Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра дослідження операцій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на здобуття ступеня бакалавра

за спеціальністю 113 «Прикладна математика»

на тему:

Імітаційне моделювання загальної моделі резервованої системи з відновленням

Студента 4-го курсу
Коваля Максима Романовича

Науковий керівник:
професор, доктор фізико-математичних наук
Мацак Іван Каленикович

Робота заслухана на засіданні кафедри обчислювальної математики та рекомендована до захисту в ЕК, протокол № ..... від ..... 2023 р.

Завідувач кафедри ДО проф. Іксанов О. М.

Київ – 2023

Зміст

ВСТУП

В сучасному світі, де технології стають все більш складними і взаємозалежними, питання надійності і ефективності їх використання стають все більш актуальними. Нерідко випадки збою одного елементу можуть призвести до серйозних наслідків, включаючи зупинку роботи всієї системи. В цьому контексті, резервовані системи з відновленням стають важливим засобом забезпечення надійності та продуктивності.

В межах цієї роботи буде розглянуто імітаційне моделювання загальної моделі резервованої системи з відновленням. Мета даної роботи полягає у визначенні ймовірності відмови такої системи за допомогою звичайного імітаційного моделювання (метод Монте-Карло) та методу швидкого імітаційного моделювання.

При виконанні роботи ми будемо розглядати систему, що містить декілька робочих елементів трьох типів, резервні елементи для кожного типу, чергу та задану кількість ремонтних одиниць. Вибір такої моделі зумовлений її широким застосуванням в практичних сценаріях.

Ця робота має на меті забезпечити корисні висновки та рекомендації, які можуть бути використані для покращення надійності та ефективності використання резервованих систем з відновленням.

1 МОДЕЛЮВАННЯ ВИПАДКОВИХ ВЕЛЧИН

1.1 Рівномірний розподіл

Моделювання випадкових величин із заданим розподілом зазвичай відбувається в два етапи: моделювання послідовності рівномірно розподілених чисел на інтервалі $(0,1)$ та перетворення цієї послідовності на послідовність, що задовольняє заданому розподілу.

Для моделювання послідовності рівномірно розподілених чисел достатньо згенерувати послідовність X_k цілих чисел на проміжку $[0, M-1]$, після чого послідовність $\frac{X_k}{M-1}$ приблизно рівномірно розподілити на інтервал $(0,1)$. Існує багато методів генерації рівномірно розподілених послідовностей. Наприклад, Дерік Лемер, автор одного з перших таких методів, запропонував наступну формулу [1, с. 141–146]:

$$X_k = a^k \bmod M.$$

Цей вираз можна представити і у рекурентній формі:

$$X_k = (a \cdot X_{k-1}) \bmod M.$$

Параметри a та M прийнято називати «множником» та «модулем» відповідно. Ці параметри можна обирати довільно, однак не всі значення даватимуть результат, наближений до теоретичного. Ось приклад значень, які були статистично перевірені на відповідність рівномірному розподілу:

$$a = 742938285, \;\; M = 2^{31} - 1.$$

В подальших викладках будемо припускати наявність генератора випадкової величини, рівномірно розподіленої на $(0,1)$, без прив’язки до конкретного методу її моделювання.

1.2 Метод зворотної функції

Як вже згадувалося, моделювання нерівномірних випадкових величин часто виконується шляхом перетворення (псевдовипадкових) рівномірних випадкових величин. Розглянемо найпростіший метод моделювання такого типу – метод зворотної функції.

Нехай маємо деяку випадкову величину ξ із функцією розподілу $F_\xi(x)$. Оберненою до функції розподілу $F_\xi(x)$ є функція $F_\xi^{-1}(u)$, яка визначена на інтервалі $(0,1)$: для $0 < u < 1$ значення $F_\xi^{-1}(u)$ є унікальним дійсним числом x таким, що $F(x) = u$. Іншими словами

$$F_\xi\left(F_\xi^{-1}(u)\right) = u, \qquad F_\xi^{-1}\left(F_\xi(x)\right) = x.$$

Нехай U – рівномірно розподілена величина на $(0, 1)$. Тоді

$$P\left(F_\xi^{-1}(U) \le x\right) = P\left(U \le F_\xi(x)\right) = F(x).$$

Отже, $F_\xi^{-1}(U)$ має функцію розподілу F_ξ. Таким чином, для того, щоб згенерувати випадкову величину ξ з функцією розподілу F_ξ, нам достатньо згенерувати U – рівномірно розподілену випадкову величину на $(0,1)$ – та обчислити $F_\xi^{-1}(U)$.

1.3 Показниковий розподіл

Розглянемо метод зворотної функції на прикладі генерації випадкової величини з показниковим розподілом.

Нехай маємо випадкову величину ξ, розподілену за показниковим розподілом з параметром λ. Відомо, що ξ має наступну функцію розподілу:

$$F_\xi(x) = 1 - e^{-\lambda x}.$$

Зворотною до цієї функції розподілу буде функція

$$F_\xi^{-1}(u) = -\frac{1}{\lambda}\ln(1-u).$$

Отже, для того, щоб змоделювати випадкову величину ξ потрібно:

1) згенерувати U – рівномірно розподілену випадкову величину на $(0,1)$;
2) обчислити $F_{\xi}^{-1}(U) = -\frac{1}{\lambda}\ln(1-U)$ – шукана реалізація випадкової величини ξ.

2 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

2.1 Поняття імітаційного моделювання

Імітаційне моделювання (або метод Монте-Карло, або статистичне моделювання) використовується як універсальний інструмент для знаходження наближених рішень здебільшого прикладних задач, особливо коли традиційні аналітичні методи виявляються непридатними у зв'язку з високою складністю відповідних задач.

Основоположні принципи імітаційного моделювання ґрунтуються на класичних теоремах теорії ймовірності, таких як ЗВЧ (закон великих чисел) [2] та ЦГТ (центральна гранична теорема) [3]. Закон великих чисел гарантує високу точність оцінок, що базуються на великих вибірках, вказуючи на те, що середні значення випадкових величин з часом наближаються до їх теоретичних математичних очікувань. Центральна гранична теорема в свою чергу важлива для імітаційного моделювання, оскільки вона дозволяє використовувати нормальний розподіл для апроксимації суми великої кількості випадкових величин, незалежно від форми їхніх окремих розподілів.

Основна ідея імітаційного моделювання полягає в генерації множини випадкових величин, що імітують реальні явища, замість використання складного аналітичного апарату. Після проведення достатньої кількості таких "експериментів", отримані дані можуть бути оброблені статистичними методами для отримання наближених характеристик: ймовірностей подій, середніх значень, стандартних відхилень, тощо [4, с. 10-11].

Діапазон можливих застосувань імітаційного моделювання є досить широким, проте його раціонально використовувати тоді, коли імітаційна процедура не є значно складнішою за аналітичний розрахунок. Так, значна частина реальних задач з теорії масового обслуговування та математичної теорії надійності вимагає використання імітаційного моделювання для розв'язку.

2.2 Приклад застосування

Розглянемо приклад застосування імітаційного моделювання. Нехай потрібно обчислити інтеграл:

$$I = \int_a^b g(x)dx,$$

де $g(x)$ – неперервна функція, $a, b \in \mathbb{R}$.

Виберемо $\xi_1, \xi_2, \dots, \xi_n$ – незалежні однаково-розподілені випадкові величини з функціями щільності $f_\xi(x)$:

$$f_\xi(x) > 0,\ x \in [a, b],$$
$$f_\xi(x) = 0,\ x \notin [a, b].$$

Розглянемо наступну функції від випадкових величин $\xi_i,\ i = \overline{1, n}$:

$$\eta_i(\xi_i) = \frac{g(\xi_i)}{f_\xi(\xi_i)}.$$

Тоді матимемо

$$E\eta_i(\xi) = \int_a^b \eta_i(x) f_\xi(x)dx = \int_a^b \frac{g(x)}{f_\xi(x)} f_\xi(x)dx = \int_a^b g(x)dx.$$

Згідно з законом великих чисел

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \eta_i = E\eta_1 = \int_a^b g(x)dx.$$

Таким чином, отримали наступний алгоритм наближеного обчислення інтегралу:

1) модулюємо послідовність $\xi_1, \xi_2, \dots, \xi_N$ – незалежні однаково-розподілені випадкові величини з функціями щільності $f_\xi(x)$;
2) обчислюємо відповідні значення $\eta_i(\xi_i) = \frac{g(\xi_i)}{f_\xi(\xi_i)},\ i = \overline{1, N}$;
3) обчислюємо $\frac{1}{n}\sum_{i=1}^{N} \eta_i \approx \int_a^b g(x)dx$.

Застосуємо даний алгоритм для наближеного обчислення наступного інтегралу:

$$I_0 = \int_1^e \ln(x)\,dx.$$

Не складно порахувати, що $I_0 = 1$.

Нехай $\xi_1, \xi_2, \dots, \xi_n$ мають рівномірний розподіл на $[1, e]$. Тоді

$$f_\xi(x) = \begin{cases} \dfrac{1}{e-1}, & x \in [1, e] \\ 0, & x \notin [1, e] \end{cases}$$

Для генерації $\xi_1, \xi_2, \dots, \xi_n$ скористаємося наступною формулою:

$$\xi_i = 1 + u \cdot (e - 1),$$

де u – рівномірно розподілена величина на $(0,1)$.

Таким чином, для наближеного обчислення заданого інтегралу потрібно:

- згенерувати N рівномірно розподілених випадкових величин на $(0,1)$;
- для кожної з N реалізацій обчислити

$$\xi_i = 1 + e \cdot u, \qquad i = \overline{1, N};$$

- для кожної з N реалізацій знайти

$$\eta_i(\xi_i) = \frac{\ln \xi_i}{\dfrac{1}{e-1}} = (e-1)\ln \xi_i\,, \qquad i = \overline{1, N};$$

- обчислити

$$\frac{1}{n}\sum_{i=1}^{N} \eta_i \approx I_0.$$

Запрограмувавши отриманий алгоритм, отримаємо наступні результати:

$$N = 100 \Rightarrow I_0 = 0.9390298002772277,$$
$$N = 10^3 \Rightarrow I_0 = 0.9956193974785688,$$
$$N = 10^6 \Rightarrow I_0 = 1.000226040347655.$$

Можемо бачити, що при збільшені N отримане методом імітаційного моделювання значення наближається до 1.

2.3 Точність методу

Нехай нам потрібно обчислити деяку невідому величину p, що відповідає ймовірності «успіху» певної події. Розглядаємо саме такий випадок, оскільки з ним будемо мати справу при імітації резервованої системи з відновленням («успіх» відповідатиме за те, що система не відмовила на заданому проміжку). Отже, виберемо випадкову величину ξ, розподілену за біноміальним розподілом $B(N,p)$. Розглянемо $\xi_1, \xi_2, \dots, \xi_N$ – незалежні копії випадкової величини ξ.

Нехай на основі N реалізацій випадкових величин $\xi_1, \xi_2, \dots, \xi_N$ методом імітаційного моделювання було знайдено $\hat{p}$ – оцінку величини p. Тоді

$$\Delta_m = |p - \hat{p}| = \left| p - \frac{1}{N}\sum_{i=1}^{N} \xi_i \right| = \left| \frac{\sum_{i=1}^{N} \xi_i - Np}{N} \right| = \left| \frac{\sum_{i=1}^{N} \xi_i - Np}{\sqrt{Npq}} \cdot \frac{\sqrt{pq}}{\sqrt{N}} \right|.$$

Візьмемо за рівень значущості $\alpha = 0.05$. Тоді треба обрати таке $\epsilon > 0$, щоб

$$P(\Delta_m < \epsilon) = P\left(\left| \frac{\sum_{i=1}^{N} \xi_i - Np}{\sqrt{Npq}} \cdot \frac{\sqrt{pq}}{\sqrt{N}} \right| < \epsilon \right) = P\left(\left| \frac{\sum_{i=1}^{N} \xi_i - Np}{\sqrt{Npq}} \right| < \frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) = 0.95.$$

Згідно інтегральній граничній теоремі Муавра-Лапласа [5, с. 356–357]:

$$P\left(\left| \frac{\sum_{i=1}^{N} \xi_i - Np}{\sqrt{Npq}} \right| < \frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) = \Phi\left(\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) - \Phi\left(-\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right)$$

$$= \Phi\left(\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) - \left(1 - \Phi\left(\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) \right) = 2\Phi\left(\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) - 1 \approx 0.95.$$

Звідки

$$\Phi\left(\frac{\epsilon\sqrt{N}}{\sqrt{pq}} \right) = \frac{0.95 + 1}{2} = 0.975 \Rightarrow \frac{\epsilon\sqrt{N}}{\sqrt{pq}} = U_{0.975} \Rightarrow \epsilon = U_{0.975} \frac{\sqrt{pq}}{\sqrt{N}},$$

де $U_{0.975}$ – квантиль рівня 0.975 стандартного нормального розподілу $N(0,1)$.

Отже, при рівні значущості $\alpha = 0.05$ отримали

$$\Delta_m = |m - \widehat{m}| \leq U_{0.975}\frac{\sqrt{pq}}{\sqrt{N}} \approx \ 2\frac{\sqrt{pq}}{\sqrt{N}} \leq \frac{1}{\sqrt{N}}.$$

3 ЕЛЕМЕНТИ ТЕОРІЇ НАДІЙНОСТІ

3.1 Основні елементи теорії надійності

Теорія надійності – це галузь статистики та інженерії, яка є важливою частиною процесу проектування і розробки, оскільки вона допомагає визначити ймовірність виникнення відмов і розробити стратегії для мінімізації їх наслідків.

Ось декілька ключових моментів з теорії надійності, з частиною яких ми будемо працювати в подальшому:

- *Надійність*: надійність визначає ймовірність, що система буде працювати без збоїв протягом визначеного періоду за визначених умов [6, с. 10]. Вона часто виражається як функція часу, з відмовами, які стають більш ймовірними з часом. Методи для визначення надійності системи можуть включати статистичний аналіз даних про відмову та прогнозування на основі моделей надійності.
- *Несправність*: несправність – це стан, коли система не виконує задані функції або виконує їх недостатньо ефективно [6, с. 10]. Несправності можуть бути пов'язані з відмовами обладнання, помилками програмного забезпечення, або проблемами взаємодії компонентів. Аналіз несправностей допомагає виявити причини відмов та розробити стратегії для їх усунення або мінімізації.
- *Час до відмови*: час до відмови вимірює період роботи системи від моменту запуску до моменту першої відмови. Час до відмови важливий для планування обслуговування та встановлення циклів заміни компонентів.
- *Інтенсивність відмов*: інтенсивність відмов – це кількість відмов, які виникають за одиницю часу. Цей показник допомагає визначити, наскільки часто можуть виникати проблеми з системою. Ця

інформація важлива для планування програм обслуговування і ремонту, а також для розробки стратегій з підвищення надійності.

3.2 Резервування та відновлення

Резервування та відновлення є важливими стратегіями, що забезпечують надійність та доступність систем.

Резервування є процесом, при якому використовуються додаткові, запасні компоненти або системи, щоб підтримувати роботу в разі відмови основних компонентів [6, с. 61-62]. Ефективне резервування залежить від декількох факторів, включаючи типи відмов, які відбуваються, їхню частоту, а також час та витрати на ремонт або заміну компонентів. Є різні стратегії резервування, які можна використовувати залежно від специфіки системи та вимог до надійності:

- *Гарячий резерв*: гарячий резерв включає постійно запущені запасні компоненти, які можуть негайно перейняти обробку у випадку відмови основних компонентів. Гаряче резервування забезпечує найвищий рівень доступності, але вимагає найбільших витрат на резервне обладнання та енергоспоживання.
- *Холодний резерв*: холодний резерв включає запасні компоненти, які є вимкненими або в стані очікування до моменту відмови основних компонентів. Це є найекономічнішою формою резервування, але вона може вимагати значного часу для активації резерву та відновлення роботи системи.
- *Очікувальний резерв* (також відомий як теплий резерв): очікувальний резерв включає запасні компоненти, які працюють з пониженою продуктивністю або в стані часткової готовності. У випадку відмови, ці компоненти можуть бути швидко активовані для відновлення повної роботи системи.

Відновлення – це процес повернення системи до працездатного стану після відмови [6, с. 10-11]. Він може включати в себе виправлення, ремонт або заміну несправних компонентів. Стратегії відновлення можуть включати в себе планування регулярного обслуговування, заміну компонентів після певного часу використання, або використання алгоритмів прогнозування для визначення оптимального моменту для обслуговування або заміни компонентів.

3.3 Дубльована система з відновленням

Однією з найпростіших резервованих систем з відновленням є дубльована система з відновленням (рис. 3.3).

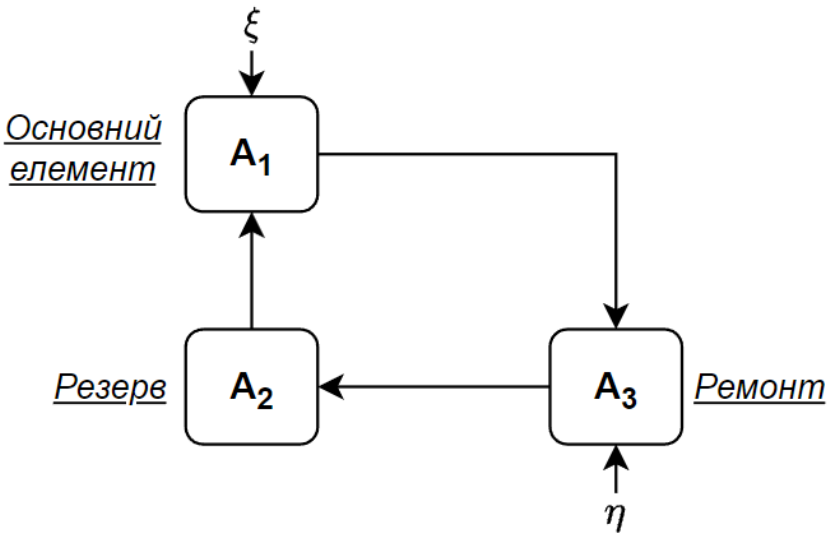

Рисунок 3.3 – Дубльована система з відновленням

Вона складається з трьох частин:

1) *робоча* (A_1): містить основний елемент;
2) *резервна* (A_2): містить резервний елемент, який у випадку відмови основного готовий його замінити;
3) *ремонтна* (A_3): містить ремонтну одиницю, яка здатна ремонтувати елемент, який вийшов з ладу.

Час безвідмовної роботи основного елемента ξ (див. рис. 3.3) та час ремонту η (див. рис. 3.3) задаються випадковими величинами з певними розподілами.

3.4 Резервована система з елементами декількох типів

Більш складним є випадок, коли наявні декілька типів робочих та резервних елементів (рис. 3.4).

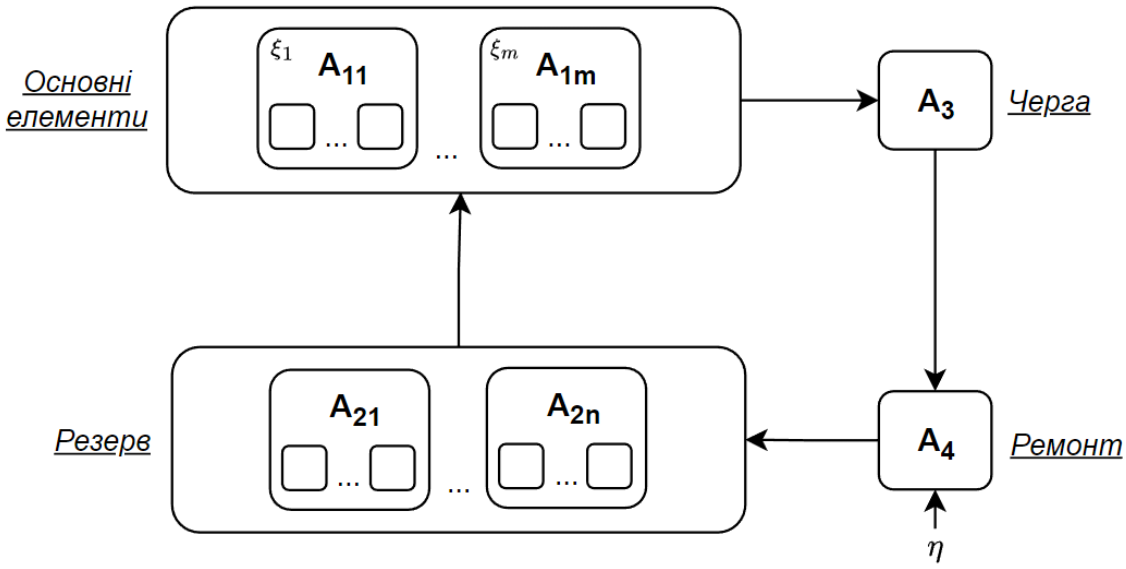

Рис. 3.4 – Резервована система з елементами декількох типів

В цьому випадку у нас додається нова частина – черга. Таким чином, система складається з чотирьох частин:

1) *робоча* ($A_{11}, \dots, A_{1m}$): містить декілька основних елементів кожного з типів;
2) *резервна* ($A_{21}, \dots, A_{2n}$): містить резервні елементи кожного типу, які у випадку відмови одного з основних елементів готові його замінити; заміна зламаного елемента певного типу може відбутися лише на резервний елемент того ж типу;
3) *ремонтна* (A_4): містить декілька ремонтних одиниць, які здатні ремонтувати елементи, що вийшли з ладу; ремонтні одиниці можуть бути спроектовані під окремий тип, або бути універсальними, тобто мати зможу ремонтувати елементи не залежно від їх типів;

4) *черга* (A_3): містить елементи, котрі вийшли з ладу та чекають на ремонт; переважно черга працює за принципом FIFO (first in, first out), тобто елемент, який швидше за інші вийшов з ладу, є першим в черзі на ремонт.

Час безвідмовної роботи основних елементів кожного типу $\xi_1, \dots, \xi_m$ (див. рис. 3.4) та час ремонту η (див. рис. 3.4) задаються випадковими величинами з певними розподілами.

4 ОЦІНЮВАННЯ НАДІЙНОСТІ РЕЗЕРВОВАНОЇ СИСТЕМИ З ВІДНОВЛЕННЯ

4.1 Постановка задачі

Розглядається резервована система з відновленням, яка містить робочі елементи трьох типів, кількість яких складає m_1, m_2, m_3 одиниць відповідно. Час безвідмовної роботи для кожного типу задається показниковим розподілом з параметрами $\lambda_1, \lambda_2, \lambda_3$ відповідно. У системі наявний холодний резерв, що містить n_1, n_2, n_3 резервних елементів кожного типу. Система має r універсальних ремонтних одиниць, які можуть відновлювати робочі елементи будь-якого з трьох типів. Час, потрібний для відновлення, задається випадковою величиною η, яка має довільний розподіл. Вважається, що система відмовляє, коли відмовляє один із робочих елементів, а усі резервні елементи відповідного типу перебувають в ремонті.

Нехай τ_c – випадкова величина, що відповідає за час відмови системи, тоді $P_c(t) = P\{\tau_C > t\}$ – надійність системи.

Знайти $P_c(t)$:

a) звичайним імітаційним моделюванням (методом Монте-Карло);
b) швидким імітаційним моделюванням.

4.2 Позначення

Нехай τ_1, τ_2, τ_3 – випадкові величини, що відповідають за час безвідмовної роботи елементів першого, другого та третього типів відповідно. Випадкова величина τ_1 має функцію розподілу $F_1(x)$, τ_2 – функцію розподілу $F_2(x)$, а τ_3 – функцію розподілу $F_3(x)$. Час ремонту позначатимемо випадковою величиною η з функцією розподілу $G(x)$. Нехай (τ_{k1}) – незалежні копії випадкової величини τ_1, (τ_{k2}) – незалежні копії випадкової величини τ_2, (τ_{k3}) – незалежні копії випадкової величини τ_3, а $(\eta_{k1}), (\eta_{k2}), (\eta_{k3})$ – незалежні копії випадкової величини η.

Введемо наступні об’єкти ($i = 1,2,3$):

- $A_{1i} = \{\nu_{1i}, \bar{\tau}_{1i}, \bar{\tau}_{2i}, \dots, \bar{\tau}_{\nu_{1i}i}\}$, де ν_{1i} – кількість працюючих елементів i-го типу, $\bar{\tau}_{ji}$ – залишковий час роботи j-го елемента i-го типу;
- $A_2 = \{\nu_{21}, \nu_{22}, \nu_{23}\}$, де ν_{2i} – кількість резервованих елементів i-го типу;
- $A_{3i} = \{\nu_{3i}, q_{1i}, q_{2i} \dots, q_{\nu_{3i}i}\}$, де ν_{3i} – кількість елементів i-го типу в черзі, q_{ji} – порядковий номер j-го елемента i-го типу в загальній черзі з усіх елементів;
- $A_{4i} = \{\nu_{4i}, \bar{\eta}_{1i}, \bar{\eta}_{2i}, \dots, \bar{\eta}_{\nu_{4i}i}\}$, де ν_{4i} – кількість елементів i-го типу в ремонті, $\bar{\eta}_{ji}$ – залишковий час ремонту j-го елемента i-го типу.

Таким чином задана система матиме вигляд як на рис. 4.2.

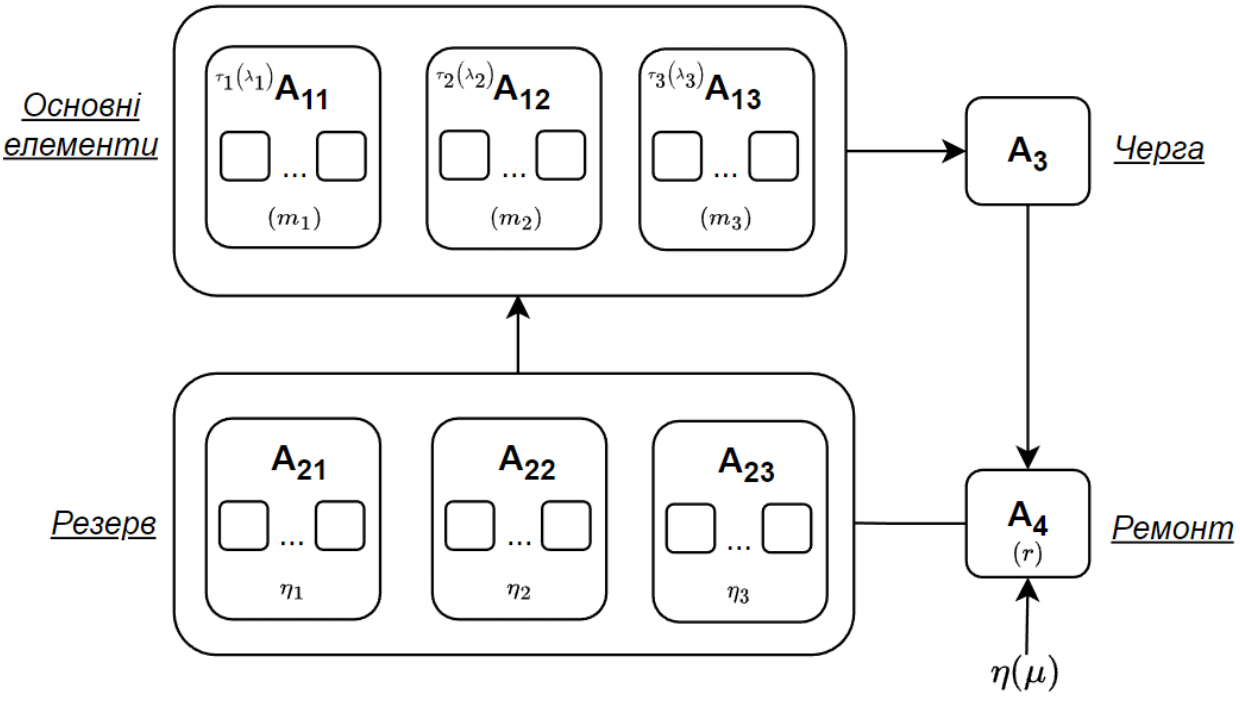

Рис. 4.2 – Резервована система з елементами трьох типів

Зазначимо, що черга працює за принципом FIFO (first in, first out). Тобто елемент, який зламався першим, буде відремонтовано в першу чергу, елемент, який зламався другим, буде відремонтовано другим і так далі. Також важливо

зазначити, що в об'єкті A_3 порядкові номери елементів в черзі q_{ji} зберігаються в порядку їх додавання, тобто в порядку зростання ($q_{1i} < q_{2i} < \cdots < q_{\nu_{3}i}$).

4.3 Імітаційне моделювання

Опишемо зміни станів заданої системи в часі на інтервалі $(0, t)$. Зміна станів в системі відбуватиметься лише в деякі моменти T_k, коли або відмовив один із основних елементів, або закінчила ремонт одна з ремонтних одиниць. Вважатимемо, що в інші моменти часу система не змінюється. Тому переходи в нашій системі будемо робити лише в моменти часу T_k. Зазначимо, що переходи слід здійснювати лише до моменту $T_{k+1} < t$.

Отже, початковий стан системи ($T_0 = 0$) матиме вигляд:

$$A_{11}^0 = \{m_1, \bar{\tau}_{11}, \bar{\tau}_{21}, \dots, \bar{\tau}_{m_1 1}\},\ \ A_{12}^0 = \{m_2, \bar{\tau}_{12}, \bar{\tau}_{22}, \dots, \bar{\tau}_{m_2 2}\},$$
$$A_{13}^0 = \{m_3, \bar{\tau}_{13}, \bar{\tau}_{23}, \dots, \bar{\tau}_{m_3 3}\},$$

$$A_2^0 = \{n_1, n_2, n_3\},$$

$$A_{31}^0 = A_{32}^0 = A_{33}^0 = \{0\},$$

$$A_{41}^0 = A_{42}^0 = A_{43}^0 = \{0\}.$$

Нехай зроблено k кроків. Тоді в момент T_k матимемо:

$$A_{11} = \left\{\nu_{11}^k, \bar{\tau}_{11}^k, \bar{\tau}_{21}^k, \dots, \bar{\tau}_{\nu_{11}^k 1}^k\right\},\ \ A_{12} = \left\{\nu_{12}^k, \bar{\tau}_{12}^k, \bar{\tau}_{22}^k, \dots, \bar{\tau}_{\nu_{12}^k 2}^k\right\},$$
$$A_{13} = \left\{\nu_{13}^k, \bar{\tau}_{13}^k, \bar{\tau}_{23}^k, \dots, \bar{\tau}_{\nu_{13}^k 3}^k\right\},$$

$$A_2 = \{\nu_{21}^k, \nu_{22}^k, \nu_{23}^k\},$$

$$A_{31} = \{\nu_{31}^k, q_{11}^k, q_{21}^k, \dots, q_{\nu_{31}1}^k\},\ \ A_{32} = \{\nu_{32}^k, q_{12}^k, q_{22}^k, \dots, q_{\nu_{31}2}^k\},$$
$$A_{33} = \{\nu_{33}^k, q_{13}^k, q_{23}^k, \dots, q_{\nu_{31}3}^k\},$$

$$A_{41} = \left\{\nu_{41}^k, \bar{\eta}_{11}^k, \bar{\eta}_{21}^k, \dots, \bar{\eta}_{\nu_{41}^k 1}^k\right\},\ \ A_{42} = \left\{\nu_{42}^k, \bar{\eta}_{12}^k, \bar{\eta}_{22}^k, \dots, \bar{\eta}_{\nu_{42}^k 2}^k\right\},$$
$$A_{43} = \left\{\nu_{43}^k, \bar{\eta}_{13}^k, \bar{\eta}_{23}^k, \dots, \bar{\eta}_{\nu_{43}^k 3}^k\right\}.$$

Перейдемо до опису станів системи при переході $T_k \to T_{k+1}$. Покладемо

$$\theta_k = \min\left(\bar{\tau}_{min}^{\,k}, \bar{\eta}_{min}^{\,k}\right),$$

де

$$\bar{\tau}_{min}^{\,k} = \min_{0<i\le 3} \min_{0<j\le \nu_{1i}^k} \bar{\tau}_{ji}^{\,k}, \qquad \bar{\eta}_{min}^{\,k} = \min_{0<i\le 3} \min_{0<j\le \nu_{4i}^k} \bar{\eta}_{ji}^{\,k}.$$

Тоді

$$T_{k+1} = T_k + \theta_k.$$

Обчислимо стани об'єктів $A_1 - A_4$ в момент часу T_{k+1}. Розглянемо два випадки:

а) $\theta_k = \bar{\tau}_{min}^{\,k} < \bar{\eta}_{min}^{\,k}$ (в момент часу T_{k+1} відбувається відмова робочого елемента)

Нехай при цьому $\bar{\tau}_{min}^{\,k} = \bar{\tau}_{r1}^{\,k}$ (відмовив r-ий робочий елемент 1-го типу). Тоді матимемо:

Робочі елементи

$$A_{11} = \left\{\nu_{11}^{k+1}, \bar{\tau}_{11}^{\,k+1}, \bar{\tau}_{21}^{\,k+1}, \dots, \bar{\tau}_{\nu_{11}^{k+1}1}^{\,k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{11}^k, \bar{\tau}_{11}^{\,k} - \theta_k, \dots, \bar{\tau}_{r-1\,1}^{\,k} - \theta_k, \bar{\tau}_{1}^{\,k}, \bar{\tau}_{r+1\,1}^{\,k} - \theta_k, \dots, \bar{\tau}_{\nu_{11}^k 1}^{\,k} - \theta_k\right\}, & \nu_{21}^k > 0 \\ \left\{\nu_{11}^k - 1, \bar{\tau}_{11}^{\,k} - \theta_k, \dots, \bar{\tau}_{r-1\,1}^{\,k} - \theta_k, \bar{\tau}_{r+1\,1}^{\,k} - \theta_k, \dots, \bar{\tau}_{\nu_{11}^k 1}^{\,k} - \theta_k\right\}, & \nu_{21}^k = 0, \end{cases}$$

$$A_{12} = \left\{\nu_{12}^{k+1}, \bar{\tau}_{12}^{\,k+1}, \bar{\tau}_{22}^{\,k+1}, \dots, \bar{\tau}_{\nu_{12}^{k+1}2}^{\,k+1}\right\} = \left\{\nu_{12}^k, \bar{\tau}_{12}^{\,k} - \theta_k, \bar{\tau}_{22}^{\,k} - \theta_k, \dots, \bar{\tau}_{\nu_{12}^k 2}^{\,k} - \theta_k\right\},$$

$$A_{13} = \left\{\nu_{13}^{k+1}, \bar{\tau}_{13}^{\,k+1}, \bar{\tau}_{23}^{\,k+1}, \dots, \bar{\tau}_{\nu_{13}^{k+1}3}^{\,k+1}\right\} = \left\{\nu_{13}^k, \bar{\tau}_{13}^{\,k} - \theta_k, \bar{\tau}_{23}^{\,k} - \theta_k, \dots, \bar{\tau}_{\nu_{13}^k 3}^{\,k} - \theta_k\right\}.$$

Резервні елементи

$$A_2^{k+1} = \left\{\nu_{21}^{k+1}, \nu_{22}^{k+1}, \nu_{23}^{k+1}\right\} = \left\{\max\left(0, \nu_{21}^k - 1\right), \nu_{22}^k, \nu_{23}^k\right\}.$$

Черга

$$A_{31} = \{\nu_{31}^{k+1}, q_{11}^{k+1}, q_{21}^{k+1}, \dots, q_{\nu_{31}1}^{k+1}\}$$

$$= \begin{cases} \{0\}, & \sum_{i=1}^{3} \nu_{4i}^{k} < r \\ \left\{\nu_{31}^{k} + 1, q_{11}^{k}, \dots, q_{\nu_{31}^{k}1}^{k}, q_{max}^{k} + 1\right\}, & \sum_{i=1}^{3} \nu_{4i}^{k} = r. \end{cases}$$

Ремонтні одиниці

$$A_{41} = \left\{\nu_{41}^{k+1}, \bar{\eta}_{11}^{k+1}, \bar{\eta}_{21}^{k+1}, \dots, \bar{\eta}_{\nu_{41}^{k+1}1}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{41}^{k}, \bar{\eta}_{11}^{k} - \theta_k, \bar{\eta}_{21}^{k} - \theta_k, \dots, \bar{\eta}_{\nu_{41}^{k}1}^{k} - \theta_k\right\}, & \sum_{i=1}^{3} \nu_{4i} = r \\ \{1, \bar{\eta}^{k}\ \}, & \sum_{i=1}^{3} \nu_{4i} < r, \end{cases}$$

$$A_{42} = \left\{\nu_{42}^{k+1}, \bar{\eta}_{12}^{k+1}, \bar{\eta}_{22}^{k+1}, \dots, \bar{\eta}_{\nu_{42}^{k+1}2}^{k+1}\right\} = \left\{\nu_{42}^{k}, \bar{\eta}_{12}^{k} - \theta_k, \bar{\eta}_{22}^{k} - \theta_k, \dots, \bar{\eta}_{\nu_{42}^{k}2}^{k} - \theta_k\right\},$$

$$A_{43} = \left\{\nu_{43}^{k+1}, \bar{\eta}_{13}^{k+1}, \bar{\eta}_{23}^{k+1}, \dots, \bar{\eta}_{\nu_{43}^{k+1}3}^{k+1}\right\} = \left\{\nu_{43}^{k}, \bar{\eta}_{13}^{k} - \theta_k, \bar{\eta}_{23}^{k} - \theta_k, \dots, \bar{\eta}_{\nu_{43}^{k}3}^{k} - \theta_k\right\}.$$

b) $\theta_k = \bar{\eta}_{min}^{k} < \bar{\tau}_{min}^{k}$ (в момент часу T_{k+1} завершується ремонт одна із ремонтних одиниць)

Нехай при цьому $\bar{\eta}_{min}^{k} = \bar{\eta}_{r1}^{k}$ (r-та ремонтна одиниця завершила ремонт робочого елементу 1-го типу). Тоді матимемо:

Робочі елементи

$$A_{11} = \left\{\nu_{11}^{k+1}, \bar{\tau}_{11}^{k+1}, \bar{\tau}_{21}^{k+1}, \dots, \bar{\tau}_{\nu_{11}^{k+1}1}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{11}^{k}, \bar{\tau}_{11}^{k} - \theta_k, \bar{\tau}_{21}^{k} - \theta_k, \dots, \bar{\tau}_{\nu_{11}^{k}1}^{k} - \theta_k\right\}, & \nu_{11}^{k} = m_1 \\ \left\{\nu_{11}^{k} + 1, \bar{\tau}_{11}^{k} - \theta_k, \bar{\tau}_{21}^{k} - \theta_k, \dots, \bar{\tau}_{\nu_{11}^{k}1}^{k} - \theta_k, \bar{\tau}_{1}^{k}\right\}, & \nu_{11}^{k} < m_1, \end{cases}$$

$$A_{12} = \left\{\nu_{12}^{k+1}, \bar{\tau}_{12}^{k+1}, \bar{\tau}_{22}^{k+1}, \dots, \bar{\tau}_{\nu_{12}^{k+1}2}^{k+1}\right\} = \left\{\nu_{12}^{k}, \bar{\tau}_{12}^{k} - \theta_k, \bar{\tau}_{22}^{k} - \theta_k, \dots, \bar{\tau}_{\nu_{12}^{k}2}^{k} - \theta_k\right\},$$

$$A_{13} = \left\{\nu_{13}^{k+1}, \bar{\tau}_{13}^{k+1}, \bar{\tau}_{23}^{k+1}, \dots, \bar{\tau}_{\nu_{13}^{k+1}3}^{k+1}\right\} = \left\{\nu_{13}^{k}, \bar{\tau}_{13}^{k} - \theta_k, \bar{\tau}_{23}^{k} - \theta_k, \dots, \bar{\tau}_{\nu_{13}^{k}3}^{k} - \theta_k\right\}.$$

Резервні елементи

$$A_2^{k+1} = \{\nu_{21}^{k+1}, \nu_{22}^{k+1}, \nu_{23}^{k+1}\} = \begin{cases} \{\nu_{21}^k + 1, \nu_{22}^k, \nu_{23}^k\}, & \nu_{11}^k = m \\ \{0, \nu_{22}^k, \nu_{23}^k\}, & \nu_{11}^k < m. \end{cases}$$

Черга

$$A_{31} = \left\{\nu_{31}^{k+1}, q_{11}^{k+1}, q_{21}^{k+1}, \dots, q_{\nu_{31}^{k+1}1}^{k+1}\right\}$$

$$= \begin{cases} \{\nu_{31}^k - 1, q_{21}^k - 1, \dots, q_{\nu_{31}1}^k - 1\}, & q_{11}^k = 1 \\ \left\{\nu_{31}^k, q_{11}^k - 1, q_{21}^k - 1, \dots, q_{\nu_{31}^k 1}^k - 1\right\}, & q_{11}^k \neq 1, \end{cases}$$

$$A_{32} = \left\{\nu_{32}^{k+1}, q_{12}^{k+1}, q_{22}^{k+1}, \dots, q_{\nu_{31}2}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{32}^k - 1, q_{22}^k - 1, \dots, q_{\nu_{32}^k 2}^k - 1\right\}, & q_{12}^k = 1 \\ \left\{\nu_{32}^k, q_{12}^k - 1, q_{22}^k - 1, \dots, q_{\nu_{32}^k 2}^k - 1\right\}, & q_{12}^k \neq 1, \end{cases}$$

$$A_{33} = \left\{\nu_{33}^{k+1}, q_{13}^{k+1}, q_{23}^{k+1}, \dots, q_{\nu_{31}3}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{33}^k - 1, q_{23}^k - 1, \dots, q_{\nu_{33}^k 3}^k - 1\right\}, & q_{13}^k = 1 \\ \left\{\nu_{33}^k, q_{13}^k - 1, q_{23}^k - 1, \dots, q_{\nu_{33}^k 3}^k - 1\right\}, & q_{13}^k \neq 1. \end{cases}$$

Ремонтні одиниці

$$A_{41} = \left\{\nu_{41}^{k+1}, \bar{\eta}_{11}^{k+1}, \bar{\eta}_{21}^{k+1}, \dots, \bar{\eta}_{\nu_{41}^{k+1}1}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{41}^k, \bar{\eta}_{11}^k - \theta_k, \bar{\eta}_{r-1\,1}^k - \theta_k, \bar{\eta}^k, \bar{\eta}_{r+1\,1}^k - \theta_k, \dots, \bar{\eta}_{\nu_{41}^k 1}^k - \theta_k\right\}, & q_{11}^k = 1 \\ \{\nu_{41}^k - 1, \bar{\eta}_{11}^k - \theta_k, \bar{\eta}_{r-1\,1}^k - \theta_k, \bar{\eta}_{r+1\,1}^k - \theta_k, \dots, \bar{\eta}_{\nu_{41}1}^k - \theta_k\}, & q_{11}^k \neq 1, \end{cases}$$

$$A_{42} = \left\{\nu_{42}^{k+1}, \bar{\eta}_{12}^{k+1}, \bar{\eta}_{22}^{k+1}, \dots, \bar{\eta}_{\nu_{42}^{k+1}2}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{42}^k + 1, \bar{\eta}_{12}^k - \theta_k, \bar{\eta}_{22}^k - \theta_k, \dots, \bar{\eta}_{\nu_{42}^k 2}^k - \theta_k, \bar{\eta}^k\right\}, & q_{12}^k = 1 \\ \left\{\nu_{42}^k, \bar{\eta}_{12}^k - \theta_k, \bar{\eta}_{22}^k - \theta_k, \dots, \bar{\eta}_{\nu_{42}^k 2}^k - \theta_k\right\}, & q_{12}^k \neq 1, \end{cases}$$

$$A_{43} = \left\{\nu_{43}^{k+1}, \bar{\eta}_{13}^{k+1}, \bar{\eta}_{23}^{k+1}, \dots, \bar{\eta}_{\nu_{43}^{k+1}3}^{k+1}\right\}$$

$$= \begin{cases} \left\{\nu_{43}^k + 1, \bar{\eta}_{13}^k - \theta_k, \bar{\eta}_{23}^k - \theta_k, \dots, \bar{\eta}_{\nu_{43}^k 3}^k - \theta_k, \bar{\eta}^k\right\}, & q_{13}^k = 1 \\ \left\{\nu_{43}^k, \bar{\eta}_{13}^k - \theta_k, \bar{\eta}_{23}^k - \theta_k, \dots, \bar{\eta}_{\nu_{43}^k 3}^k - \theta_k\right\}, & q_{13}^k \neq 1. \end{cases}$$

Звернемо увагу, що величини $\bar{\tau}_1^{\,k}, \bar{\eta}^{\,k}$ – це реалізації випадкових величин τ_1 та η, що відповідають за тривалість роботи робочого елемента першого типу та тривалість ремонту відповідно. Згідно з умовами задачі випадкова величина τ_1 має експоненціальний розподіл з параметром λ_1, а η – той же розподіл з параметром μ. Обидві величини отримані за допомогою алгоритму, описаного в розділі 1.

Для того, щоб знайти величину $P_c(t)$ – ймовірність відмови системи на інтервалі $(0, t)$, – будуємо N ($N \approx 10^4 - 10^6$) траєкторій на інтервалі $(0, t)$. Поведінка цих траєкторій задається переходами об'єктів $A_1 - A_4$, які були описані вище. Тобто будується зовнішній цикл по $j = \overline{1, N}$. Для j-ої траєкторії перевіряється умова:

$$\exists T_k < t: \quad \nu_{11}^k < m_1 \vee \nu_{12}^k < m_2 \vee \nu_{13}^k < m_3.$$

Якщо дана умова виконується, тобто система відмовляє в певний момент часу T_k, то кладемо $I_j = 1$ і виходимо із циклу по k. Після цього переходимо до побудови наступної $(j+1)$-ї траєкторії. Якщо ж дана умова не виконується (відмова відсутня), тобто

$$\forall k, T_k < t: \quad \nu_{11}^k = m_1 \ \wedge \ \nu_{12}^k = m_2 \ \wedge \ \nu_{13}^k = m_3.$$

то кладемо $I_j = 0$ і аналогічно переходимо до наступної траєкторії.

Таким чином, після закінченню циклу по j маємо індикатори відмов $I_j, \ \ j = \overline{1, N}$. Далі знаходимо

$$\hat{P}_{\text{відм}}(t) = \frac{1}{N} \sum_{j=1}^{N} I_j\,, \qquad \hat{P}_c(t) = 1 - \hat{P}_{\text{відм}}(t), \qquad P_c(t) \approx \hat{P}_c(t).$$

При рівні значущості $\alpha = 0.05$ матимемо наступну точність отриманих результатів:

$$\left|P_c(t) - \hat{P}_c(t)\right| \le 2\sqrt{\frac{pq}{N}} \approx 2\sqrt{\frac{\hat{p}\hat{q}}{N}} \le \frac{1}{\sqrt{N}}, \qquad \hat{p} = \hat{P}_c(t),\ \ q = \hat{P}_{\text{відм}}(t).$$

Наведемо результати імітаційного моделювання при різних параметрах та показниковому розподілі часу ремонту (таблиця 4.3).

Таблиця 4.3 – Приклади обрахунку надійності системи методом імітаційного моделювання

	1	2	3	4	5
N	1_000	100_000	1_000	100_000	10_000
m_1, m_2, m_3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
$\lambda_1, \lambda_2, \lambda_3$	1,3,2	1,3,2	1,3,2	1,3,2	2,3,2
n_1, n_2, n_3	3,4,5	3,4,5	3,4,5	3,4,5	3,4,5
r	4	4	4	4	1
μ	8	8	8	8	20
t	0.1	0.1	0.01	0.01	5
Надійність системи	1	0.99981	1	1	0.5505

Можна помітити, що для певних систем отримаємо нульову ймовірність відмови, що на перший погляд видається дивним. Справа в тому, що для високонадійних систем (наприклад, систем з ймовірністю відмови 10^{-6}) навіть при великих значеннях N можемо не натрапити на відмову, в результаті чого отримати не зовсім коректні результати (абсолютну надійність). Зокрема для таких випадків і було придумано швидке імітаційне моделювання, яке нівелює можливість “пропустити” відмову.

4.4 Швидке імітаційне моделювання

Основна ідея швидкого імітаційного моделювання полягає в імітації системи таким чином, щоб гарантовано натрапити на відмову [7, с. 222-228]. Таким чином, якщо при звичайному імітаційному моделюванні після кожної ітерації ми отримували 1, або 0, в залежності від того, чи відмовила системи, то при швидкому імітаційному модулюванні на кожній ітерації ми отримуємо умовну ймовірність того, що відмова таки відбудеться. Це досягається за допомогою поєднання деяких ймовірнісних формул (наприклад, формули повної ймовірності) з безпосереднім імітаційним моделюванням (метод Монте-Карло).

Нехай нас буде цікавити надійність системи на інтервалі $(0, t)$:

$$P_c(t) = P(\tau_c > t).$$

В даному випадку нам буде зручніше обчислювати величину $Q_c(t) = 1 - P_c(t)$ – ймовірність відмови системи.

Алгоритм матиме наступний вигляд: для досить великого N будується зовнішній цикл по $j = \overline{1, N}$, де на кожній ітерації знаходиться величина $\hat{Q}(j)$ (деяка груба оцінка $Q_c(t)$). Тоді при великих N матимемо:

$$\hat{Q}_c(t) = \frac{1}{N}\sum_{j=1}^{N} \hat{Q}_j \approx Q_c(t).$$

Розглянемо алгоритм більш детально. Для фіксованого j будуємо внутрішній цикл по $k = 0,1,2, \ldots$. Аналогічно до звичайного імітаційного моделювання будемо проходити лише особливі точки $0 = T_0 < T_1 < T_2, \ldots, < T_\nu < t$, де точка T_k, $k \geq 1$ – це або момент відмови одного із основних елементів, або момент закінчення ремонту однієї із ремонтних одиниць, а T_ν – момент відмови системи на j-ій траєкторії. В момент T_k обчислюється деяка ймовірність p_k, а після закінчення циклу по k знаходимо $\hat{Q}_j = \prod_{k=1}^{\nu-1} p_k$ – умовну ймовірність відмови системи при фіксованих значеннях (η_i) для j-ої

траєкторії. Зазначимо, що ν та p_k залежать від j, а точніше від реалізацій випадкових чисел на j-ій ітерації.

Опишемо детальніше проходження внутрішнього циклу по k.

Крок 0 (стан в момент $T_0 = 0$)

1. Обчислюємо ймовірність того, що на наступному кроці ми залишимося в заданому інтервалі $(0, t)$:

$$p_0 = P\left(\min\left(\tau_{11}, \dots, \tau_{m_1 1}, \tau_{12}, \dots, \tau_{m_2 2}, \tau_{13}, \dots, \tau_{m_3 3}\right) < t\right)$$
$$= 1$$
$$- P\left(\min\left(\tau_{11}, \dots, \tau_{m_1 1}, \tau_{12}, \dots, \tau_{m_2 2}, \tau_{13}, \dots, \tau_{m_3 3}\right) \geq t\right)$$
$$= 1 - P(\tau_{11} \geq t) \cdot \dots \cdot P\left(\tau_{m_1 1} \geq t\right) \cdot P(\tau_{12} \geq t) \cdot \dots$$
$$\cdot P\left(\tau_{m_2 2} \geq t\right) \cdot P(\tau_{13} \geq t) \cdot \dots \cdot P\left(\tau_{m_1 3} \geq t\right)$$
$$= 1 - \left(1 - F_1(t)\right)^{m_1} \cdot \left(1 - F_2(t)\right)^{m_2} \cdot \left(1 - F_3(t)\right)^{m_3}$$
$$= 1 - \left(1 - 1 + e^{-\lambda_1 t}\right)^{m_1} \cdot \left(1 - 1 + e^{-\lambda_2 t}\right)^{m_2}$$
$$\cdot \left(1 - 1 + e^{-\lambda_3 t}\right)^{m_3} = 1 - e^{-m_1\lambda_1 t} \cdot e^{-m_2\lambda_2 t} \cdot e^{-m_3\lambda_3 t}$$
$$= 1 - e^{-(m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3)t} = \tilde{F}(t),$$

де $\tilde{F}(x)$ – функція розподілу деякої випадкової величини, що задана показниковим розподілом з параметром $m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3$.

2. Моделюємо випадкову величину $\tilde{\tau}_0$ з функцією розподілу $\tilde{F}_0(x)$, яка побудована таким чином, щоб не вийти за проміжок (Δ_0, t):

$$\tilde{F}_0(x) = P(\tilde{\tau}_0 < x)$$

$$= P\big(\min\big(\tau_{11}, \dots, \tau_{m_1 1}, \tau_{12}, \dots, \tau_{m_2 2}, \tau_{13}, \dots, \tau_{m_3 3}\big)$$

$$< x \mid \min\big(\tau_{11}, \dots, \tau_{m_1 1}, \tau_{12}, \dots, \tau_{m_2 2}, \tau_{13}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)$$

$$= \frac{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < x \ \wedge \min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}$$

$$= \begin{cases} \dfrac{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < x\big)}{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}, & 0 < x < \Delta_0 \\ \dfrac{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}, & x \geq \Delta_0 \end{cases}$$

$$= \begin{cases} \dfrac{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < x\big)}{P\big(\min\big(\tau_{11}, \dots, \tau_{m_3 3}\big) < \Delta_0\big)}, & 0 < x < \Delta_0 \\ 1, & x \geq \Delta_0 \end{cases}$$

$$= \begin{cases} \dfrac{\tilde{F}(x)}{\tilde{F}(\Delta_0)}, & 0 < x < \Delta_0 \\ 1, & x \geq \Delta_0 \end{cases}, \qquad \Delta_0 = t - T_0 = t.$$

Під $\tilde{\tau}_0$ будемо розуміти найменший час безвідмовної роботи серед усіх елементів усіх трьох типів, тому не будемо моделювати час роботи для елементів кожного типу окремо, а зважатимемо лише на значення $\tilde{\tau}_0$.

3. Стан об’єктів системи:

$$A_1^0 = \{\tilde{\tau}_0\},$$
$$A_2^0 = \{n_1, n_2, n_3\},$$
$$A_{31}^0 = A_{32}^0 = A_{33}^0 = \{0\},$$
$$A_{41}^0 = A_{42}^0 = A_{43}^0 = \{0\}.$$

Крок 1 (стан в момент T_1)

1. $T_1 = T_0 + \tilde{\tau}_0$.
2. Обчислюємо

$$p_1 = P\left(\min\left(\tau_{11}, \dots, \tau_{m_3 3}\right) + T_1 < t\right)$$
$$= P\left(\min\left(\tau_{11}, \dots, \tau_{m_3 3}\right) < t - T_1\right) = \tilde{F}(t - T_1) = \tilde{F}(\Delta_1).$$

3. Моделюємо випадкову величину $\tilde{\tau}_1$ з функцією розподілу $\tilde{F}_1(x)$:

$$\tilde{F}_1(x) = \begin{cases} \dfrac{\tilde{F}(x)}{\tilde{F}(\Delta_1)}, & 0 < x < \Delta_0 \\ 1, & x \geq \Delta_0 \end{cases}, \qquad \Delta_1 = t - T_1.$$

4. Оскільки ми зважаємо лише на найменший час безвідмовної роботи $\tilde{\tau}_0$ без прив'язки до типу, нам потрібно обрати той тип, до якого відноситься елемент з найменшим часом безвідмовної роботи $\tilde{\tau}_0$, тобто той тип, який відмовив. Запишемо ймовірності, які відповідають обранню кожного із трьох типів:

$$P(\text{відмовив перший тип}) = \frac{m_1\lambda_1}{m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3},$$
$$P(\text{відмовив другий тип}) = \frac{m_2\lambda_2}{m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3},$$
$$P(\text{відмовив третій тип}) = \frac{m_3\lambda_3}{m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3}.$$

Для того, щоб обрати відповідний тип розіграємо наступний експеримент: генеруємо випадкову величину $\bar{\xi}$ з рівномірним розподілом на відрізку $[0, m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3]$. Припустимо, що $m_1\lambda_1 < m_2\lambda_2 < m_3\lambda_3$. Тоді, якщо $0 \leq \bar{\xi} < m_1\lambda_1$, обираємо перший тип, якщо $m_1\lambda_1 < \bar{\xi} < m_1\lambda_1 + m_2\lambda_2$ – другий, якщо $m_1\lambda_1 + m_2\lambda_2 < \bar{\xi} < m_2\lambda_2 \leq m_1\lambda_1 + m_2\lambda_2 + m_3\lambda_3$ – третій. Таким чином було обрано тип того елементу, який відмовив. Без обмеження загальності припустимо, що цим типом виявився перший тип. Тоді об'єкти системи матимуть вигляд:

$$A_1^1 = \{\tilde{\tau}_1\},$$
$$A_2^1 = \{n_1 - 1, n_2, n_3\},$$
$$A_{31}^1 = A_{32}^1 = A_{33}^1 = \{0\},$$
$$A_{41}^1 = \{1, \bar{\eta}_1\}, \qquad A_{42}^0 = A_{43}^0 = \{0\}.$$

Відзначимо, що на кроках $k = 0{,}1$ відмови системи не може бути, та перейдемо до кроку $k \geq 2$ (стан систем в момент T_k).

Крок $k \geq 2$ (стан в момент T_k)

Отже, на цьому кроці маємо $T_0, T_1, \dots, T_{k-1}$ та $p_0, p_1, \dots, p_{k-1}$. Нехай в момент T_{k-1} стани такі:

$$A_1^k = \{\tilde{\tau}_{k-1}\},$$

$$A_2^k = \left\{\nu_{21}^k, \nu_{22}^k, \nu_{23}^k\right\},$$

$$A_{31}^k = \left\{\nu_{31}^k, q_{11}^k, q_{21}^k, \dots, q_{\nu_{31}1}^k\right\}, \quad A_{32}^k = \left\{\nu_{32}^k, q_{12}^k, q_{22}^k, \dots, q_{\nu_{31}2}^k\right\},$$
$$A_{33}^k = \left\{\nu_{33}^k, q_{13}^k, q_{23}^k, \dots, q_{\nu_{31}3}^k\right\},$$

$$A_{41}^k = \left\{\nu_{41}^k, \bar{\eta}_{11}^k, \bar{\eta}_{21}^k, \dots, \bar{\eta}_{\nu_{41}^k 1}^k\right\}, \quad A_{42}^k = \left\{\nu_{42}^k, \bar{\eta}_{12}^k, \bar{\eta}_{22}^k, \dots, \bar{\eta}_{\nu_{42}^k 2}^k\right\},$$
$$A_{43}^k = \left\{\nu_{43}^k, \bar{\eta}_{13}^k, \bar{\eta}_{23}^k, \dots, \bar{\eta}_{\nu_{43}^k 3}^k\right\}.$$

Покладемо

$$\theta_k = \min\left(\tilde{\tau}_{k-1}, \bar{\eta}_{min}^k\right),$$

де

$$\bar{\eta}_{min}^k = \min_{0<i\leq 3} \min_{0<j\leq \nu_{4i}^k} \bar{\eta}_{ji}^k.$$

Тоді

$$T_k = T_{k-1} + \theta_k.$$

Обчислимо стани об’єктів $A_1 - A_4$ в момент часу T_k. Розглянемо два випадки:

а) $\theta_k = \tilde{\tau}_{k-1} < \bar{\eta}_{min}^k$ (в момент часу T_{k+1} відбувається відмова робочого елемента)

Нехай при цьому відмовив робочий елемент 1-го типу. Перевіряємо умову

$$\nu_{21}^{k} = 0.$$

Якщо умова виконується, тобто відсутні резервні елементи 1-го типу, то в момент T_k відбувається відмова системи. У цьому випадку кладемо

$$\hat{Q}_j = \prod_{k=1}^{\nu-1} p_k$$

і виходимо із циклу по k.

Якщо ж умова не виконується, то:

1. Обчислюємо $p_k = P\left(\min\left(\tau_{11}, \dots, \tau_{m_3 3}\right) + T_k < t\right) = F(\Delta_k)$.
2. Моделюємо випадкову величину $\tilde{\tau}_k$ з функцією розподілу $\tilde{F}_k(x)$:

$$\tilde{F}_k(x) = \begin{cases} \dfrac{\tilde{F}(x)}{\tilde{F}(\Delta_k)}, & 0 < x < \Delta_k \\ 1, & x \geq \Delta_k \end{cases}, \qquad \Delta_k = t - T_k.$$

3. Знаходимо стан об'єктів системи

   $A_1^k = \{\tilde{\tau}_k\}$,

   $A_2^k = \left\{\nu_{21}^{k-1} - 1, \nu_{22}^{k}, \nu_{23}^{k}\right\}$,

   A_3^k та A_4^k аналогічно звичайному імітаційному моделюванню.

b) $\theta_k = \bar{\eta}_{min}^{k} < \tilde{\tau}_{k-1}$ (в момент часу T_{k+1} завершується ремонт одна із ремонтних одиниць)

Нехай при цьому відмовив робочий елемент 1-го типу. Тоді:

1. Обчислюємо $p_k = 1$
2. Знаходимо стан об'єктів системи

   $A_1^k = \{\tilde{\tau}_k - \theta_k\}$,

   $A_2^k = \left\{\nu_{21}^{k-1} + 1, \nu_{22}^{k}, \nu_{23}^{k}\right\}$,

   A_3^k та A_4^k аналогічно звичайному імітаційному моделюванню.

Після закінчення циклу по $j = \overline{1, N}$ знаходимо оцінку ймовірності відмови системи на інтервалі $(0, t)$ за формулою:

$$q = Q_c(t) \approx \hat{Q}_c(t) = \hat{q}.$$

Реальна точність наближення при рівні значущості $\alpha = 0.05$ матиме вигляд:

$$\left|Q_c(t) - \hat{Q}_c(t)\right| = |q - \hat{q}| \leq 2\sqrt{\frac{pq}{N}} \approx 2\sqrt{\frac{\hat{p}\hat{q}}{N}} \leq \frac{1}{\sqrt{N}}, \qquad \hat{q} = \hat{Q}_c(t), \hat{p} = 1 - \hat{q}.$$

Порівняємо результати, отримані звичним імітаційним моделюванням (ІМ) та швидким імітаційним моделюванням (ШІМ) (таблиця 4.4).

Таблиця 4.4 – Порівняння обрахунку надійності системи методами безпосереднього та швидкого імітаційного моделювання

	1	2	3	4	5
N	1_000	100_000	1_000	100_000	10_000
m_1, m_2, m_3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
$\lambda_1, \lambda_2, \lambda_3$	1,3,2	1,3,2	1,3,2	1,3,2	2,3,2
n_1, n_2, n_3	3,4,5	3,4,5	3,4,5	3,4,5	3,4,5
r	4	4	4	4	1
μ	8	8	8	8	20
t	0.1	0.1	0.01	0.01	5
Надійність системи (ІМ)	1	0.99981	1	1	0.5505
Надійність системи(ШІМ)	0.9998549687357245	0.9998625085606804	0.999999996140757	0.999999993524611 2	0.5466063277824706

ВИСНОВКИ

В ході кваліфікаційної роботи методами безпосереднього та швидкого імітаційного моделювання було розв’язано задачу про знаходження надійності резервованої системи з відновленням, що містить робочі та резервні елементи трьох різних типів.

Результати, отримані в процесі моделювання системи з різними тестовими параметрами, свідчать про переваги застосування швидкого імітаційного моделюванню над безпосереднім, зокрема у випадку високонадійних систем.

Отримані алгоритми обчислення надійності можна використовувати для вдосконалення наявних систем, наприклад, в таких сферах, як:

- телекомунікаційні мережі: для балансування рівня резервування та ефективності обслуговування, з метою забезпечення безперебійної роботи мережі;
- дата-центри: для оптимізації запасу обладнання, розрахунку оптимальної кількості ремонтних станцій та мінімізації часу простою системи.

Отримані алгоритми також можна використати, як основу для розв’язування більш складних резервованих систем з відновленням (наприклад, при збільшені кількості типів основних та резервних елементів, заміни універсальних ремонтних одиниць на спеціалізовані одиниці, котрі здатні ремонтували лише елементи конкретного типу, зміна розподілу часу безвідмовної роботи).

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Symposium on large-scale digital calculating machinery (2nd 1949 Harvard University). Proceedings of a second symposium on large-scale digital calculating machinery. Cambridge, Massachusetts : Harvard University Press, 1951. 393 p.
2. Учасники проектів Вікімедіа. Закон великих чисел – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Закон_великих_чисел.
3. Учасники проектів Вікімедіа. Центральна гранична теорема – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Центральна_гранична_теорема.
4. Gudmundsson T. Rare-event simulation with Markov chain Monte Carlo : doctoral thesis. 2015. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-157522.
5. Dodge Y. The concise encyclopedia of statistics. New York : Springer, 2008. 616 p.
6. Основи теорії надійності технічних систем: навч. посіб. / О. М. Соболь та ін. НУЦЗУ, 2015. 133 с.
7. Kovalenko I. N., Kuznetzov N. Y., Pegg P. A. Mathematical theory of reliability of time dependent systems with practical applications. Wiley & Sons, Incorporated, John, 2000. 316 p.

ДОДАТОК А

Код реалізації методу безпосереднього імітаційного моделювання:

```
import math
from functools import reduce

import numpy as np


def calc_failure_probability(t: float,
                             working_elements_count: list[int],
                             working_time_parameters: list[float],
                             reserved_elements_count_by_type: list[int],
                             repair_size: int,
                             repairing_time_parameter: float,
experiment_count=100_000) -> float:
    success_count = 0
    for i in range(experiment_count):
        success_count += __emulate(t,
                                   working_elements_count,
                                   working_time_parameters,
                                   reserved_elements_count_by_type,
                                   repair_size,
                                   repairing_time_parameter)

    return 1 - success_count / experiment_count


def __emulate(t: float,
              working_elements_count_by_type: list[int],
              working_time_parameters: list[float],
              reserved_elements_count_by_type: list[int],
              repair_size: int,
              repairing_time_parameter: float) -> bool:
    # A_1_i
    working_elements = [
        [__generate_exponential_sample(working_time_parameters[0]) for _ in
range(working_elements_count_by_type[0])],
        [__generate_exponential_sample(working_time_parameters[1]) for _ in
range(working_elements_count_by_type[1])],
        [__generate_exponential_sample(working_time_parameters[2]) for _ in
range(working_elements_count_by_type[2])]
    ]

    # A_2
    reserved_elements_types = [reserved_elements_count_by_type[0],
                               reserved_elements_count_by_type[1],
                               reserved_elements_count_by_type[2]]

    # A_3_i
    queue_for_repair = [[], [], []]

    # A_4_i
    repairing_elements = [[], [], []]

    time_spent = 0
    while True:
        min_working_time = min([min(working_elements[i], default=math.inf)
for i in range(3)])
```

```
        min_repair_time = min([min(repairing_elements[i], default=math.inf)
for i in range(3)])

        theta = min(min_working_time, min_repair_time)

        time_spent += theta

        if time_spent > t:
            return True

        if theta == min_working_time:
            failure_type, failure_index =
__get_min_element_indexes(working_elements)

            # A_1_i
            if reserved_elements_types[failure_type] == 0:
                return False

            for i in range(3):
                for j in range(len(working_elements[i])):
                    working_elements[i][j] -= theta
            failure_working_parameter = working_time_parameters[failure_type]
            working_elements[failure_type][failure_index] =
__generate_exponential_sample(failure_working_parameter)

            # A_2
            reserved_elements_types[failure_type] -= 1

            # A_3_i
            repairing_elements_count = reduce(lambda count, l: count +
len(l), repairing_elements, 0)

            if repairing_elements_count == repair_size:
                queue_for_repair[failure_type] +=
[max([max(queue_for_repair[i], default=0) for i in range(3)]) + 1]

            # A_4_i
            for i in range(3):
                for j in range(len(repairing_elements[i])):
                    repairing_elements[i][j] -= theta

            if repairing_elements_count < repair_size:
                repairing_elements[failure_type] +=
[__generate_exponential_sample(repairing_time_parameter)]
        else:
            repaired_type, _ = __get_min_element_indexes(repairing_elements)

            # A_1_i
            for i in range(3):
                for j in range(len(working_elements[i])):
                    working_elements[i][j] -= theta

            # A_2
            reserved_elements_types[repaired_type] += 1

            # A_3_i
            queue_first_type = -1
            for i in range(3):
                for j in range(len(queue_for_repair[i])):
                    if queue_for_repair[i][j] == 1:
                        queue_first_type = i

            if queue_first_type != -1:
```

```
                queue_for_repair[queue_first_type].remove(1)

            for i in range(3):
                for j in range(len(queue_for_repair[i])):
                    queue_for_repair[i][j] -= 1

            # A_4_i
            repairing_elements[repaired_type].remove(theta)
            for i in range(3):
                for j in range(len(repairing_elements[i])):
                    repairing_elements[i][j] -= theta

            if queue_first_type != -1:
                repairing_elements[queue_first_type] +=
[__generate_exponential_sample(repairing_time_parameter)]


def __generate_exponential_sample(lambda_) -> float:
    return np.random.exponential(scale=1 / lambda_)


def __get_min_element_indexes(list_: list[list]) -> tuple[int, int]:
    min_value = math.inf
    min_indexes = None

    for i in range(len(list_)):
        for j in range(len(list_[i])):
            if list_[i][j] < min_value:
                min_value = list_[i][j]
                min_indexes = (i, j)

    return min_indexes
```

Посилання на GitHub: https://github.com/maxym-ko/thesis/blob/main/custom_system/simple_modeling.py

ДОДАТОК Б

Код реалізації методу швидкого імітаційного моделювання:

```
import math
from functools import reduce

import numpy as np


def calc_failure_probability(t: float,
                             working_elements_count_by_type: list[int],
                             working_time_parameters: list[float],
                             reserved_elements_count_by_type: list[int],
                             repair_size: int,
                             repairing_time_parameter: float,
experiment_count=100_000) -> float:
    fail_probability_sum = 0
    for i in range(experiment_count):
        fail_probability_sum += __emulate(t,
                                          working_elements_count_by_type,
                                          working_time_parameters,
                                          reserved_elements_count_by_type,
                                          repair_size,
                                          repairing_time_parameter)

    return fail_probability_sum / experiment_count


def __emulate(t: float,
              working_elements_count_by_type: list[int],
              working_time_parameters: list[float],
              reserved_elements_count_by_type: list[int],
              repair_size: int,
              repairing_time_parameter: float) -> float:
    working_time_parameters_sum = 0
    for i in range(len(working_time_parameters)):
        working_time_parameters_sum += working_elements_count_by_type[i] *
working_time_parameters[i]

    delta_k = t

    u = __generate_uniform_sample()
    tau_k = - (1 / working_time_parameters_sum) * math.log(1 - u + u *
math.exp(-working_time_parameters_sum * delta_k))

    p_k = 1 - math.exp(-working_time_parameters_sum * delta_k)

    # A_1
    min_working_elements_time = tau_k

    # A_2
    reserved_elements_types = [reserved_elements_count_by_type[0],
                               reserved_elements_count_by_type[1],
                               reserved_elements_count_by_type[2]]

    # A_3_i
```

```
    queue_for_repair = [[], [], []]

    # A_4_i
    repairing_elements = [[], [], []]

    t_k = 0
    while True:
        min_repairing_elements_time = min(min(repairing_elements[i],
default=math.inf) for i in range(3))
        theta = min(min_working_elements_time, min_repairing_elements_time)

        t_k += theta
        if theta == min_working_elements_time:
            failure_type =
__random_weighted_index_selection(working_time_parameters,
working_elements_count_by_type)

            if reserved_elements_types[failure_type] == 0:
                return p_k

            delta_k = t - t_k

            u = __generate_uniform_sample()
            tau_k = - (1 / working_time_parameters_sum) * math.log(1 - u + u
* math.exp(-working_time_parameters_sum * delta_k))

            p_k *= 1 - math.exp(-working_time_parameters_sum * delta_k)

            # A_1
            min_working_elements_time = tau_k

            # A_2
            reserved_elements_types[failure_type] -= 1

            # A_3_i
            repairing_elements_count = reduce(lambda count, l: count +
len(l), repairing_elements, 0)

            if repairing_elements_count == repair_size:
                queue_for_repair[failure_type] +=
[max([max(queue_for_repair[i], default=0) for i in range(3)]) + 1]

            # A_4_i
            for i in range(3):
                for j in range(len(repairing_elements[i])):
                    repairing_elements[i][j] -= theta

            if repairing_elements_count < repair_size:
                repairing_elements[failure_type] +=
[__generate_exponential_sample(repairing_time_parameter)]
        else:
            repaired_type = -1
            for i, repairing_element in enumerate(repairing_elements):
                if min_repairing_elements_time in repairing_element:
                    repaired_type = i
                    break

            # A_1
            min_working_elements_time -= theta

            # A_2
            reserved_elements_types[repaired_type] += 1
```

```
            # A_3_i
            queue_first_type = -1
            for i in range(3):
                for j in range(len(queue_for_repair[i])):
                    if queue_for_repair[i][j] == 1:
                        queue_first_type = i

            if queue_first_type != -1:
                queue_for_repair[queue_first_type].remove(1)

            for i in range(3):
                for j in range(len(queue_for_repair[i])):
                    queue_for_repair[i][j] -= 1

            # A_4_i
            repairing_elements[repaired_type].remove(theta)
            for i in range(3):
                for j in range(len(repairing_elements[i])):
                    repairing_elements[i][j] -= theta

            if queue_first_type != -1:
                repairing_elements[queue_first_type] +=
[__generate_exponential_sample(repairing_time_parameter)]


def __random_weighted_index_selection(elements: list, multiplicators: list):
    elements_sum = 0
    for i in range(len(elements)):
        elements_sum += multiplicators[i] * elements[i]

    select_probabilities = [multiplicators[i] * elements[i] / elements_sum
for i in range(len(elements))]

    random_uniform = np.random.uniform(high=sum(select_probabilities))

    selected_index = 0
    current_segment = select_probabilities[0]
    for i in range(1, len(select_probabilities)):
        if random_uniform > current_segment:
            selected_index = i
        current_segment += select_probabilities[i]

    return selected_index


def __generate_exponential_sample(lambda_) -> float:
    return np.random.exponential(scale=1 / lambda_)


def __generate_uniform_sample() -> float:
    return np.random.uniform()
```

Посилання на GitHub: https://github.com/maxym-ko/thesis/blob/main/custom_system/rapid_modeling.py