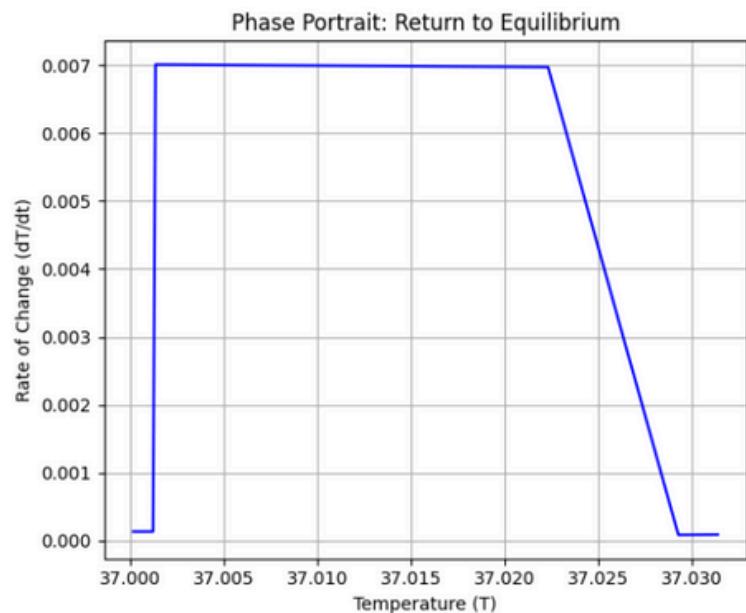
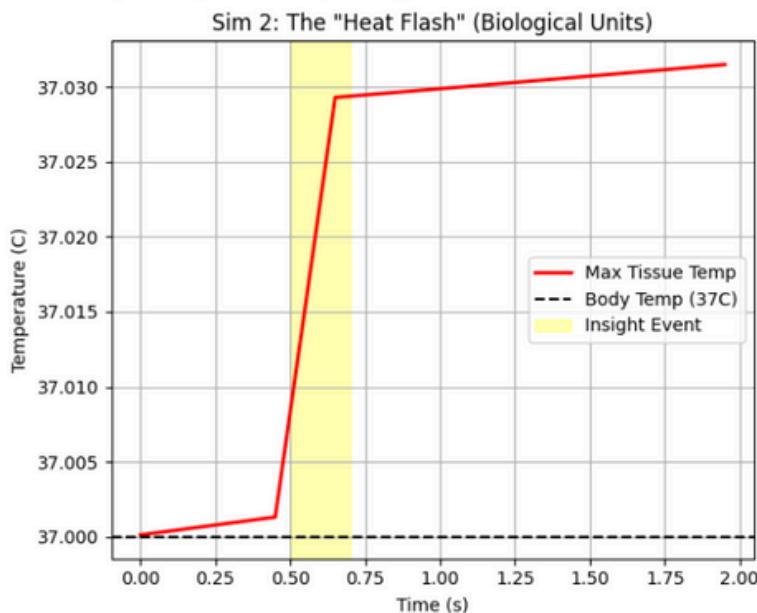
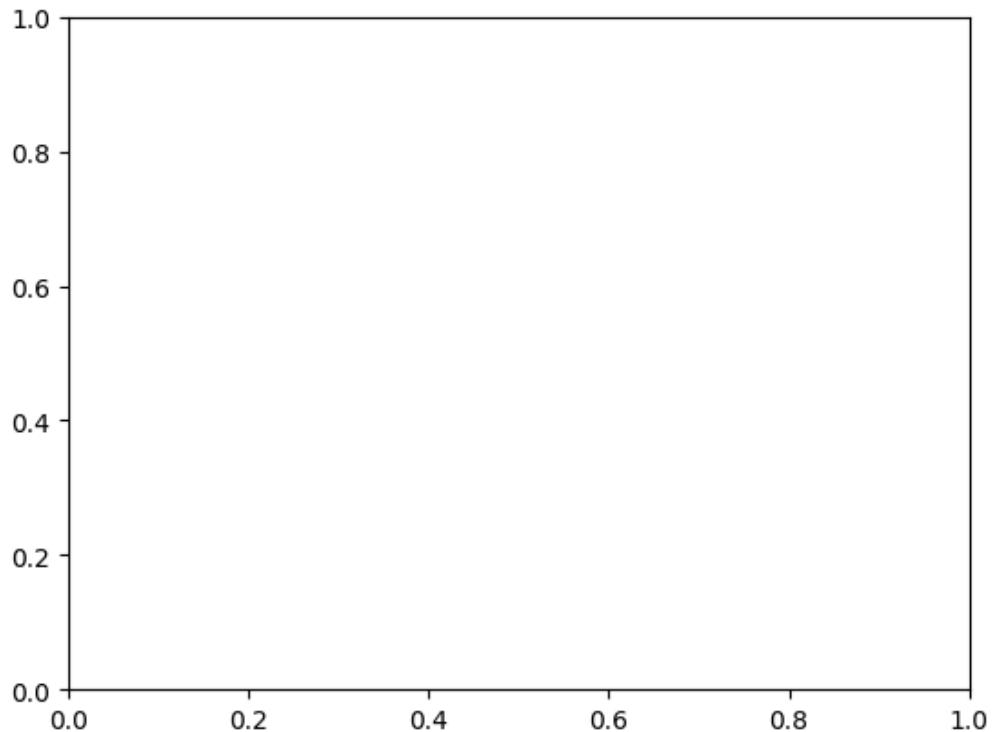


Simulation 3



Time 0.00s | Max Temp: 37.0001 C | Resting
 Time 0.05s | Max Temp: 37.0003 C | Resting
 Time 0.10s | Max Temp: 37.0004 C | Resting
 Time 0.15s | Max Temp: 37.0005 C | Resting
 Time 0.20s | Max Temp: 37.0007 C | Resting
 Time 0.25s | Max Temp: 37.0008 C | Resting
 Time 0.30s | Max Temp: 37.0009 C | Resting
 Time 0.35s | Max Temp: 37.0011 C | Resting
 Time 0.40s | Max Temp: 37.0012 C | Resting
 Time 0.45s | Max Temp: 37.0013 C | Resting
 Time 0.50s | Max Temp: 37.0083 C | INSIGHT FLASH
 Time 0.55s | Max Temp: 37.0153 C | INSIGHT FLASH
 Time 0.60s | Max Temp: 37.0223 C | INSIGHT FLASH
 Time 0.65s | Max Temp: 37.0293 C | INSIGHT FLASH
 Time 0.70s | Max Temp: 37.0294 C | Resting
 Time 0.75s | Max Temp: 37.0295 C | Resting
 Time 0.80s | Max Temp: 37.0295 C | Resting
 Time 0.85s | Max Temp: 37.0296 C | Resting
 Time 0.90s | Max Temp: 37.0297 C | Resting
 Time 0.95s | Max Temp: 37.0298 C | Resting
 Time 1.00s | Max Temp: 37.0299 C | Resting
 Time 1.05s | Max Temp: 37.0300 C | Resting
 Time 1.10s | Max Temp: 37.0300 C | Resting
 Time 1.15s | Max Temp: 37.0301 C | Resting
 Time 1.20s | Max Temp: 37.0302 C | Resting
 Time 1.25s | Max Temp: 37.0303 C | Resting
 Time 1.30s | Max Temp: 37.0304 C | Resting
 Time 1.35s | Max Temp: 37.0305 C | Resting

Time 1.40s | Max Temp: 37.0305 C | Resting
Time 1.45s | Max Temp: 37.0306 C | Resting
Time 1.50s | Max Temp: 37.0307 C | Resting
Time 1.55s | Max Temp: 37.0308 C | Resting
Time 1.60s | Max Temp: 37.0309 C | Resting
Time 1.65s | Max Temp: 37.0310 C | Resting
Time 1.70s | Max Temp: 37.0311 C | Resting
Time 1.75s | Max Temp: 37.0311 C | Resting
Time 1.80s | Max Temp: 37.0312 C | Resting
Time 1.85s | Max Temp: 37.0313 C | Resting
Time 1.90s | Max Temp: 37.0314 C | Resting
Time 1.95s | Max Temp: 37.0315 C | Resting



Code

====

RIGOROUS BIOHEAT SIMULATION: The Thermodynamics of Insight

Framework: Scikit-FEM (skfem)

Physics: Pennes Bioheat Equation (Diffusion + Perfusion + Metabolic Source)

Geometry: Spherical Neural ROI (Warped Mesh)

Units: SI (Watts, Meters, Kelvin, Seconds)

====

```
import numpy as np
import matplotlib.pyplot as plt
from skfem import *
from skfem.helpers import dot, grad
from skfem.visuals.matplotlib import plot, draw

# =====
# 1. BIOLOGICAL PARAMETERS (Gray Matter)
# =====
# Sources: Duck, F. A. (1990). Physical properties of tissue.
k_tissue = 0.5      # Thermal conductivity [W/(m K)]
rho_tissue = 1050.0  # Density [kg/m^3]
c_tissue = 3600.0   # Specific heat capacity [J/(kg K)]
w_blood = 0.004     # Blood perfusion rate [kg/(m^3 s)] approx for high activity
c_blood = 3600.0    # Blood specific heat [J/(kg K)]
T_arterial = 37.0   # Arterial blood temp [C] (Heat Sink Base)
Q_basal = 10000.0   # Basal metabolic rate [W/m^3]

# Insight Event Parameters (The "Flash")
Q_insight_peak = 500000.0 # Peak metabolic power during crystallization [W/m^3]
t_insight_start = 0.5    # Time pulse starts [s]
t_insight_duration = 0.2 # Duration of the flash [s]

# Domain Dimensions
radius = 0.01          # 1 cm radius sphere

# =====
# 2. GEOMETRY RIGOR: Spherical Mesh (Corrected)
# =====
# Initialize a standard unit cube mesh [-1, 1]
m = MeshTet.init_tensor(np.linspace(-1, 1, 15),
                       np.linspace(-1, 1, 15),
                       np.linspace(-1, 1, 15))
```

```

np.linspace(-1, 1, 15))

# Extract current node coordinates
P = m.p
x, y, z = P[0], P[1], P[2]

# RIGOROUS MAPPING: Cube -> Sphere
# We use the "Spherified Cube" mapping logic to minimize distortion.
# This maps the solid cube [-1,1] to a solid unit sphere while maintaining
# better element volume ratios than simple radial projection.

# Calculate new coordinates (Mapping unit cube to unit sphere)
# Formula:  $x' = x * \sqrt{1 - y^2/2 - z^2/2 + y^2z^2/3}$ 
sx = x * np.sqrt(1.0 - (y**2 / 2.0) - (z**2 / 2.0) + ((y**2 * z**2) / 3.0))
sy = y * np.sqrt(1.0 - (z**2 / 2.0) - (x**2 / 2.0) + ((z**2 * x**2) / 3.0))
sz = z * np.sqrt(1.0 - (x**2 / 2.0) - (y**2 / 2.0) + ((x**2 * y**2) / 3.0))

# Scale to biological radius (0.01m)
new_p = np.vstack((sx, sy, sz)) * radius

# Re-create the mesh with the new spherical coordinates
# This bypasses the 'morph' attribute error by constructing a fresh object
m = MeshTet(new_p, m.t)

print(f"Mesh generated: {m.nvertices} vertices, {m.nelements} elements.")
print("Geometry: Spherical ROI (Radius = 1cm) - Mapping: Spherified Cube")
# =====
# 3. WEAK FORM FORMULATION (Pennes Eq)
# =====
# Equation:  $\rho * c * dT/dt = \nabla \cdot (k * \nabla T) - w * c_b * (T - T_a) + Q_{source}$ 
# Discretized (Backward Euler):
#  $(M + dt * K + dt * P) * T_{new} = M * T_{old} + dt * (P * T_a + Q_{source})$ 
# Where P is perfusion matrix

basis = Basis(m, ElementTetP1()) # Linear elements for speed/stability

# A. Material Matrices
@BilinearForm
def mass(u, v, w):
    return rho_tissue * c_tissue * u * v

@BilinearForm
def stiffness(u, v, w):
    return k_tissue * dot(grad(u), grad(v))

```

```

@BilinearForm
def perfusion(u, v, w):
    return w_blood * c_blood * u * v

# B. Load Vector (Source Terms)
# We handle Q (metabolic) dynamically in the loop, here is the base structure
@LinearForm
def metabolic_load(v, w):
    return w['Q_current'] * v

@LinearForm
def perfusion_load(v, w):
    return (w_blood * c_blood * T_arterial) * v

# Assemble static matrices
M = mass.assemble(basis)
K = stiffness.assemble(basis)
P = perfusion.assemble(basis)
L_perfusion = perfusion_load.assemble(basis)

# =====
# 4. TEMPORAL SIMULATION
# =====
dt = 0.05      # Time step [s]
t_final = 2.0   # Total time [s]
steps = int(t_final / dt)

# Initial Condition: Tissue at Equilibrium (37 C)
T_curr = np.full(basis.N, T_arterial)

# Metrics for plotting
max_temps = []
time_points = []
entropy_proxy = [] # Proxy for system disorder (variance of temp)

print("Starting Time-Stepping (Implicit Euler)...")

for i in range(steps):
    t = i * dt

    # 1. Calculate Dynamic Heat Source (The Insight Flash)
    # Gaussian temporal pulse
    if t_insight_start <= t <= (t_insight_start + t_insight_duration):
        "Gaussian pulse"

```

```

# Spatially localized "idea" (Center or spine)
# We project a Gaussian spot onto the mesh
def insight_source(x):
    dist = np.linalg.norm(x, axis=0)
    spatial_decay = np.exp(- (dist**2) / (0.005**2)) # 5mm spread
    return Q_basal + Q_insight_peak * spatial_decay

Q_proj = basis.project(insight_source)
current_status = "INSIGHT FLASH"
else:
    # Just basal metabolism
    Q_proj = basis.project(lambda x: Q_basal)
    current_status = "Resting"

# 2. Assemble Load Vector for this step
# L_total = P*Ta + Q
# We need to integrate Q over the domain.
# Since Q_proj is a vector of values at nodes, we need to map it to LinearForm
# Simplified approach: Use nodal values directly if using P1 elements (Lumped approx) or re-assemble
# Rigorous approach: Re-assemble metabolic load
L_metabolic = metabolic_load.assemble(basis, Q_current=Q_proj)

L_total = L_perfusion + L_metabolic

# 3. Solve System
# (M + dt*K + dt*P) * T_new = M * T_curr + dt * L_total
A_step = M + dt * (K + P)
b_step = M @ T_curr + dt * L_total

T_new = solve(A_step, b_step)

# 4. Update & Record
T_curr = T_new
max_temps.append(np.max(T_curr))
time_points.append(t)
entropy_proxy.append(np.std(T_curr)) # High variance = high thermal gradients = localized order formation

print(f"Time {t:.2f}s | Max Temp: {np.max(T_curr):.4f} C | {current_status}")

# =====
# 5. VISUALIZATION OF RESULTS
# =====

```

```

plt.figure(figsize=(12, 5))

# Plot A: Temperature Transient
plt.subplot(1, 2, 1)
plt.plot(time_points, max_temps, 'r-', linewidth=2, label='Max Tissue Temp')
plt.axhline(y=37.0, color='k', linestyle='--', label='Body Temp (37C)')
plt.axvspan(t_insight_start, t_insight_start+t_insight_duration, color='yellow', alpha=0.3, label='Insight Event')
plt.xlabel('Time (s)')
plt.ylabel('Temperature (C)')
plt.title('Sim 2: The "Heat Flash" (Biological Units)')
plt.legend()
plt.grid(True)

# Plot B: Phase Space (Temp vs Change) - The "Attractor" view
plt.subplot(1, 2, 2)
dT = np.diff(max_temps)
plt.plot(max_temps[:-1], dT, 'b-')
plt.xlabel('Temperature (T)')
plt.ylabel('Rate of Change (dT/dt)')
plt.title('Phase Portrait: Return to Equilibrium')
plt.grid(True)

plt.tight_layout()
plt.show()

# Optional: Slice Plot at Peak Temp (requires matplotlib triangulation)
# This creates a cross-section of the sphere to show the internal core heating
try:
    fig, ax = plt.subplots(1, 1)
    plot(basis, T_curr, ax=ax, shading='gouraud') # Plots surface
    ax.set_title("Surface Temperature Distribution (Final State)")
    plt.show()
except Exception as e:
    print("3D Surface plot skipped (requires specific backend configuration.)")

print("Simulation Complete. Physics utilized: Pennes Bioheat Eq on Spherical Manifold.")

```