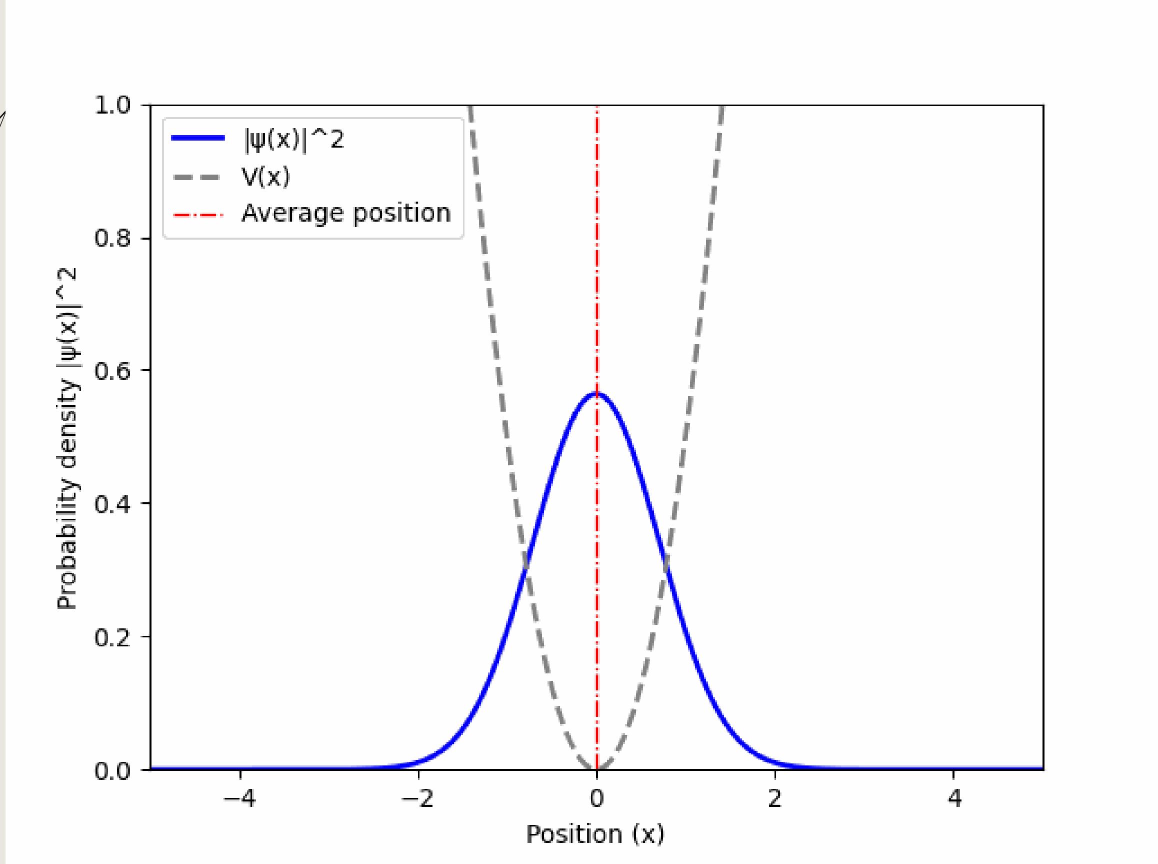


Real time evolution (T = 106.05)

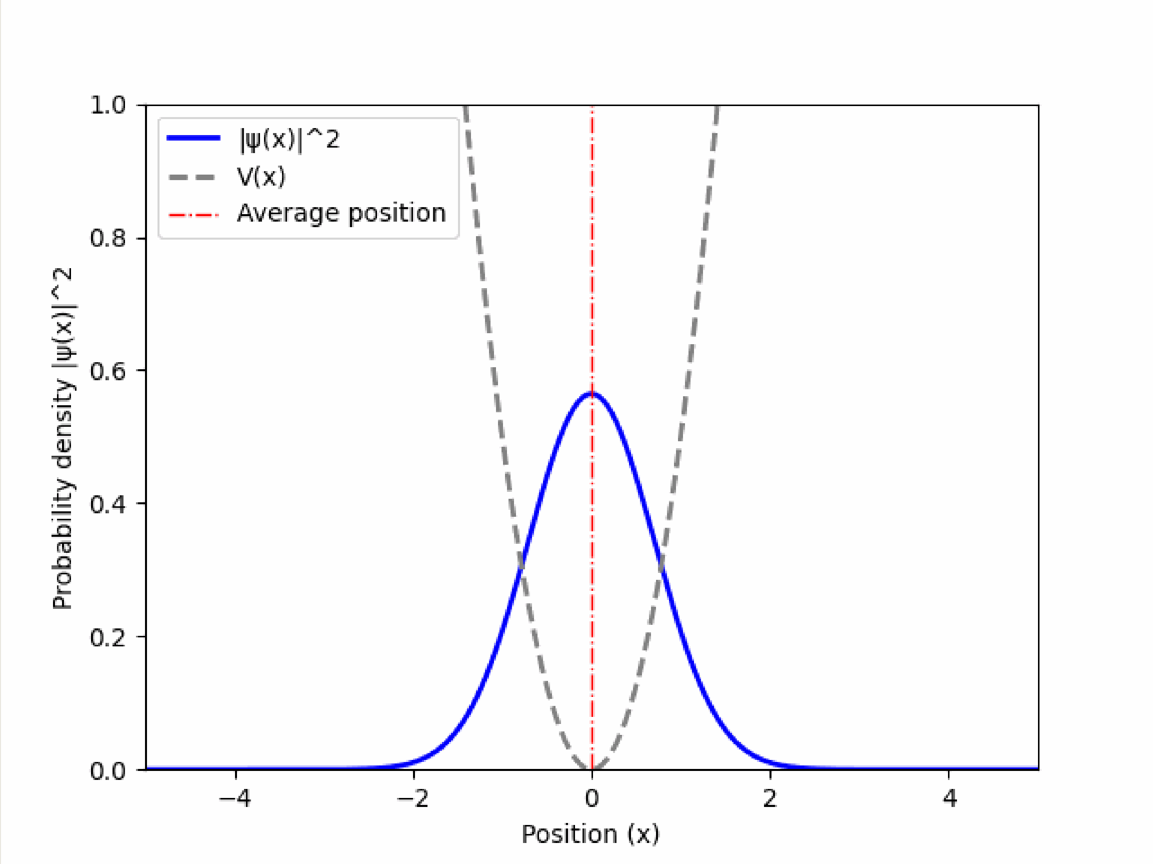


Imaginary time evolution (T = 106.05)

RESULTS (4)

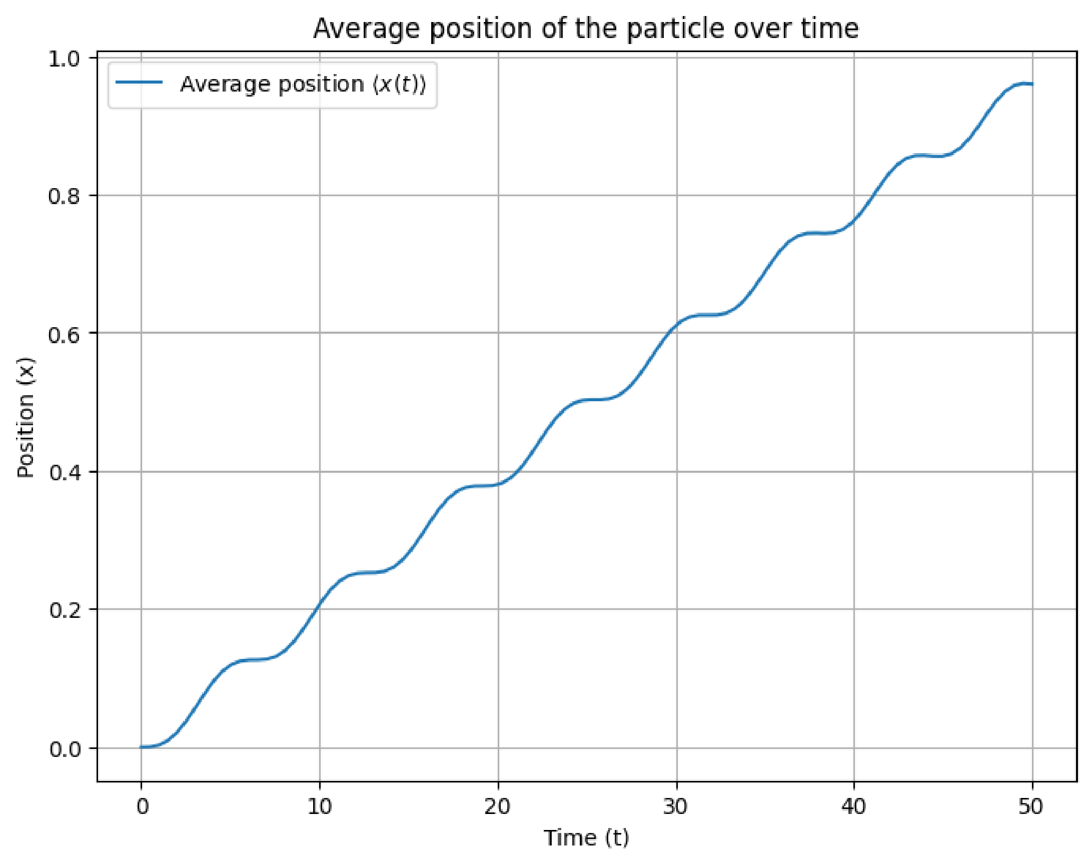


Real time evolution (T = 421.21)

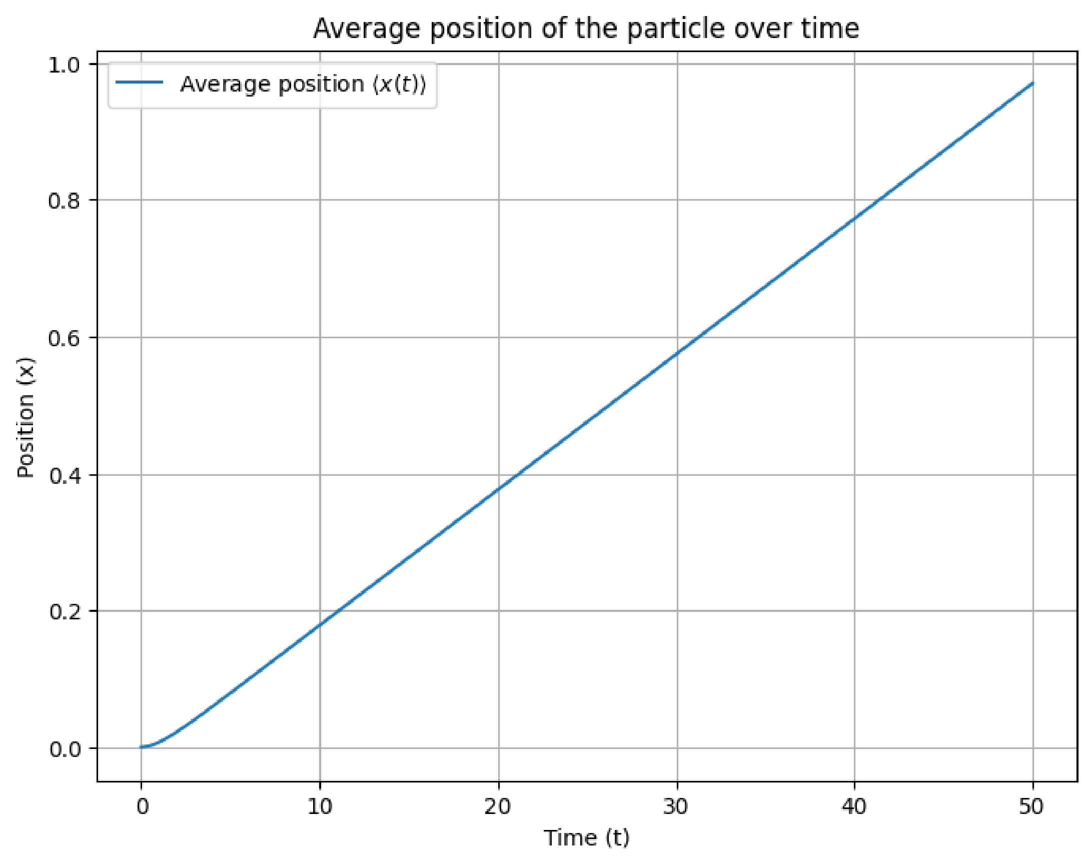


Imaginary time evolution (T = 421.21)

RESULTS (5)

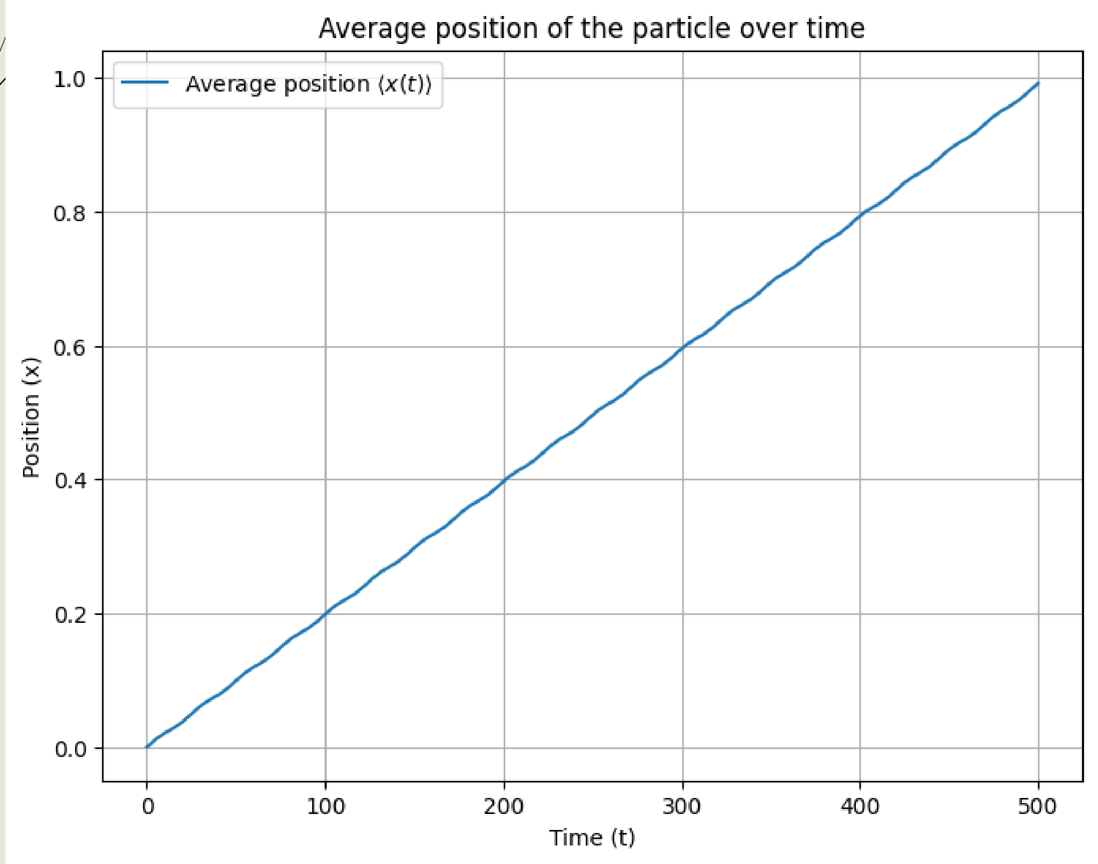


Real time (T = 50.00)

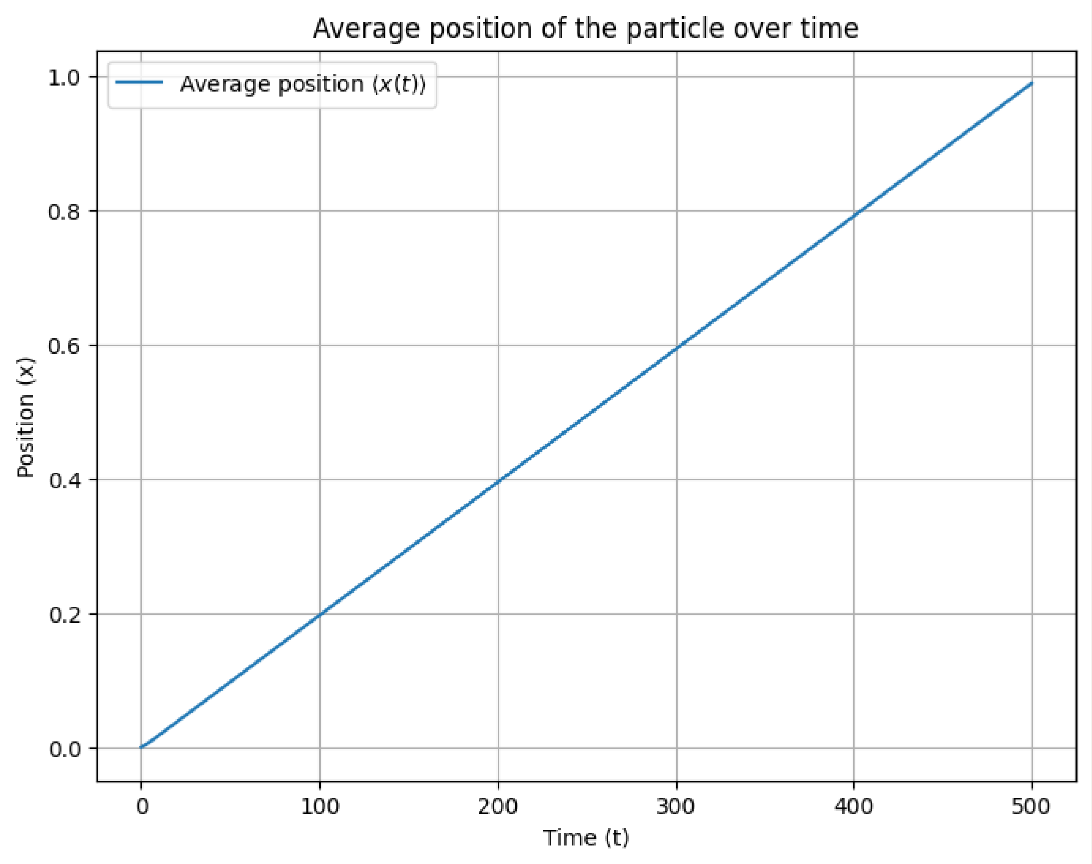


Imaginary time (T = 50.00)

RESULTS (6)

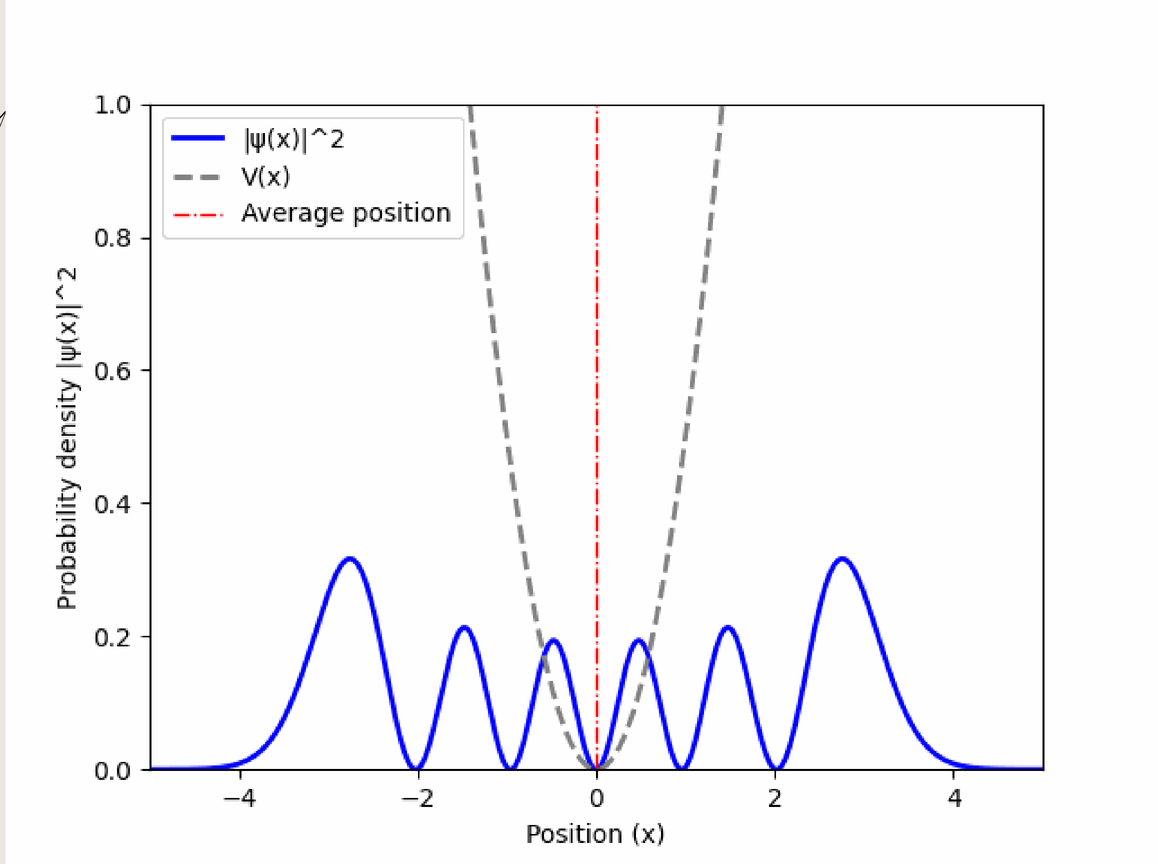


Real time (T = 50.00)

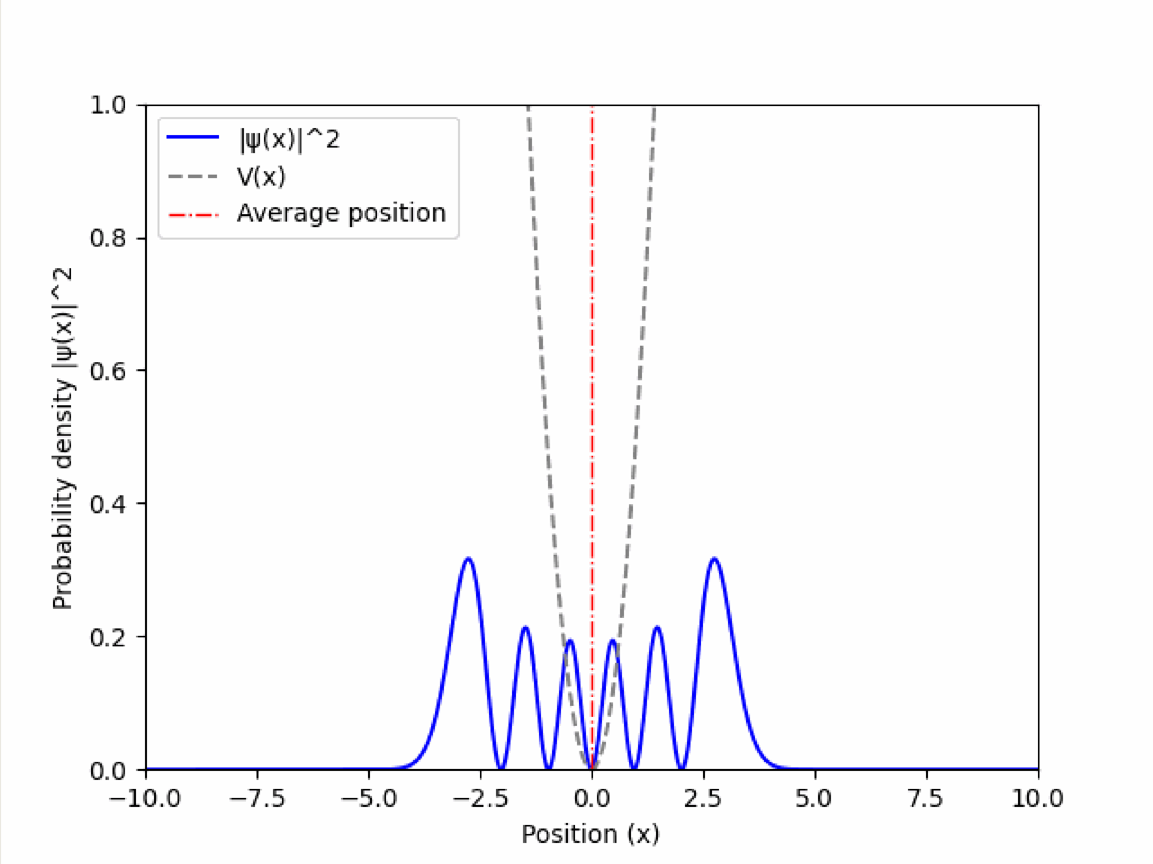


Imaginary time (T = 50.00)

RESULTS (7)

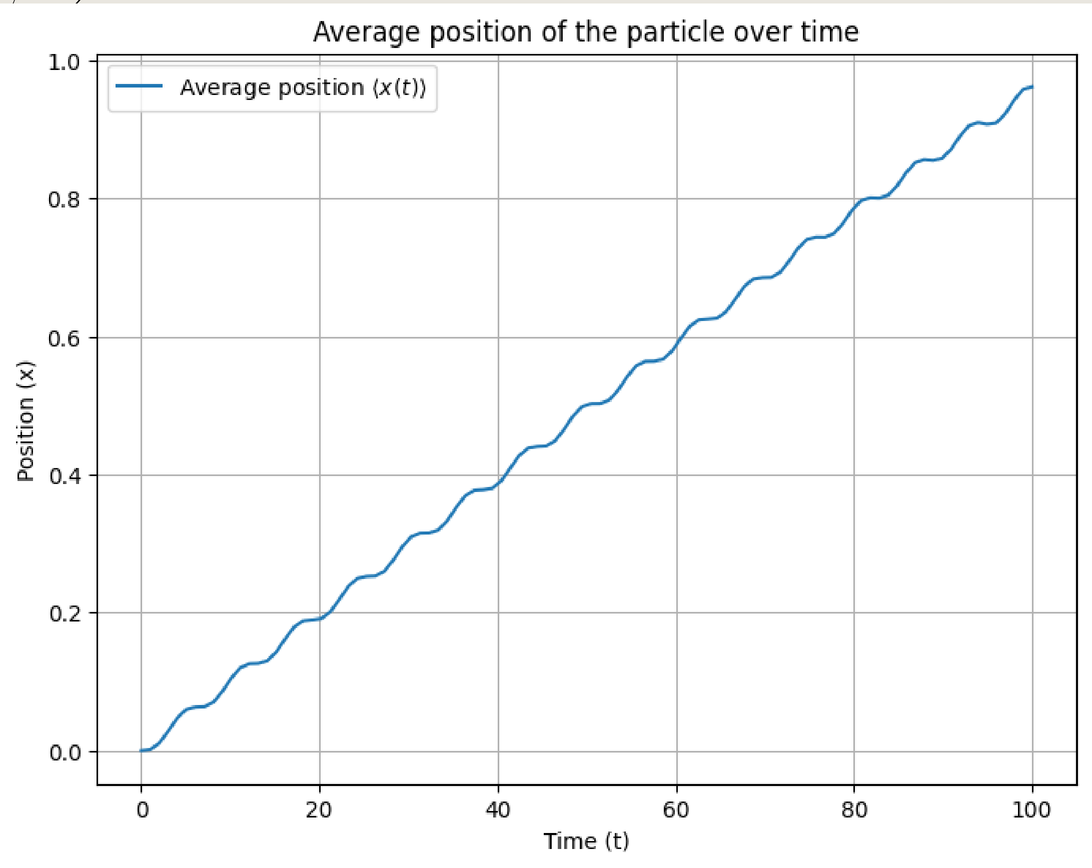


Real time evolution (n = 5)

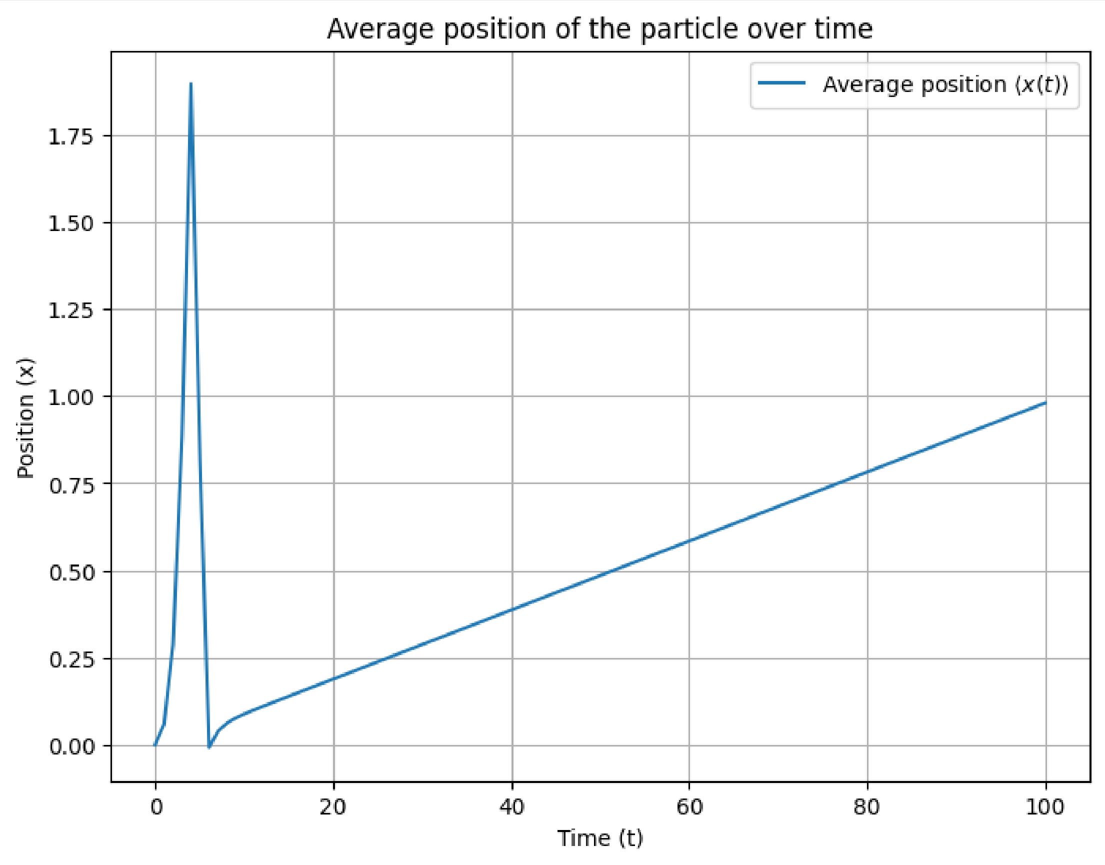


Imaginary time evolution (n = 5)

RESULTS (8)



Real time (n = 5)



Imaginary time (n = 5)

CONCLUSIONS

- **Real-Time Evolution:**
  - Models physical dynamics, with oscillatory behavior governed by energy eigenvalues (‘dies’ in adiabatic limit).

- **Imaginary-Time Evolution:**
  - Projects onto the ground state by exponential decay of higher-energy contributions (important for ground-state determination).

- **Split-Operator Method:**
  - Efficiently handles both real and imaginary-time evolution, relying on operator splitting for accuracy and simplicity.