class QuantumGravityIntegration:

def \_\_init\_\_(self):

self.planck\_length = 1.616255e-35 # meters

self.planck\_time = 5.391247e-44 # seconds

def compute\_holographic\_dynamics(self, bulk\_state, boundary\_state):

"""

Implement holographic principle in temporal flow theory

"""

# AdS/CFT correspondence for temporal flow

bulk\_dynamics = self.bulk\_evolution(bulk\_state)

boundary\_dynamics = self.boundary\_evolution(boundary\_state)

# Ensure holographic consistency

assert self.check\_holographic\_consistency(bulk\_dynamics, boundary\_dynamics)

return {

'bulk': bulk\_dynamics,

'boundary': boundary\_dynamics,

'correspondence\_map': self.compute\_correspondence\_map()

}

def planck\_scale\_behavior(self, state, scale):

"""

Handle Planck scale physics with discrete spacetime

"""

if scale <= self.planck\_length:

# Implement discrete spacetime dynamics

return self.discrete\_evolution(state)

else:

# Smooth transition to continuous dynamics

return self.continuous\_evolution(state)

def discrete\_evolution(self, state):

"""

Evolution equations for discrete spacetime

"""

# Causal dynamical triangulations approach

triangulation = self.build\_spacetime\_triangulation(state)

# Discrete temporal flow

discrete\_flow = self.compute\_discrete\_flow(triangulation)

return discrete\_flow