Aim: To compute and visualize the probability of detecting a quantum particle, initially localized in a Gaussian wavepacket with zero average momentum, beyond a certain position as a function of time. Additionally, to estimate the time it would likely take for the particle to reach the detector based on classical considerations.

Objective:

- Calculate the expected squared momentum $\langle p^2 \rangle$ of the initial wavefunction.
- Estimate the classical time for the particle to reach the detector using $\langle p^2 \rangle$.
- Determine the time evolution of the wavefunction in free space.
- Compute the probability of detecting the particle beyond position *L* as a function of time.
- Visualize the detection probability over time and compare it with the classical estimation.

Summary: This project investigates the dynamics of a free quantum particle initially described by a Gaussian wavefunction centered at the origin with zero average momentum. Using both classical and quantum mechanical approaches, we estimate the time it would take for the particle to reach a detector positioned at a distance *L*. The classical estimation is derived from the expected momentum, while the quantum mechanical approach involves solving the time-dependent Schrödinger equation. By expanding the initial wavefunction in terms of momentum eigenstates, we obtain an analytical expression for its time evolution. Numerical integration techniques are then employed to calculate the detection probability beyond position *L* as a function of time. The results are visualized to highlight the differences between classical and quantum mechanical predictions.

Tools and Libraries Used:

- **NumPy:** Numerical computations and array operations.
- **SymPy:** Symbolic mathematics for integration and algebraic manipulations.
- **SciPy:** Numerical integration using nquad for multiple integrals.
- **Matplotlib:** Plotting and data visualization.

Procedure:

1. **Importing Libraries:**

CODE:

```
import numpy as np

import sympy as smp
```

```
from scipy.integrate import nquad

import matplotlib.pyplot as plt
```

2. **Defining Symbols:**
    - Define symbolic variables for position (*x*), characteristic length (*a*), detector position (*L*), mass (*m*), momentum variables (*k* and *k'*), and time (*t*).

CODE:

```
x = smp.symbols('x', real=True)

a, L, m, k, kp, t = smp.symbols('a L m k k\' t', real=True, positive=True)
```

3. **Normalizing the Wavefunction:**
    - Define the initial Gaussian wavefunction and normalize it over all space.

CODE:

```
psi = smp.exp(-x**2 / (2*a**2))

psi = psi / smp.sqrt(smp.integrate(psi**2, (x, -smp.oo, smp.oo)))
```

4. **Calculating $\langle p^2 \rangle$:**
    - Compute the expectation value of the squared momentum operator.

CODE:

```
p2 = smp.integrate(psi * (-smp.diff(psi, (x, 2))), (x, -smp.oo, smp.oo))
```

5. **Estimating Classical Time (*T*):**
    - Based on $\langle p^2 \rangle$, estimate the classical time for the particle to reach the detector.

CODE:

```
T = m * L / smp.sqrt(p2)
```

6. **Expressing Time Evolution:**
    - Expand the initial wavefunction in momentum eigenstates using Fourier transform.
    - Apply the free particle time evolution operator.

CODE:

```
psi_k = smp.exp(smp.I * k * x)

c_k = 1 / (2 * smp.pi) * smp.integrate(psi * psi_k.conjugate(), (x, -smp.oo, smp.oo)).simplify()
```

7. **Calculating Detection Probability:**
    - Derive the probability of finding the particle beyond position *L* at time *t*.

- Simplify the triple integral involved in the computation.

CODE:

```
c_kp_c = c_k.subs(k, kp).conjugate()

psi_kp_c = psi_k.subs(k, kp).conjugate()

r = smp.integrate(psi_k * psi_kp_c, (x, -L, L))

r = r.rewrite(smp.sin).simplify().rewrite(smp.sinc).simplify()

integrand = c_k * c_kp_c * r * smp.exp(smp.I * (k**2 - kp**2) * t / (2 * m))

integrand = smp.re(integrand).simplify()
```

8. **Defining Numerical Functions:**
    - Create numerical (lambdified) versions of the symbolic expressions for evaluation.

CODE:

```
integrand_f = smp.lambdify((k, kp, L, m, t), integrand)
```

9. **Performing Numerical Integration:**
    - Compute the detection probability at various times using nquad.

CODE:

```
def compute_prob(t, L, lim=np.inf):

  return 0.5 * (1 - nquad(integrand_f, ranges=[[-lim, lim], [-lim, lim]], args=(L, m, t), opts={'limit': 100})[0])
```

10. **Plotting Results:**
    - Visualize the detection probability over time and indicate the classical estimation time for comparison.

CODE:

```
t_values = [0, 1, 2, 3, 5, 8, 11, 15, 20, 30]

probabilities = [compute_prob(t, 5, lim=5) for t in t_values]


plt.plot(t_values, probabilities, 'ko--')

plt.axvline(5 * np.sqrt(2), color='r', linestyle='--', label='Classical Estimation')

plt.title('Probability of Observation Beyond $x=5a$')

plt.ylabel('Probability')
```

```
plt.xlabel('Time (t)')

plt.legend()

plt.show()
```

Highlights:

- **Symbolic Computations with SymPy:** Utilized for precise analytical calculations before numerical evaluations, ensuring accuracy and reducing computational load.
- **Time Evolution via Fourier Transform:** Demonstrated a fundamental quantum mechanics technique by expanding the initial wavefunction in momentum eigenstates and applying time evolution.
- **Analytical Simplification:** Prioritized simplifying complex integrals analytically to enhance computational efficiency during numerical integration.
- **Visualization:** Provided clear graphical representation of the detection probability over time, juxtaposed with the classical estimation, offering insights into the quantum behavior of the particle.

Conclusion: The project effectively combines classical and quantum mechanical methods to analyze the dynamics of a free quantum particle. While the classical estimation offers a rough guideline for the particle's arrival time at the detector, the quantum mechanical approach reveals a more nuanced probability distribution over time. The visualization underscores the inherent differences between classical predictions and quantum reality, emphasizing the importance of quantum mechanics in accurately describing particle behavior at microscopic scales.