NUCE 2101: Exam 2

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

Sunday 9th November, 2025

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

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

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

# Resonance escape probability
```

```python
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:

- ε = 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)
- η = reproduction factor (neutrons produced per thermal neutron absorbed in fuel)

Given cross-sections:

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

```python
# 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

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:

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

```
# 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 β 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:

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

With people nearby acting as reflectors:

- The effective non-leakage probability increases
- Less fissile material is needed to achieve $k_{eff} = 1$
- Critical height would be **lower** than our calculated 31.72 cm

Safety implications:

This is a **serious criticality safety concern**. The presence of personnel near fissile liquid containers can:

- Make systems go critical at lower fill levels than predicted by bare reactor calculations
- Create inadvertent criticality accidents
- Necessitate larger safety margins and administrative controls

Historical note: Several criticality accidents have occurred due to personnel proximity acting as reflectors, including incidents during the Manhattan Project. This is why strict distance requirements and neutron shielding are mandated in facilities handling fissile materials.

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

# FIND RHO AFTER ROD PULL
rho = DRW * STEPS  # pcm

sur = 26.06 * (RHO_DOT + LAMBDA_EFF * rho) / (BETA - rho)

print(f"The Start Up Rate is: {sur:.3f}")
```

Solution

Given:

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

```
# Problem 3C
import sympy as sm

D_POWER = 2.5  # %
D_T_AVG = 4  # degrees
HEAT_UP_RATE = 0.15  # F/s
ALPHA_F = 10  # pcm/%power

rho_rod = rho

# 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:.3
    f} 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
- 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:

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

$$0.15\alpha_w + 0.4\alpha_w + 10.5 = 0$$

$$0.55\alpha_w = -10.5$$

$$\boxed{\alpha_w = -19.091 \text{ pcm/°F}}$$

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** ($\Delta T = 4°$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:

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

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 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 ρ is the normalized power (1.0 = 100% power).

Power History:

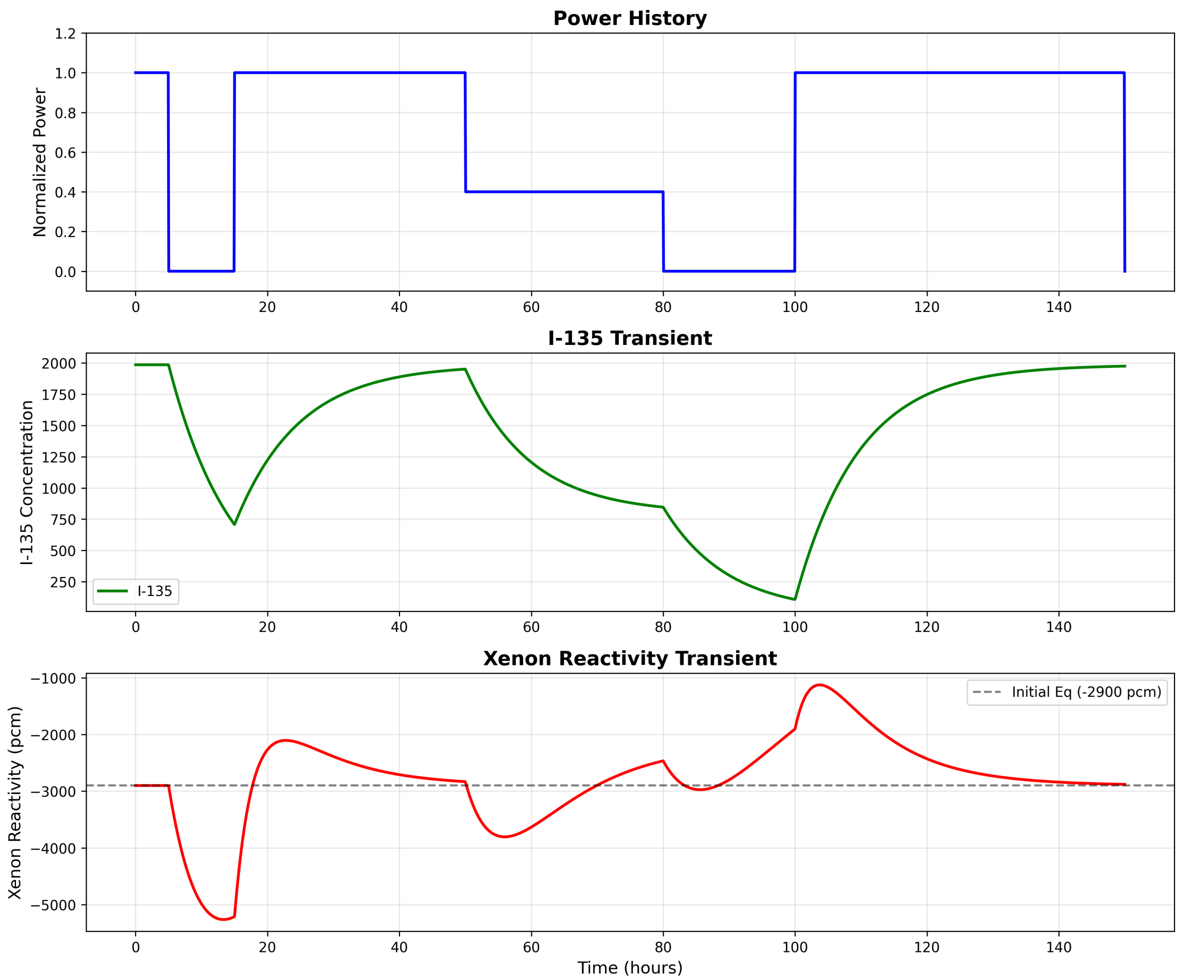
- 0-5 hours: 100% power

- 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

```
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
```

```python
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:

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

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

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

Interpretation:

- The xenon reactivity becomes 2361 pcm more negative than equilibrium
- Peak occurs approximately 8.4 hours after shutdown
- This represents a significant reactivity penalty that must be overcome to restart
- If reactivity worth is insufficient, the reactor cannot be restarted until xenon decays
- At t = 15 hours, when power returns to 100%, xenon has already started to decay from peak

Physical insight:

The time to peak can be estimated analytically. After shutdown, I-135 decays with time constant $1/\lambda_I \approx 10$ hours, while Xe-135 decays with $1/\lambda_{Xe} \approx 13$ hours. The peak occurs when:

$$\frac{dX}{dt} = \lambda_I I(t) - \lambda_{Xe} X(t) = 0$$

This typically occurs at $t \approx 8 - 12$ hours after shutdown for thermal reactors, consistent with our computed value of 8.36 hours. The reactor restarts at t = 15 hours, which is about 1.6 hours after the xenon peak, when xenon has already begun to decay.

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..