class ObservationalPredictions:

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

self.cosmological\_params = None

self.quantum\_params = None

def compute\_predictions(self, scale):

"""

Compute scale-dependent predictions for various observables

"""

if scale == 'quantum':

return self.quantum\_predictions()

elif scale == 'classical':

return self.classical\_predictions()

elif scale == 'cosmological':

return self.cosmological\_predictions()

def quantum\_predictions(self):

"""

Quantum scale predictions including interference modifications

"""

# Modified interference pattern

I = lambda x: I0 \* (1 + np.cos(k\*x)) \* (1 + mu \* g(r) \* abs(W)\*\*2)

# Enhanced entanglement correlations

C = lambda r1, r2: C0 \* np.exp(-r/xi) \* (1 + kappa \* abs(W)\*\*2)

return {'interference': I, 'correlations': C}

def classical\_predictions(self):

"""

Classical scale predictions

"""

# Modified gravitational potential

Phi = lambda r: -G\*M/r \* (1 + alpha \* g(r) \* abs(W)\*\*2)

# Enhanced frame dragging

omega = lambda r: omega\_GR \* (1 + gamma \* g(r) \* abs(W)\*\*2)

return {'potential': Phi, 'frame\_dragging': omega}

def cosmological\_predictions(self):

"""

Cosmological scale predictions

"""

# Dark matter distribution

rho\_DM = lambda r: rho0 \* (1 + f\_DM(r) \* abs(W)\*\*2)

# Dark energy density

rho\_DE = lambda r: Lambda0 \* (1 + h\_DE(r) \* abs(W)\*\*2)

return {'dark\_matter': rho\_DM, 'dark\_energy': rho\_DE}

# Generate predictions

predictions = ObservationalPredictions()

quantum\_effects = predictions.compute\_predictions('quantum')

cosmological\_effects = predictions.compute\_predictions('cosmological')