"""

Multi-Scale Integration for Temporal Flow Theory

==============================================

"""

class MultiScaleIntegrator:

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

self.scales = scales

self.integrators = {}

self.setup\_integrators()

def setup\_integrators(self):

"""

Initialize appropriate integrators for each scale

"""

for scale in self.scales:

if scale == 'quantum':

self.integrators[scale] = QuantumScaleIntegrator()

elif scale == 'classical':

self.integrators[scale] = ClassicalScaleIntegrator()

elif scale == 'cosmological':

self.integrators[scale] = CosmologicalScaleIntegrator()

def integrate\_scales(self, state):

"""

Perform multi-scale integration

"""

results = {}

for scale, integrator in self.integrators.items():

results[scale] = integrator.evolve(state[scale])

return self.combine\_scale\_results(results)

class QuantumScaleIntegrator:

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

self.h\_bar = 1.0545718e-34 # Reduced Planck constant

def evolve(self, quantum\_state):

"""

Evolve quantum scale physics

"""

# Split-operator method for quantum evolution

psi\_k = np.fft.fft(quantum\_state)

T = self.kinetic\_evolution(psi\_k)

V = self.potential\_evolution(quantum\_state)

return self.combine\_evolution(T, V)

class ClassicalScaleIntegrator:

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

self.G = 6.67430e-11 # Gravitational constant

def evolve(self, classical\_state):

"""

Evolve classical scale physics

"""

# Symplectic integration for classical dynamics

return self.symplectic\_integrator(classical\_state)

class CosmologicalScaleIntegrator:

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

self.H0 = 70.0 # Hubble constant

def evolve(self, cosmological\_state):

"""

Evolve cosmological scale physics

"""

# N-body method for cosmological evolution

return self.nbody\_integrator(cosmological\_state)

class ScaleCoupling:

"""

Handle coupling between different scales

"""

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

self.coupling\_strengths = {}

def compute\_coupling(self, scale1, scale2, state):

"""

Compute coupling between different scales

"""

if (scale1, scale2) not in self.coupling\_strengths:

self.coupling\_strengths[(scale1, scale2)] = self.estimate\_coupling(scale1, scale2)

return self.coupling\_strengths[(scale1, scale2)] \* self.coupling\_function(state)

"""

Usage Example:

-------------

# Initialize multi-scale integration

scales = ['quantum', 'classical', 'cosmological']

integrator = MultiScaleIntegrator(scales)

# Prepare initial state

initial\_state = {

'quantum': quantum\_initial\_state,

'classical': classical\_initial\_state,

'cosmological': cosmological\_initial\_state

}

# Perform integration

result = integrator.integrate\_scales(initial\_state)

"""