# MBT Curvature Capture Simulation – Mass from Motion (Colab Ready)

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

# Simulation parameters

grid\_size = 150

timesteps = 200

dt = 0.1

decay = 0.995

# Initialize fields

field = np.zeros((grid\_size, grid\_size))

velocity = np.zeros\_like(field)

# Create a central negative tension basin (MBT-style attractor)

x = np.linspace(-1, 1, grid\_size)

y = np.linspace(-1, 1, grid\_size)

xx, yy = np.meshgrid(x, y)

r = np.sqrt(xx\*\*2 + yy\*\*2)

curvature\_well = -np.exp(-20 \* r\*\*2) # Sharp tension at center

# Inject moving energy pulse

pulse = np.exp(-100 \* ((xx + 0.4)\*\*2 + yy\*\*2))

velocity += pulse \* 1.5

# Visualization setup

fig, ax = plt.subplots(figsize=(6, 6))

im = ax.imshow(field, cmap='coolwarm', vmin=-1, vmax=1)

ax.set\_title("MBT: Curvature Capture (Mass from Motion)")

plt.axis('off')

def laplacian(data):

return (

-4 \* data +

np.roll(data, 1, axis=0) + np.roll(data, -1, axis=0) +

np.roll(data, 1, axis=1) + np.roll(data, -1, axis=1)

)

# Update function for animation

def update(frame):

global field, velocity

resist = 1 + curvature\_well \* 5 + np.abs(field) \* 2

v\_lap = laplacian(field) \* dt / resist

velocity = velocity \* decay + v\_lap

field += velocity \* dt

im.set\_array(field)

return [im]

ani = FuncAnimation(fig, update, frames=timesteps, interval=50, blit=True)

plt.tight\_layout()

plt.show()