# РОЗДІЛ 4

Звісно, ось проєкт четвертого розділу, який логічно завершує опис вашої роботи, поєднуючи теорію з практичною реалізацією.

## РОЗДІЛ 4. ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗНОГО КЕРУВАННЯ ПРОЦЕСОМ ЗБАГАЧЕННЯ ЗАЛІЗНОЇ РУДИ НА ОСНОВІ ЯДРОВИХ МОДЕЛЕЙ

Цей розділ присвячено опису програмної архітектури та інструментальних засобів, використаних для реалізації розробленої системи модельного предиктивного керування. Метою є демонстрація того, як теоретичні алгоритми були втілені у гнучкий та масштабований програмний комплекс для симуляції та дослідження.

### 4.1. Загальна схема системи предиктивного керування

Програмна реалізація системи виконана на мові програмування

Ключові компоненти технологічного стеку включають:

• **NumPy** та **Pandas:** для ефективної роботи з багатовимірними масивами даних та часовими рядами, що є основою для всіх обчислень.

• **Scikit-learn:** для реалізації базових алгоритмів ядрових моделей (KRR, GPR, SVR) та інструментів для валідації (наприклад, RandomizedSearchCV).

• **CVXPY:** для декларативного формулювання та розв'язання задачі опуклої оптимізації, що лежить в основі MPC контролера1111. Ця бібліотека дозволяє описати цільову функцію та обмеження у високорівневому вигляді, автоматично транслюючи їх у стандартну форму, що розв'язується спеціалізованими солверами (наприклад, OSQP).

Загальна архітектура програмного комплексу відображає функціональну схему, описану в попередньому розділі, де кожен логічний блок системи реалізовано як окремий програмний модуль22.

### 4.2. Постановка вимог до схеми реалізації системи керування

Основною вимогою при розробці була

Система складається з чітко визначених, слабко зв'язаних модулів:

• data\_gen.py**:** Відповідає виключно за генерацію реалістичних даних, імітацію "реального об'єкта" та аномалій4444.

• model.py**:** Інкапсулює різні стратегії прогнозування (KRR, GPR, SVR) за єдиним інтерфейсом-фасадом KernelModel5555. Це дозволяє легко замінювати одну модель на іншу без змін у коді контролера.

• objectives.py**:** Містить класи цільових функцій, що дозволяє гнучко змінювати критерії оптимізації для MPC6666.

• ekf.py**:** Реалізує логіку розширеного фільтра Калмана як окремий, незалежний компонент для оцінки стану7.

• mpc.py**:** Є ядром системи, що інтегрує будь-яку модель та цільову функцію, які відповідають визначеному інтерфейсу8888.

• sim.py**:** Виступає в ролі оркестратора, який поєднує всі компоненти в єдиний симуляційний цикл, керує потоками даних та збирає результати для аналізу9999.

Такий підхід не лише спрощує тестування та модифікацію системи, але й дозволяє легко розширювати її функціонал у майбутньому.

### 4.3. Апробація технічного рішення

Надійність та валідність отриманих у ході симуляції результатів забезпечується двома ключовими аспектами реалізації:

1. **Відтворюваність експериментів:** У всіх модулях, що використовують стохастичні процеси (генерація даних, пошук гіперпараметрів), застосовується фіксоване зерно для генератора випадкових чисел (seed). Це гарантує, що результати будь-якого експерименту можуть бути точно відтворені, що є фундаментальною вимогою для наукових досліджень.

2. **Обчислювальна ефективність:** Незважаючи на те, що розв'язання задачі оптимізації на кожному кроці є обчислювально складною операцією, використання бібліотеки **CVXPY** дозволяє значно прискорити цей процес. Завдяки механізму параметризації задачі, структура оптимізаційної проблеми компілюється лише один раз при ініціалізації контролера. На кожному наступному кроці оновлюються лише числові значення параметрів (поточний стан, коефіцієнти лінеаризованої моделі), що суттєво зменшує накладні витрати.

Таким чином, розроблений програмний комплекс є не просто реалізацією теоретичних алгоритмів, а повноцінним, робастним та верифікованим симуляційним стендом, придатним для проведення комплексних досліджень систем предиктивного керування.