# NUCE 2101: Exam 2

PI: Dane A. Sabo
dane.sabo@pitt.edu

Monday 10th November, 2025

# Problem 1

## System Description

A two-loop reactor system with:

- Reactor (lumped hot leg) at temperature $T_{hot}$
- Steam Generator 1 with cold leg at $T_{cold1}$
- Steam Generator 2 with cold leg at $T_{cold2}$
- Equal mass flow rates and water masses in each loop
- Reactor Water Mass Fraction (RWMF): $\mu$

**Given parameters:**

- Base heat capacity: $C_0 = 33.33$ %-sec/F
- Base time constant: $\tau_0 = 0.75 \times C_0 = 25.00$ sec
- Initial temperature: $T_{hot} = T_{cold1} = T_{cold2} = 450$ F

**Derived parameter:**

The flow heat capacity rate per loop is *not directly given* in the problem statement but can be derived from the relationship between time constants and heat capacities:

$$W = \frac{C_0}{2\tau_0} = \frac{33.33}{2 \times 25.00} = 0.6667 \text{ \%-sec/F}$$

This represents the (mass flow rate $\times$ specific heat) for each loop.

**System parameters as functions of $\mu$:**

- Reactor time constant: $\tau_r = \mu\tau_0$
- Reactor heat capacity: $C_r = \mu C_0$
- Steam generator time constant (each): $\tau_{sg} = \frac{(1-\mu)\tau_r}{2}$
- Steam generator heat capacity (each): $C_{sg} = \frac{(1-\mu)C_0}{2}$

## Part A

## Python Code

```
import numpy as np
import sympy as sm

# Given parameters
C_0 = 33.33  # Base heat capacity [%-sec/degF]
tau_0 = 25.00  # Base time constant [sec]
W = C_0 / (2 * tau_0)  # Flow heat capacity rate per loop

def calculate_system_parameters(mu):
    C_r = mu * C_0  # Reactor heat capacity
    C_sg = (1 - mu) * C_0 / 2  # Steam generator heat capacity (each
        )
    return C_r, C_sg
```

```
13
14 # Matrix form: dT/dt = A*T + B
15 def get_matrix_A(C_r, C_sg, W):
16     A = np.array([
17         [-2*W/C_r,   W/C_r,       W/C_r],
18         [W/C_sg,     -W/C_sg,     0],
19         [W/C_sg,     0,           -W/C_sg]
20     ])
21     return A
22
23 # Vector B (forcing terms):
24 # B = [P_r/C_r, -Qdot1/C_sg, -Qdot2/C_sg]^T
```

**Solution**

The energy balance for each component yields differential equations:

**Reactor energy balance:**

$$C_r \frac{dT_{hot}}{dt} = P_r - W(T_{hot} - T_{cold1}) - W(T_{hot} - T_{cold2})$$

**Steam Generator 1 energy balance:**

$$C_{sg} \frac{dT_{cold1}}{dt} = W(T_{hot} - T_{cold1}) - \dot{Q}_1$$

**Steam Generator 2 energy balance:**

$$C_{sg} \frac{dT_{cold2}}{dt} = W(T_{hot} - T_{cold2}) - \dot{Q}_2$$

where:

- $P_r$ = reactor power (positive for heat generation)
- $\dot{Q}_1, \dot{Q}_2$ = steam generator heat removal rates (positive for heat removal)
- $W = \frac{C_0}{2\tau_0} = 0.6667$ %-sec/F = flow heat capacity rate per loop

**Matrix Form:**

Define the temperature vector: $\mathbf{T} = \begin{bmatrix} T_{hot} \\ T_{cold1} \\ T_{cold2} \end{bmatrix}$

The system can be written as:

$$\boxed{\frac{d\mathbf{T}}{dt} = \mathbf{AT} + \mathbf{B}}$$

where the coefficient matrix $\mathbf{A}$ is:

$$\mathbf{A} = \begin{bmatrix} -\frac{2W}{C_r} & \frac{W}{C_r} & \frac{W}{C_r} \\ \frac{W}{C_{sg}} & -\frac{W}{C_{sg}} & 0 \\ \frac{W}{C_{sg}} & 0 & -\frac{W}{C_{sg}} \end{bmatrix}$$

and the forcing vector **B** is:

$$\mathbf{B} = \begin{bmatrix} \frac{P_r}{C_r} \\ -\frac{\dot{Q}_1}{C_{sg}} \\ -\frac{\dot{Q}_2}{C_{sg}} \end{bmatrix}$$

**Numerical example for $\mu = 0.5$:**
With $\mu = 0.5$: $C_r = 16.66$ %-sec/F, $C_{sg} = 8.33$ %-sec/F, $W = 0.6667$ %-sec/F

$$\boxed{\mathbf{A} = \begin{bmatrix} -0.0800 & 0.0400 & 0.0400 \\ 0.0800 & -0.0800 & 0 \\ 0.0800 & 0 & -0.0800 \end{bmatrix}}$$

**Part B**

**Python Code**

```
# At steady state, dT/dt = 0
# From SG1: 0 = W*(T_hot - T_cold1) - Qdot1
DeltaT1 = Qdot1 / W

# From SG2: 0 = W*(T_hot - T_cold2) - Qdot2
DeltaT2 = Qdot2 / W

# From reactor: P_r = W*DeltaT1 + W*DeltaT2
power_balance = Qdot1 + Qdot2
```

**Solution**

At steady state, all time derivatives are zero ($\frac{dT}{dt} = 0$).

**From Steam Generator 1 equation:**

$$0 = W(T_{hot} - T_{cold1}) - \dot{Q}_1$$

$$\boxed{\Delta T_1 = T_{hot} - T_{cold1} = \frac{\dot{Q}_1}{W}}$$

**From Steam Generator 2 equation:**

$$0 = W(T_{hot} - T_{cold2}) - \dot{Q}_2$$

$$\boxed{\Delta T_2 = T_{hot} - T_{cold2} = \frac{\dot{Q}_2}{W}}$$

**From Reactor equation:**

$$0 = P_r - W\Delta T_1 - W\Delta T_2$$

$$P_r = W\Delta T_1 + W\Delta T_2 = \dot{Q}_1 + \dot{Q}_2$$

This confirms the power balance: reactor power equals total steam generator heat removal rates.

**Part C**

**Python Code**

```python
# Average reactor temperature (mass-weighted)
def calculate_T_ave(T_hot, T_cold1, T_cold2, mu, C_0):
    C_r = mu * C_0
    C_sg = (1 - mu) * C_0 / 2

    # Mass-weighted average
    T_ave = (C_r*T_hot + C_sg*T_cold1 + C_sg*T_cold2) / C_0

    # Simplified form
    T_ave_simplified = mu*T_hot + (1-mu)*(T_cold1 + T_cold2)/2

    return T_ave_simplified

# For mu = 0.5:
T_ave_50 = 0.5*T_hot + 0.25*T_cold1 + 0.25*T_cold2

# For mu = 0.75:
T_ave_75 = 0.75*T_hot + 0.125*T_cold1 + 0.125*T_cold2
```

**Solution**

The average reactor temperature must be calculated as a **mass-weighted average**:

$$T_{ave} = \frac{C_r T_{hot} + C_{sg} T_{cold1} + C_{sg} T_{cold2}}{C_r + 2C_{sg}}$$

Since $C_r + 2C_{sg} = \mu C_0 + 2 \cdot \frac{(1-\mu)C_0}{2} = C_0$, this simplifies to:

$$T_{ave} = \frac{C_r T_{hot} + C_{sg} T_{cold1} + C_{sg} T_{cold2}}{C_0}$$

Substituting $C_r = \mu C_0$ and $C_{sg} = \frac{(1-\mu)C_0}{2}$:

$$\boxed{T_{ave} = \mu T_{hot} + \frac{(1-\mu)}{2}(T_{cold1} + T_{cold2})}$$

**Important:** This formula *depends on* $\mu$!
**For $\mu = 0.5$:**

$$T_{ave} = \frac{T_{hot}}{2} + \frac{T_{cold1} + T_{cold2}}{4}$$

**For $\mu = 0.75$:**

$$T_{ave} = \frac{3T_{hot}}{4} + \frac{T_{cold1} + T_{cold2}}{8}$$

**Note:** When the system is balanced ($P_r = \dot{Q}_1 + \dot{Q}_2$), the mass-weighted average temperature remains constant even during transients, since $\frac{d(C_0 \cdot T_{ave})}{dt} = P_r - \dot{Q}_1 - \dot{Q}_2 = 0$.

**Part D**

**Python Code**

```python
from scipy.integrate import odeint

def reactor_odes(y, t, W, C_r, C_sg, P_r, Qdot1, Qdot2):
    """
    P_r: Reactor power (heat generation rate)
    Qdot1, Qdot2: SG heat removal rates
    """
    T_hot, T_cold1, T_cold2 = y

    dT_hot_dt = (P_r - W*(T_hot - T_cold1) - W*(T_hot - T_cold2)) / C_r
    dT_cold1_dt = (W*(T_hot - T_cold1) - Qdot1) / C_sg
    dT_cold2_dt = (W*(T_hot - T_cold2) - Qdot2) / C_sg

    return [dT_hot_dt, dT_cold1_dt, dT_cold2_dt]

# Initial conditions at equilibrium
y0 = [450, 450, 450]

# Time array: 0 to 30 seconds
t = np.linspace(0, 30, 500)

# Solve for different cases
solution = odeint(reactor_odes, y0, t, args=(W, C_r, C_sg, P_r, Qdot1, Qdot2))

T_hot = solution[:, 0]
T_cold1 = solution[:, 1]
T_cold2 = solution[:, 2]

# Calculate average temperature (mass-weighted, depends on mu)
T_ave = mu*T_hot + (1-mu)*(T_cold1 + T_cold2)/2
```

**Solution**

Transient simulations were performed for four cases using numerical integration (scipy.integrate.odeint):

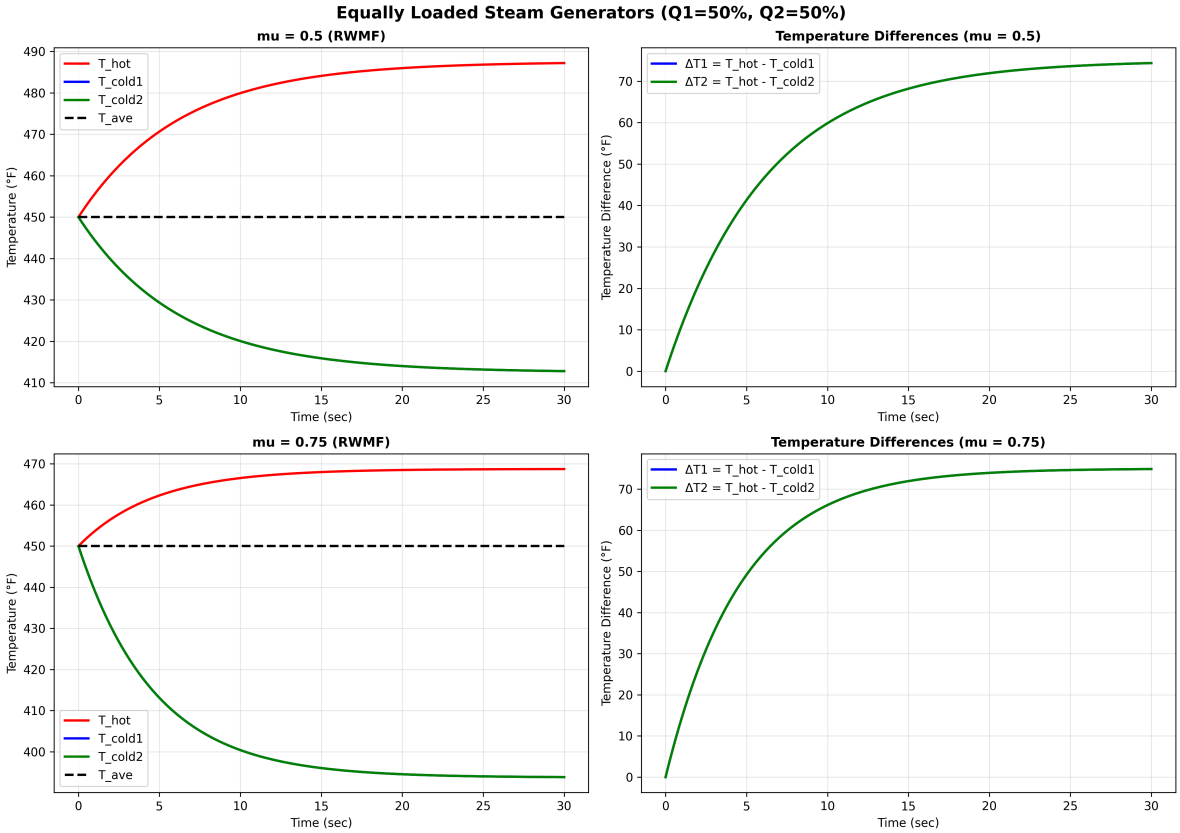
**Cases simulated:**

1. $\mu = 0.5$, equally loaded ($\dot{Q}_1 = 50\%$, $\dot{Q}_2 = 50\%$)
2. $\mu = 0.75$, equally loaded ($\dot{Q}_1 = 50\%$, $\dot{Q}_2 = 50\%$)
3. $\mu = 0.5$, unequally loaded ($\dot{Q}_1 = 60\%$, $\dot{Q}_2 = 40\%$)
4. $\mu = 0.75$, unequally loaded ($\dot{Q}_1 = 60\%$, $\dot{Q}_2 = 40\%$)

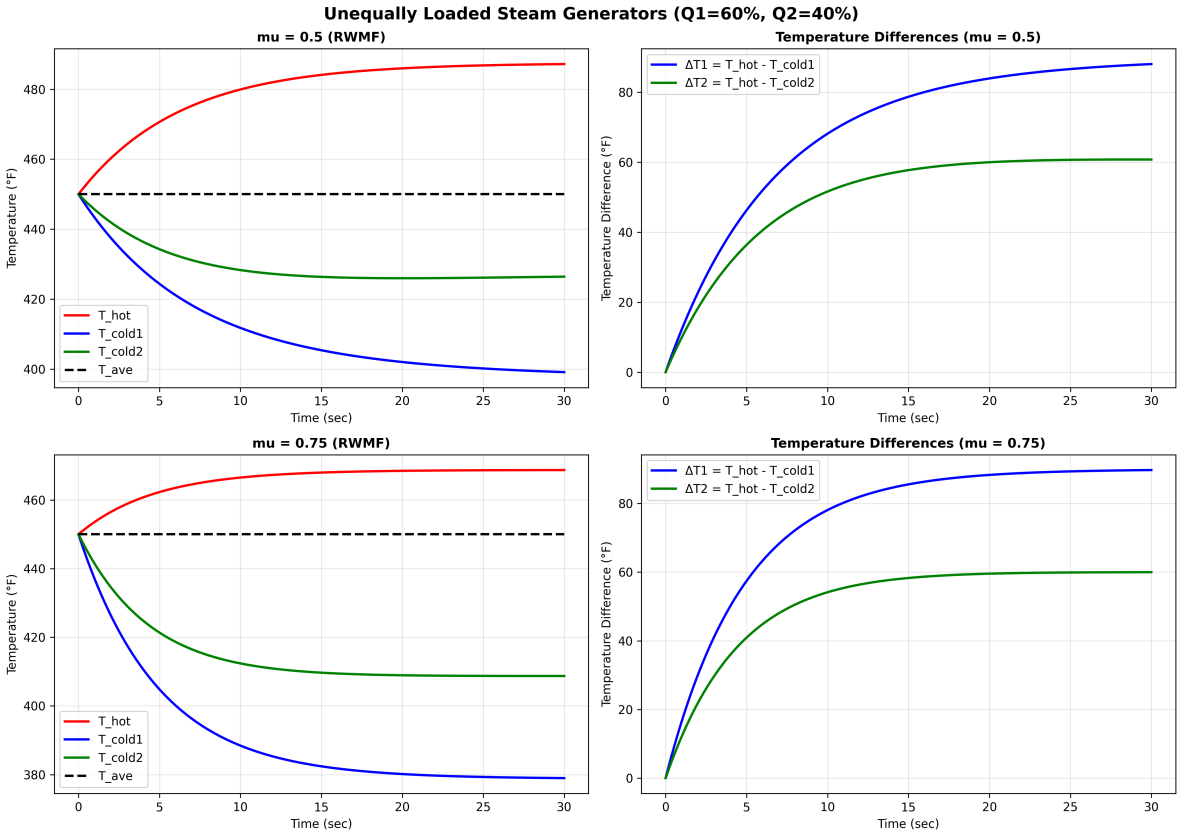
All cases assume reactor power $P_r = 100\%$ and initial conditions at equilibrium ($T_{hot} = T_{cold1} = T_{cold2} = 450$ F).

**Note:** Power percentages refer to fraction of total reactor power, not individual SG ratings.

**Equally Loaded Cases:**




**Unequally Loaded Cases:**




# Problem 2

## Cross-Section Data

Two-group cross-section data stored in Python dictionary:

```python
import numpy as np

cross_sections = {
    'fast': {
        'D': 1.4,                    # Diffusion constant [cm]
        'Sigma_a': 0.010,            # Absorption [cm^-1]
        'Sigma_s': 0.050,            # Scattering from fast to
           thermal [cm^-1]
        'nu_Sigma_f': 0.000,         # nu*Sigma_f [cm^-1]
        'chi': 1,                    # Fission spectrum
        'v': 1.8e7,                  # Average group velocity [cm/
           sec]
    },
    'thermal': {
        'D': 0.35,                   # Diffusion constant [cm]
        'Sigma_a': 0.080,            # Absorption [cm^-1]
        'Sigma_s': 0.0,              # Scattering from thermal to
           fast [cm^-1]
        'nu_Sigma_f': 0.125,         # nu*Sigma_f [cm^-1]
```

```
        'chi': 0,                       # Fission spectrum
        'v': 2.2e5,                     # Average group velocity [cm/
            sec]
    }
}

# Extract variables for easy access
D_fast = cross_sections['fast']['D']
D_thermal = cross_sections['thermal']['D']
Sigma_a_fast = cross_sections['fast']['Sigma_a']
Sigma_a_thermal = cross_sections['thermal']['Sigma_a']
Sigma_s_fast = cross_sections['fast']['Sigma_s']
nu_Sigma_f_fast = cross_sections['fast']['nu_Sigma_f']
nu_Sigma_f_thermal = cross_sections['thermal']['nu_Sigma_f']
```

**Part A**

**Python Code**

```
# Four-Factor Formula: k_inf = epsilon * p * f * eta

# Fast fission factor: epsilon = 1 (no fast fissions)
epsilon = 1.0

# Resonance escape probability
p = Sigma_s_fast / (Sigma_a_fast + Sigma_s_fast)

# Thermal utilization factor: f = 1 (single-region)
f = 1.0

# Reproduction factor
eta = nu_Sigma_f_thermal / Sigma_a_thermal

# Four-Factor Formula
k_inf = epsilon * p * f * eta

print(f"k_inf = epsilon * p * f * eta = {k_inf:.4f}")
```

**Solution**

The infinite multiplication factor is calculated using the **Four-Factor Formula**:

$$k_\infty = \varepsilon \cdot p \cdot f \cdot \eta$$

where:

- $\varepsilon$ = fast fission factor (neutrons from fast fissions per thermal fission)
- $p$ = resonance escape probability (fraction of fast neutrons reaching thermal energies)
- $f$ = thermal utilization factor (fraction of thermal neutrons absorbed in fuel)

- $\eta$ = reproduction factor (neutrons produced per thermal neutron absorbed in fuel)

**Given cross-sections:**

- $\nu\Sigma_{f,fast} = 0.000\ \text{cm}^{-1}$ (no fast fissions)
- $\nu\Sigma_{f,thermal} = 0.125\ \text{cm}^{-1}$
- $\Sigma_{a,fast} = 0.010\ \text{cm}^{-1}$
- $\Sigma_{a,thermal} = 0.080\ \text{cm}^{-1}$
- $\Sigma_{s,fast} = 0.050\ \text{cm}^{-1}$ (scattering from fast to thermal)

**Calculating each factor:**
**1. Fast fission factor:**

$$\varepsilon = 1.0000 \quad \text{(no fast fissions since } \nu\Sigma_{f,fast} = 0\text{)}$$

**2. Resonance escape probability:**

$$p = \frac{\Sigma_{s,fast}}{\Sigma_{a,fast} + \Sigma_{s,fast}} = \frac{0.050}{0.010 + 0.050} = \frac{0.050}{0.060} = 0.8333$$

**3. Thermal utilization factor:**

$$f = 1.0000 \quad \text{(single-region, homogeneous medium)}$$

**4. Reproduction factor:**

$$\eta = \frac{\nu\Sigma_{f,thermal}}{\Sigma_{a,thermal}} = \frac{0.125}{0.080} = 1.5625$$

**Final calculation:**

$$k_\infty = \varepsilon \cdot p \cdot f \cdot \eta = 1.0000 \times 0.8333 \times 1.0000 \times 1.5625 = 1.3021$$

$$\boxed{k_\infty = 1.302}$$

**Part B**

**Python Code**

```
# Calculate diffusion lengths
# L^2_fast = D_fast / Sigma_total_fast
# where Sigma_total_fast = Sigma_a_fast + Sigma_s_fast (removal from
    fast group)
Sigma_total_fast = Sigma_a_fast + Sigma_s_fast
L_squared_fast = D_fast / Sigma_total_fast

# L^2_th = D_th / Sigma_a_th
L_squared_th = D_thermal / Sigma_a_thermal

```

```python
L_fast = np.sqrt(L_squared_fast)
L_thermal = np.sqrt(L_squared_th)

print(f"L_fast = {L_fast:.3f} cm")
print(f"L_thermal = {L_thermal:.3f} cm")
```

**Solution**

The diffusion lengths for each group are calculated as:

$$L^2 = \frac{D}{\Sigma_{removal}}$$

**Fast Group:**
The removal cross-section includes both absorption and scattering out:

$$\Sigma_{removal,fast} = \Sigma_{a,fast} + \Sigma_{s,fast} = 0.010 + 0.050 = 0.060 \text{ cm}^{-1}$$

$$L^2_{fast} = \frac{D_{fast}}{\Sigma_{removal,fast}} = \frac{1.4}{0.060} = 23.333 \text{ cm}^2$$

$$L_{fast} = \sqrt{23.333} = 4.830 \text{ cm}$$

**Thermal Group:**
For the thermal group (lowest energy group), only absorption removes neutrons:

$$L^2_{thermal} = \frac{D_{thermal}}{\Sigma_{a,thermal}} = \frac{0.35}{0.080} = 4.375 \text{ cm}^2$$

$$L_{thermal} = \sqrt{4.375} = 2.092 \text{ cm}$$

$$\boxed{L_{fast} = 4.830 \text{ cm}, \quad L_{thermal} = 2.092 \text{ cm}}$$

**Part C**

**Solution**

For a rectangular solid geometry (box) with dimensions $L_x$, $L_y$, and $L_z$, where the neutron flux goes to zero at the edges (bare reactor boundary condition), the **geometric buckling** is:

$$\boxed{B^2 = \left(\frac{\pi}{L_x}\right)^2 + \left(\frac{\pi}{L_y}\right)^2 + \left(\frac{\pi}{L_z}\right)^2}$$

This expression comes from solving the neutron diffusion equation with boundary conditions $\phi = 0$ at the reactor boundaries. The solution for the fundamental mode has the form:

$$\phi(x,y,z) = A \sin\left(\frac{\pi x}{L_x}\right) \sin\left(\frac{\pi y}{L_y}\right) \sin\left(\frac{\pi z}{L_z}\right)$$

The geometric buckling is the eigenvalue associated with this spatial mode, representing the curvature of the neutron flux distribution. Each term corresponds to the buckling in one spatial dimension:

- $B_x^2 = \left(\frac{\pi}{L_x}\right)^2$ - buckling in x-direction
- $B_y^2 = \left(\frac{\pi}{L_y}\right)^2$ - buckling in y-direction
- $B_z^2 = \left(\frac{\pi}{L_z}\right)^2$ - buckling in z-direction

The total geometric buckling is the sum of the directional components.

**Note:** The derivation of this formula from the diffusion equation was completed in Exam 1. The proof is left to that work.

**Part D**

**Python Code**

```python
import sympy as sm

# Given dimensions
L_x_val = 150  # cm (width)
L_y_val = 200  # cm (length)

# Define L_z (height) as unknown
L_z_sym = sm.Symbol('L_z', positive=True)

# Buckling with unknown height
B_sq = (sm.pi / L_x_val)**2 + (sm.pi / L_y_val)**2 + (sm.pi / L_z_sym)**2

# Criticality equation: k_inf = (L^2_fast * B^2 + 1)(L^2_thermal * B^2 + 1)
criticality_eq = (L_squared_fast * B_sq + 1) * (L_squared_th * B_sq + 1) - k_inf

# Solve for L_z
L_z_solutions = sm.solve(criticality_eq, L_z_sym)
L_z_critical = float([sol for sol in L_z_solutions if sol.is_real and sol > 0][0])

print(f"Critical height L_z = {L_z_critical:.2f} cm")
```

**Solution**

For a trough with width $L_x = 150$ cm and length $L_y = 200$ cm, we need to find the critical height $L_z$ where $k_{eff} = 1$.

**Criticality condition using two-group theory:**

At criticality, the effective multiplication factor equals unity:

$$k_{eff} = \frac{k_\infty}{(L_{fast}^2 B^2 + 1)(L_{thermal}^2 B^2 + 1)} = 1$$

Rearranging:

$$(L_{fast}^2 B^2 + 1)(L_{thermal}^2 B^2 + 1) = k_\infty$$

The geometric buckling for the rectangular trough is:

$$B^2 = \left(\frac{\pi}{L_x}\right)^2 + \left(\frac{\pi}{L_y}\right)^2 + \left(\frac{\pi}{L_z}\right)^2$$

**Known values:**

- $k_\infty = 1.3021$
- $L_{fast}^2 = 23.333\ \text{cm}^2$
- $L_{thermal}^2 = 4.375\ \text{cm}^2$
- $L_x = 150\ \text{cm}$
- $L_y = 200\ \text{cm}$

**Calculation:**
Substituting the buckling expression:

$$B^2 = \left(\frac{\pi}{150}\right)^2 + \left(\frac{\pi}{200}\right)^2 + \left(\frac{\pi}{L_z}\right)^2$$

$$B^2 = 4.386 \times 10^{-4} + 2.467 \times 10^{-4} + \frac{\pi^2}{L_z^2}$$

Substituting into the criticality equation:

$$\left(23.333\left(6.853 \times 10^{-4} + \frac{\pi^2}{L_z^2}\right) + 1\right)\left(4.375\left(6.853 \times 10^{-4} + \frac{\pi^2}{L_z^2}\right) + 1\right) = 1.3021$$

Solving this equation numerically (or symbolically with SymPy) yields:

$$\boxed{L_z = 31.72\ \text{cm}}$$

**Verification:**

- $B^2 = 0.010496\ \text{cm}^{-2}$
- $k_{eff} = 1.000000$ ✓

The trough would become critical at a height of approximately 31.7 cm.

**Part E**

**Python Code**

```python
# Prompt criticality
BETA = 640e-5  # Delayed neutron fraction

# Prompt critical k_eff = 1/(1-beta)
k_eff_prompt = 1 / (1 - BETA)

# Solve for height at prompt criticality
L_z_prompt_sym = sm.Symbol('L_z_prompt', positive=True)
B_sq_prompt = (sm.pi / L_x_val)**2 + (sm.pi / L_y_val)**2 + (sm.pi /
    L_z_prompt_sym)**2

prompt_crit_eq = (L_squared_fast * B_sq_prompt + 1) * \
                 (L_squared_th * B_sq_prompt + 1) - k_inf /
                    k_eff_prompt

L_z_prompt_solutions = sm.solve(prompt_crit_eq, L_z_prompt_sym)
L_z_prompt = float([sol for sol in L_z_prompt_solutions
                    if sol.is_real and sol > 0][0])

print(f"Prompt critical height L_z = {L_z_prompt:.2f} cm")
```

**Solution**

**Prompt criticality** occurs when the reactor can sustain a chain reaction on prompt neutrons alone, without relying on delayed neutrons. This happens when:

$$k_{eff} = \frac{1}{1-\beta}$$

where $\beta$ is the delayed neutron fraction.

**Given:**

- $\beta = 640 \times 10^{-5} = 0.00640$

**Prompt critical condition:**

$$k_{eff,prompt} = \frac{1}{1-0.00640} = \frac{1}{0.99360} = 1.00644$$

Using the same two-group criticality equation from Part D, but now solving for the height where $k_{eff} = 1.00644$:

$$\frac{k_\infty}{(L_{fast}^2 B^2 + 1)(L_{thermal}^2 B^2 + 1)} = 1.00644$$

Rearranging:

$$(L_{fast}^2 B^2 + 1)(L_{thermal}^2 B^2 + 1) = \frac{k_\infty}{k_{eff,prompt}} = \frac{1.3021}{1.00644} = 1.2938$$

With $B^2 = \left(\frac{\pi}{150}\right)^2 + \left(\frac{\pi}{200}\right)^2 + \left(\frac{\pi}{L_z}\right)^2$, solving numerically:

$$\boxed{L_{z,prompt} = 32.18 \text{ cm}}$$

**Comparison:**

- Delayed critical height: $L_z = 31.72$ cm
- Prompt critical height: $L_{z,prompt} = 32.18$ cm
- Difference: $\Delta L_z = 0.46$ cm

The liquid must rise an additional 0.46 cm above delayed criticality to reach prompt criticality. This small difference highlights why delayed neutrons are crucial for reactor control.

**Part F**

**Solution**

The presence of people near the trough could significantly impact the critical height.

**Physical mechanism:**

If the neutron flux is not actually zero at the trough edges (as assumed in our bare reactor model), people standing nearby would:

1. **Act as neutron reflectors:** Human bodies contain significant amounts of water (∼60% by mass), which is an excellent neutron moderator and reflector
2. **Reduce neutron leakage:** Neutrons that would have escaped the trough can be scattered back by the hydrogen in the water content of human tissue
3. **Increase system reactivity:** Reduced leakage means more neutrons remain in the system to cause fissions

**Impact on critical height:**

$$\boxed{\text{Critical height would DECREASE}}$$

## Problem 3

**Part A**

**Python Code**

```python
import numpy as np

# Problem 3A
## Using formulas from Fundamental Kinetics Ideas R17 Page 51
DRW = 10  # pcm/step
STEPS = 8
LAMBDA_EFF = 0.1  # hz

# ASSUMING AFTER ROD PULL COMPLETE, RHO_DOT = 0
RHO_DOT = 0
BETA = 640  # pcm
```

```
12
13 # FIND RHO AFTER ROD PULL
14 rho = DRW * STEPS  # pcm
15
16 sur = 26.06 * (RHO_DOT + LAMBDA_EFF * rho) / (BETA - rho)
17
18 print(f"The Start Up Rate is: {sur:.3f}")
```

**Solution**

Given:

- Differential Rod Worth (DRW) = 10 pcm/step
- Number of steps = 8
- $\lambda_{eff}$ = 0.1 Hz
- $\dot{\rho}$ = 0 (after rod pull complete)
- $\beta$ = 640 pcm

Reactivity after rod pull:

$$\rho = \text{DRW} \times \text{STEPS} = 10 \times 8 = 80 \text{ pcm}$$

Start-up rate calculation:

$$\text{SUR} = \frac{26.06 \times (\dot{\rho} + \lambda_{eff} \times \rho)}{\beta - \rho} = \frac{26.06 \times (0 + 0.1 \times 80)}{640 - 80}$$

$$\boxed{\text{Start Up Rate} = 0.373 \text{ DPM}}$$

**Part B**

Negative reactivity feedback due to temperature would cause this power level off. I would expect that the average reactor temperature would have increased from the low power state significantly. I would also expect xenon concentration would have increased, but would not have been the culprit in power leveling off.

**Part C**

**Python Code**

```
1  # Problem 3C
2  import sympy as sm
3
4  D_POWER = 2.5  # %
5  D_T_AVG = 4  # degrees
6  HEAT_UP_RATE = 0.15  # F/s
7  ALPHA_F = -10  # pcm/%power (negative feedback)
8
9  rho_rod = rho
10
```

```python
# The heat up rate introduces a rho_dot, so SUR becomes 0 at the peak power.
alpha_w_sym = sm.Symbol("alpha_w")
rho_dot = alpha_w_sym * HEAT_UP_RATE
rho_net = alpha_w_sym * D_T_AVG + rho_rod + ALPHA_F * D_POWER

# At peak power, SUR = 0, which means: rho_dot + lambda_eff * rho_net = 0
# (the numerator must be zero)
equation = rho_dot + LAMBDA_EFF * rho_net

# Solve for alpha_w
alpha_w_solution = sm.solve(equation, alpha_w_sym)[0]
alpha_w = float(alpha_w_solution)

print(f"The water temperature reactivity coefficient is: {alpha_w:.3f} pcm/F")
```

**Solution**

Given:

- Power change at peak: $\Delta P = 2.5\%$
- Average temperature change at peak: $\Delta T_{avg} = 4°\text{F}$
- Heat-up rate: $\dot{T} = 0.15$ °F/s
- Fuel temperature coefficient: $\alpha_f$ = -10 pcm/%power (negative feedback)
- Rod reactivity: $\rho_{rod} = 80$ pcm (from Part A)
- $\lambda_{eff} = 0.1$ Hz

At peak power, the start-up rate becomes zero (SUR = 0), but temperature is still rising. This is the key insight: the numerator of the SUR equation must equal zero:

$$\dot{\rho} + \lambda_{eff} \times \rho_{net} = 0$$

The temperature rise creates a reactivity change rate:

$$\dot{\rho} = \alpha_w \times \dot{T} = \alpha_w \times 0.15$$

The net reactivity at the peak is:

$$\rho_{net} = \alpha_w \Delta T_{avg} + \rho_{rod} + \alpha_f \Delta P = \alpha_w \times 4 + 80 + (-10) \times 2.5$$

Substituting into the SUR = 0 condition:

$$\alpha_w \times 0.15 + 0.1 \times (\alpha_w \times 4 + 80 - 25) = 0$$

$$0.15\alpha_w + 0.4\alpha_w + 5.5 = 0$$

$$0.55\alpha_w = -5.5$$

$$\boxed{\alpha_w = -10.000 \text{ pcm/}^\circ\text{F}}$$

**Part D**

**Solution**

At final equilibrium when the transient is complete:

- Temperature stops changing: $\dot{T} = 0 \Rightarrow \dot{\rho} = 0$
- Start-up rate returns to zero: SUR = 0
- Net reactivity must be zero: $\rho_{net} = 0$

Since $\dot{\rho} = 0$ at equilibrium, the SUR equation requires:

$$\text{SUR} = \frac{26.06 \times (0 + \lambda_{eff} \times \rho_{net})}{\beta - \rho_{net}} = 0$$

This is satisfied when $\rho_{net} = 0$:

$$\alpha_w T_{final} + \rho_{rod} + \alpha_f P_{final} = 0$$

However, without knowing the heat removal characteristics (i.e., the relationship between power generation and temperature at thermal equilibrium with ambient losses), we cannot solve for exact values of $T_{final}$ and $P_{final}$.

**Qualitative Analysis:**

The transient behavior proceeds as follows:

1. **At the peak** ($\Delta T = 4^\circ$F, $\Delta P = 2.5\%$): SUR = 0, but temperature is still rising at 0.15 °F/s
2. **After the peak**: Temperature continues to rise $\Rightarrow$ more negative reactivity is added $\Rightarrow$ power decreases from its maximum
3. **At final equilibrium**: Temperature plateaus when heat generation equals ambient heat removal

Therefore:

$$\boxed{T_{final} > 4^\circ\text{F} \quad \text{and} \quad P_{final} < 2.5\%}$$

The final power is lower than the peak power, but the final temperature is higher than the peak temperature. The peak power at 2.5% is a transient maximum, not the steady-state equilibrium value.

# Problem 4

**Problem Statement**

A pressurized water reactor (with highly enriched fuel) is initially at a steady state of 25% steam load. The operators are directed to raise power (picking up electrical load) by increasing steam flow to 50% in a single motion.

**Initial conditions:**

- $T_{ave} = 500$ F
- $T_h = 510$ F (hot leg)
- $T_c = 490$ F (cold leg)
- Power = 25%
- No automatic control forcing changes in average temperature

**Part A**

**Solution**

**Chronological list of physical impacts on the primary system and reactor:**

**Secondary Side (Given):**

1. Turbine throttle opened (operators pick up electrical load)
2. Steam flow increases from 25% to 50%
3. Pressure in steam generator drops
4. Aggressive boiling in steam generator begins
5. SG water cools (maintains saturation conditions)
6. Primary side cold leg ($T_c$) cooled by steam generator

**Primary Side and Reactor:**

7. Cold leg temperature ($T_c$) decreases
8. Average temperature ($T_{ave}$) decreases (no automatic control)
9. **Moderator temperature reactivity feedback** ($\alpha_{mod} < 0$):
    - Cooler moderator $\rightarrow$ increased water density
    - Better neutron moderation and thermalization
    - Positive reactivity insertion: $\rho = \alpha_{mod} \times \Delta T_{ave}$
    - Since $\alpha_{mod} < 0$ and $\Delta T_{ave} < 0$, then $\rho > 0$
10. Reactor power begins to increase
11. Fuel temperature increases $\rightarrow$ negative fuel feedback (Doppler effect)
12. Hot leg temperature ($T_h$) increases
13. Core $\Delta T$ increases: $\Delta T = \frac{\text{Power}}{\dot{m} \times c_p}$
14. Average temperature begins to rise back toward initial value
15. **Power levels off at 50% when:**
    - Heat removal rate matches heat generation rate
    - $\Delta T$ establishes new equilibrium: $\Delta T_{new} = \Delta T_{old} \times \frac{P_{new}}{P_{old}}$
    - $T_{ave}$ returns to approximately initial value
    - Net reactivity returns to zero (reactor critical)

**Impacts on six factors in $k_{eff} = \varepsilon \times p \times f \times \eta \times P_{FNL} \times P_{TNL}$:**

1. $\varepsilon$ **(fast fission factor):** Negligible change
    - Minimal U-238 present for fast fission
    - No significant change
2. $p$ **(resonance escape probability):** Slight increase

- Limited U-238 resonance absorption with high enrichment
- Cooler moderator → higher density → more scattering
- Faster neutron slowing down through resonance region
- Less time spent at resonance energies → less absorption
- $p$ increases slightly

3. $f$ **(thermal utilization factor):** Possible slight increase
   - $f = \frac{\Sigma_{a,fuel}}{\Sigma_{a,fuel}+\Sigma_{a,mod}}$
   - Cooler moderator → better thermalization efficiency
   - More neutrons successfully reach thermal energies where fuel cross section dominates
   - Competing effect: higher moderator density increases $\Sigma_{a,mod}$
   - Net effect: likely small increase
4. $\eta$ **(reproduction factor):** Minimal change
   - $\eta = \nu\frac{\sigma_f}{\sigma_a}$ for U-235
   - Temperature effects on U-235 cross sections are small
   - Essentially constant
5. $P_{FNL}$ **(fast non-leakage):** Minimal change
   - Large reactor → already high fast non-leakage
   - Small temperature changes don't significantly affect
6. $P_{TNL}$ **(thermal non-leakage):** Slight increase
   - Cooler moderator → higher density → shorter diffusion length
   - Reduced thermal neutron leakage
   - However, large reactor already has high $P_{TNL}$ (low leakage baseline)
   - Effect is present but modest in absolute terms

**Overall mechanism:** Net positive reactivity from moderator cooling results from the combined contributions of increased $p$ (faster slowing through resonances), increased $f$ (better thermalization), and increased $P_{TNL}$ (reduced leakage). For a large reactor with high enrichment, multiple effects contribute to the reactivity rather than a single dominant mechanism. The cooler, denser moderator improves neutron economy across several factors, causing power to increase until new equilibrium at 50%.

**Part B**

**Python Code**

```
# Initial conditions
T_ave_initial = 500  # F
T_h_initial = 510    # F
T_c_initial = 490    # F
P_initial = 25       # %
P_final = 50         # %

# Calculate DeltaT
DeltaT_initial = T_h_initial - T_c_initial  # 20 F
```

```
DeltaT_final = DeltaT_initial * (P_final / P_initial)  # 40 F

# Estimate final temperatures (assume T_ave approximately constant)
T_ave_final = T_ave_initial
T_h_final = T_ave_final + DeltaT_final / 2
T_c_final = T_ave_final - DeltaT_final / 2
```

**Solution**

At steady state, reactor power is proportional to the temperature rise across the core:

$$P \propto \dot{m} \times c_p \times \Delta T$$

For constant flow rate (no pump speed change):

$$\frac{P_{new}}{P_{old}} = \frac{\Delta T_{new}}{\Delta T_{old}}$$

**Initial state (25% power):**

- $\Delta T_{initial} = T_h - T_c = 510 - 490 = 20$ F

**Final state (50% power):**

$$\Delta T_{final} = \Delta T_{initial} \times \frac{P_{final}}{P_{initial}} = 20 \times \frac{50}{25} = \boxed{40 \text{ F}}$$

Assuming strong moderator feedback returns $T_{ave}$ to approximately its initial value:

$$T_{ave,final} \approx 500 \text{ F}$$

$$T_{h,final} = T_{ave} + \frac{\Delta T}{2} = 500 + 20 = \boxed{520 \text{ F}}$$

$$T_{c,final} = T_{ave} - \frac{\Delta T}{2} = 500 - 20 = \boxed{480 \text{ F}}$$

**Comparison table:**

| Parameter | Initial (25%) | Final (50%) | Change |
|---|---|---|---|
| Power | 25% | 50% | +25% |
| $T_h$ (F) | 510 | 520 | +10 F |
| $T_c$ (F) | 490 | 480 | −10 F |
| $T_{ave}$ (F) | 500 | ∼500 | ∼0 F |
| $\Delta T$ (F) | 20 | 40 | +20 F |

**Part C**

**Solution**

For a reactor with low enrichment fuel (3-5% U-235), the final condition will be significantly different due to the large quantity of U-238 present. Low enrichment fuel contains approximately

95% U-238, compared to only about 7% in the highly enriched case.

As power increases and fuel temperature rises, Doppler broadening of U-238 resonances creates a strong negative fuel temperature coefficient. This negative fuel feedback counteracts the positive moderator temperature feedback from the cooler water. The net reactivity insertion becomes much smaller than in the high enrichment case, and the power increase may be insufficient to reach 50%.

To restore proper operation, operators must withdraw control rods to add positive reactivity and overcome the strong negative Doppler feedback. This allows the reactor to reach the desired 50% power level matching the steam demand.

## Problem 5

### Part A

### Solution

The xenon-135 transient for the given power history is solved using the coupled differential equations for I-135 and Xe-135:

$$\frac{dI}{dt} = -\lambda_I I + \rho \gamma_I P_0$$

$$\frac{dX}{dt} = -\lambda_{Xe} X - \rho R^{Max} X + \lambda_I I + \rho \gamma_{Xe} P_0$$

where $\rho$ is the normalized power (1.0 = 100% power).

**Power History:**

- 0-5 hours: 100% power
- 5-15 hours: Shutdown
- 15-50 hours: 100% power
- 50-80 hours: 40% power
- 80-100 hours: Shutdown
- 100-150 hours: 100% power

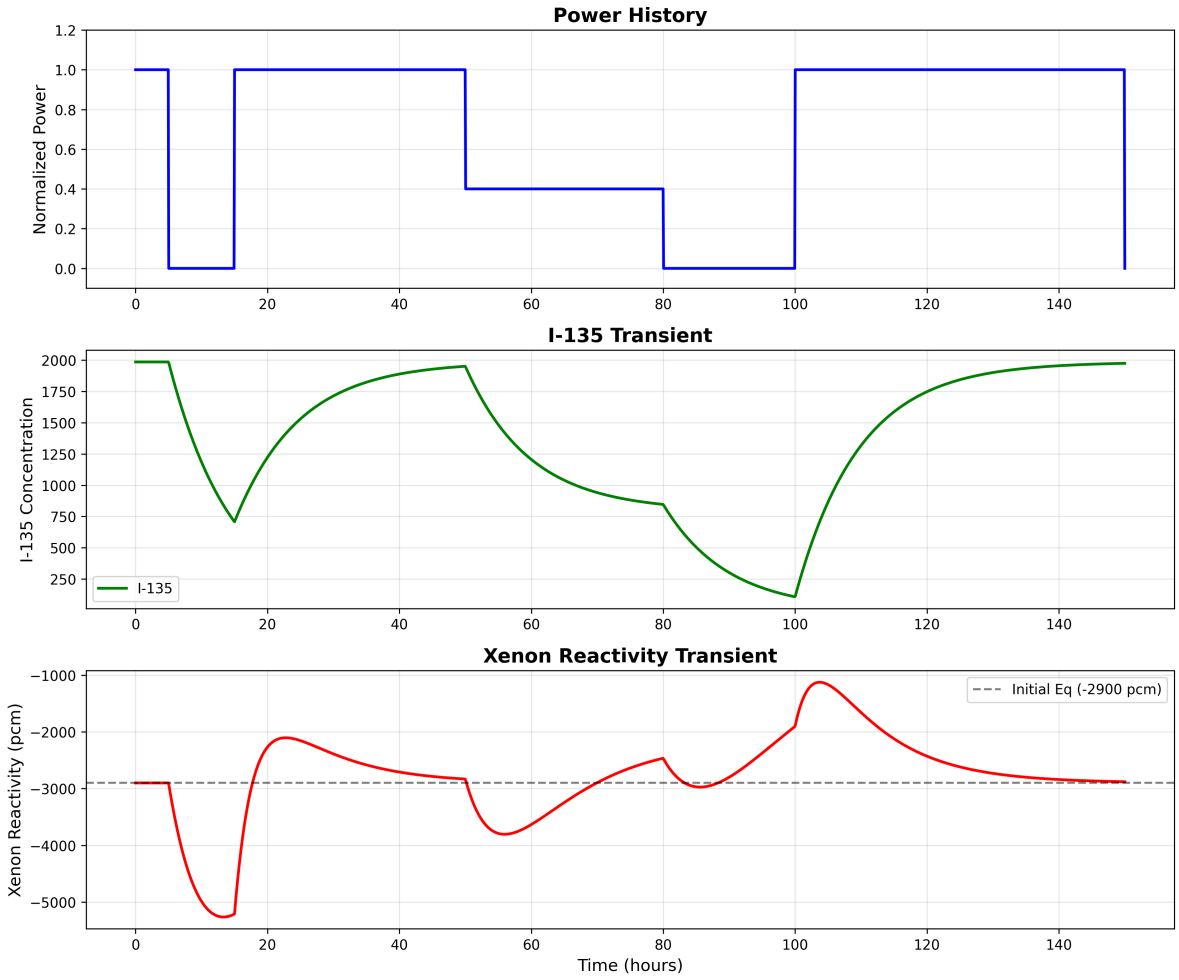
**Key features of the xenon transient:**

1. **Initial equilibrium (0-5 hours):** At 100% power, xenon reactivity = -2900 pcm
2. **First shutdown (5-15 hours):**
    - Xenon burnout stops immediately (no neutron flux)
    - I-135 continues to decay into Xe-135
    - Xenon concentration rises, reaching a peak around 8-9 hours after shutdown
    - Most negative xenon reactivity occurs
3. **Return to full power (t = 15 hours):**
    - Xenon burnout resumes at full rate
    - System returns to equilibrium at 100% power
    - Xenon reactivity returns to -2900 pcm
4. **Power reduction to 40% (t = 50 hours):**

- Reduced burnout rate (40% of full power)
- Xenon concentration increases
- System approaches new equilibrium at 40% power
- Equilibrium xenon significantly higher at lower power

5. **Second shutdown (80-100 hours):**
    - Similar xenon peak behavior to first shutdown
    - Starting from 40% power equilibrium
    - Peak less pronounced due to lower initial I-135 inventory
6. **Return to full power (t = 100 hours):**
    - Final return to 100% power operation
    - System approaches equilibrium xenon level
    - Xenon reactivity returns to -2900 pcm

The xenon transient is shown in the figure below (computed using scipy.integrate.odeint):




## Part B

## Python Code

```python
from scipy.integrate import odeint

# Define ODE system
def xenon_ode(y, t, power_func):
    I, X = y
    t_hours = t / 3600
    rho = power_func(t_hours)

    dI_dt = -lambda_I * I + rho * gamma_I * P0
    dX_dt = -lambda_Xe * X - rho * R_max * X + lambda_I * I + rho *
       gamma_Xe * P0

    return [dI_dt, dX_dt]

# Initial conditions at full power equilibrium
I0 = gamma_I * P0 / lambda_I
X0 = abs(Xe_eq_reactivity) / K

# Solve ODE over time period
t_hours = np.linspace(0, 150, 2000)
t_seconds = t_hours * 3600
solution = odeint(xenon_ode, [I0, X0], t_seconds, args=(get_power,))

# Find peak after first shutdown (5-15 hours)
X_transient = solution[:, 1]
Xe_reactivity = -K * X_transient
mask = (t_hours >= 5) & (t_hours <= 15)
peak_idx = np.argmin(Xe_reactivity[mask])
```

**Solution**

After the first shutdown at t = 5 hours (shutdown period: 5-15 hours), xenon-135 concentration increases due to:

1. Continued decay of I-135 inventory into Xe-135
2. Elimination of xenon burnout (no neutron flux)

The peak occurs when the production rate from I-135 decay equals the Xe-135 decay rate. This typically happens 8-12 hours after shutdown from full power operation.

**Given parameters:**

- $\gamma_I = 0.057$ (I-135 fission yield)
- $\gamma_{Xe} = 0.003$ (Xe-135 fission yield)
- $\lambda_I = 2.87 \times 10^{-5}$ sec$^{-1}$ (I-135 decay, $t_{1/2}$ = 6.7 hr)
- $\lambda_{Xe} = 2.09 \times 10^{-5}$ sec$^{-1}$ (Xe-135 decay, $t_{1/2}$ = 9.2 hr)
- $R^{Max} = 7.34 \times 10^{-5}$ sec$^{-1}$ (full power burnout)
- $K = 4.56$ pcm·sec$^{-1}$

- Initial Xe reactivity = -2900 pcm (at 100% power)

**Initial equilibrium concentrations (100% power):**
At equilibrium with $\rho = 1.0$:

$$I_{eq} = \frac{\gamma_I P_0}{\lambda_I} = 1985.12 \text{ [arb. units]}$$

$$X_{eq} = \frac{|\text{Xe reactivity}|}{K} = \frac{2900}{4.56} = 635.96 \text{ [arb. units]}$$

**Results from numerical integration:**

$$\boxed{\text{Time of peak: } t = 13.36 \text{ hours}}$$

$$\boxed{\text{Time after shutdown: } \Delta t = 8.36 \text{ hours}}$$

$$\boxed{\text{Peak xenon reactivity: } -5261 \text{ pcm}}$$

## Problem 6

### Part A

Core design that prohibits adequate transfer of power between core regions will increase the likelihood of oscillations. In our notes for 'Simplified Parallel Coupled Reactors', we summarized this communication between reactor regions as a parameter $g$. Designs that have connections between areas with small $g$ will suffer from worse oscillations. I would presume reactors that have large aspect ratios would suffer more from oscillations, as it would be harder for different ends of the reactor core to 'communicate' with one another.

### Part B

These oscillations will cause damage to the fuel and reactor over time. The reactor is presumably not designed to carry such high power loads in localized regions of the reactor, as opposed to a balanced power load across the entire reactor core.

### Part C

Oscillations might impact core protection or safety analysis by obscuring the actual reactivity or temperature values inside the reactor core. Without proper care to obtain good measurements, a reactor operator could not be aware that certain oscillating areas of the core are exceeding temperature and local power limits, all the while the reactor as a whole may appear as if it's behaving normally. The result is that while coolant flow in and out of the reactor maintain normal temperature, oscillating fuel rods may actually be pushing beyond designed limits, and compromising their cladding, performance, or other important characteristics..