Настанова  
щодо використання

комп’ютерної програми „Інструментальна програма для моделювання та дослідження децентралізованих систем керування”

Комп'ютерна програма створює зручний інтерфейс для моделювання та дослідження децентралізованих розподілених систем керування (РСК), імітує функції координатора локальної системи керування (ЛСК) і обмін даними з координаторами інших ЛСК децентралізованої РСК. Децентралізована РСК складається з n ЛСК, які інформаційно взаємодіють між собою. Локальні системи керують окремими частинами розподіленого об'єкта, причому ці частини об'єкта теж взаємодіють між собою на фізичному рівні - обмінюються потоками ресурсів. Кожна локальна система керування (ЛСК) має регулятор, сенсор і блок координації з іншими ЛСК – «координатор». Основні функції координатора: моделювання стану системи; оцінювання параметрів; прогнозування параметрів; оптимізація уставки.

Програма використовує файл конфігурації (json) і глобальну структуру даних типу "черга". Структура кожного елемента черги (вектора) така: n значень поточного стану частин об'єкта; n значень бажаного стану частин об'єкта; n значень вхідного параметра; n значень уставки для ЛСК кожної частини об'єкта керування; 1 значення стану оточуючого середовища.

Після запуску програми з’явиться поле для вводу шляху до файлу конфігурації (Рис.1).

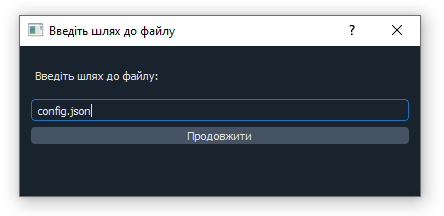


Рис. 1

Після завантаження файла конфігурації на екрані з’явиться черга (Рис.2), де буде показано реальний стан ЛСК, бажаний стан ЛСК, стан інших ЛСК і стан навколишнього середовища.

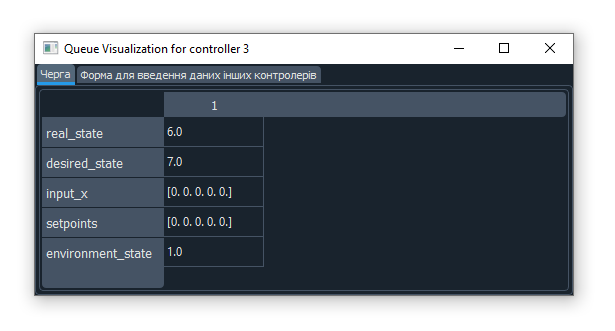


Рис. 2

Також на екрані з’являється форма для введення даних, які у реальній системі надходять від інших координаторів (Рис.3). Потрібно ввести бажаний стан ЛСК, стан інших ЛСК, і стан навколишнього середовища.

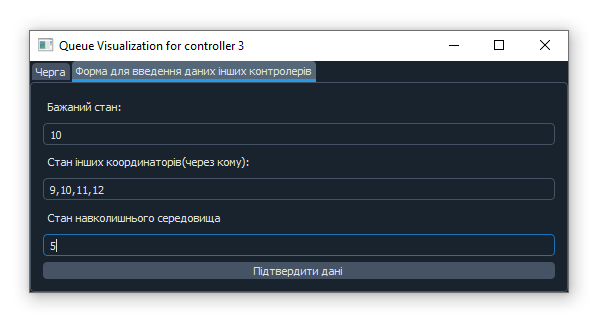


Рис. 3

Після підтвердження запускається імітація роботи координатора, виконуються всі його функції відповідно до алгоритму координації і в черзі з’являться нові дані (Рис.4).

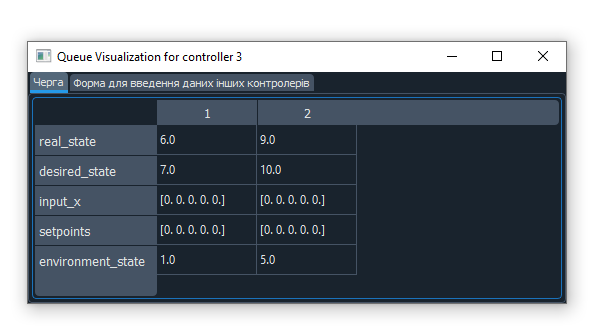


Рис.4

Програма написана на мові Python, з використанням сторонніх бібліотек PyQt5, NumPy, qdarkstyle. Програма може виконуватися на платформах Windows, Linux, MacOS.

Фрагмент лістингу

комп’ютерної програми

„Інструментальна програма для моделювання та дослідження децентралізованих систем керування”

class Coordinator(ABC):

def \_\_init\_\_(self, config\_file, coordinator\_index, filename):

# Збережіть ім'я файлу для зберігання даних

self.filename = filename

# Завантаження конфігурації

with open(config\_file) as f:

config = json.load(f)

# .................................................

class Setpoint(Coordinator):

def calculate\_setpoint\_for\_LSC(self):

# Отримання даних з черги

data = self.data\_queue.get()

current\_states = data[:self.num\_parts]

desired\_states = data[self.num\_parts : 2\*self.num\_parts]

input\_x = data[2\*self.num\_parts : 3\*self.num\_parts]

previous\_setpoints = data[3\*self.num\_parts : 4\*self.num\_parts]

environment\_state = data[-1]

# Приклади методів координації (з використанням розпакованих даних)

# 1. Модель системи

system\_model = self.\_system\_model(current\_states, input\_x, environment\_state)

# 2. Оцінювання параметрів

estimated\_parameters = self.\_estimate\_parameters(system\_model, current\_states, previous\_setpoints)

# 3. Прогнозування параметрів

predicted\_parameters = self.\_predict\_parameters(estimated\_parameters)

# 4. Оптимізація уставки

optimized\_setpoint = self.\_optimize\_setpoint(system\_model, predicted\_parameters, desired\_states)

# 5. Критерій оптимальності (перевірка)

if not self.\_check\_optimality\_criteria(optimized\_setpoint, system\_model, desired\_states):

# ... обробка ситуації, коли критерій оптимальності не виконаний

pass

return optimized\_setpoint

def \_system\_model(self, current\_states, input\_x, environment\_state):

# ... реалізація моделі системи

return randint(1, 3)

def \_estimate\_parameters(self, system\_model, current\_states, previous\_setpoints):

# ... реалізація оцінювання параметрів

return randint(1, 3)

def \_predict\_parameters(self, estimated\_parameters):

# ... реалізація прогнозування параметрів

return randint(1, 3)

def \_optimize\_setpoint(self, system\_model, predicted\_parameters, desired\_states):

# ... реалізація оптимізації уставки

return randint(1, 3)

def \_check\_optimality\_criteria(self, setpoint, system\_model, desired\_states):

# ... реалізація перевірки критерію оптимальності

return randint(1, 3)