Mixing Quantum Circuits with Classical Vector Dynamics

- **Classical Vectors**: In vector dynamics, we describe motion using position $\vec{r}$=(x,y,z)\vec{r} = (x, y, z)$\vec{r}$=(x,y,z), velocity $\vec{v}$=d$\vec{r}$dt\vec{v} = \frac{d\vec{r}}{dt}$\vec{v}$=dtdr, and acceleration $\vec{a}$=d$\vec{v}$dt\vec{a} = \frac{d\vec{v}}{dt}$\vec{a}$=dtdc. These quantities are essential for modeling objects like an airplane in motion.
- **Quantum Circuits**: Quantum circuits are composed of quantum gates (e.g., Hadamard or Pauli gates) and algorithms that transform qubits. For example:
    - A quantum rotation applies transformations akin to classical rotation matrices.
    - Superposition states created by Hadamard gates introduce bifurcations or reflections in quantum space.
- **Combining Both**: Using quantum transformation matrices, we can modify classical vectors in three-dimensional or tetrahedral (4D) space. For example:
    - An airplane's trajectory can be mapped to a quantum state $|\psi\rangle$=α|0⟩+β|1⟩|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$|\psi\rangle$=α|0⟩+β|1⟩, where the amplitudes α,β\alpha, \betaα,β relate to probabilities associated with positions or velocities.

---

4. Python

```
import numpy as np
import matplotlib.pyplot as plt

# 1. Classical vectors for airplane dynamics
position = np.array([3, 1, 0])  # Initial position (x, y, z)
velocity = np.array([2, 3, 0])  # Initial velocity
acceleration = np.array([0, 0, -9.8])  # Gravitational acceleration

# 2. Time simulation
t = np.linspace(0, 10, 100)  # Time from 0 to 10 seconds
positions = position + np.outer(t, velocity) + 0.5 * np.outer(t**2,
acceleration)

# 3. Quantum transformation with rotation gate
theta = np.pi / 4  # Rotation angle
rotation_matrix = np.array([
    [np.cos(theta), -np.sin(theta), 0],
    [np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
```

```
])

quantum_transformed_positions = positions @ rotation_matrix.T

# 4. Plot results
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(positions[:, 0], positions[:, 1], positions[:, 2],
label='Classical Trajectory')
ax.plot(quantum_transformed_positions[:, 0],
quantum_transformed_positions[:, 1],
        quantum_transformed_positions[:, 2], '--', label='Quantum
Transformed Trajectory')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.title('Airplane Dynamics with Quantum Transformations')
plt.show()
```