ДОДАТОК А

Код програми численного моделювання децентралізованого керування групою колаборативних роботів-маніпуляторів у єдиній робочій зоні

```
import numpy as np
import matplotlib.pyplot as plt

# ----------------------
# Параметри симуляції
# ----------------------
GRID_SIZE = 50                # розмір дискретної карти для SDF/візуалізації
DT = 0.1                      # крок інтегрування (s)
T_TOTAL = 40.0                # загальний час симуляції (s)
STEPS = int(T_TOTAL / DT)
NUM_ROBOTS = 4
MAX_SPEED = 4.0               # м/с, обмеження швидкості
ROBOT_RADIUS = 0.8            # ефективний радіус робота (м)
INTER_ROBOT_BUFFER = 0.2      # додатковий буфер (м)
CBF_GAMMA = 2.0               # параметр CBF (швидкість відштовхування)
PROJECT_ITERS = 6             # кількість ітерацій послідовних проекцій

# консенсус
COMM_RADIUS = 12.0            # радіус зв'язку у одиницях карти
LAMBDA_XI = 0.8               # вага узгодження швидкостей (0..1)

# перешкоди (окружності) -- формат (x, y, r)
OBSTACLES = [
    (18, 26, 5),
    (35, 12, 4),
```

```
    (30, 35, 6),
]

# початкові/цільові позиції роботів (в координатах карти)
starts = np.array([[3.0, 3.0],
                   [3.0, 46.0],
                   [46.0, 3.0],
                   [20.0, 40.0]])
goals = np.array([[46.0, 46.0],
                  [46.0, 3.0],
                  [3.0, 46.0],
                  [40.0, 6.0]])

# Модель людини (динамічна перешкода)
# початкова позиція та проста модель руху (довільна траєкторія)
human_pos = np.array([25.0, 25.0])
human_vel = np.array([0.05, 0.07])    # одиниці карти / s (швидкість людини у вимірі карти)
R0 = 2.0                        # базовий радіус зони комфорту (м/од.)
ALPHA = 1.5                     # вплив швидкості людини на радіус

# коваріація (анізотропія) для Mahalanobis (2x2)
human_Sigma = np.array([[3.0, 0.5],
                        [0.5, 2.0]])

# SDF (вираховуємо евклідову відстань до найближчої перешкоди на дискретній сітці)
def compute_sdf(grid_size, obstacles):
    # Для кожної клітини знаходимо мінімум евклідової відстані до центрів колових перешкод minus their radius
```

```
    sdf = np.full((grid_size, grid_size), 1e6, dtype=float)
    xs = np.arange(grid_size)
    ys = np.arange(grid_size)
    X, Y = np.meshgrid(xs, ys, indexing='ij')
    pts = np.stack([X, Y], axis=-1)  # shape (grid,grid,2)
    for ox, oy, r in obstacles:
        d = np.sqrt((X - ox) ** 2 + (Y - oy) ** 2) - r
        sdf = np.minimum(sdf, d)
    # outside obstacles distances positive; inside negative (if any inside)
    return sdf

SDF = compute_sdf(GRID_SIZE, OBSTACLES)

# Функції допоміжні

def norm(v):
    return np.linalg.norm(v)

def unit(v):
    n = norm(v)
    return v / n if n > 1e-9 else np.zeros_like(v)

def R_safe(t_speed):
    """Адаптивний радіус зони комфорту людини"""
    return R0 + ALPHA * t_speed

def mahalanobis_h(p, pH, Sigma, R_s):
    """CBF h for human ellipsoid: h = (p-pH)^T Sigma^{-1} (p-pH) - R_s^2"""
    d = p - pH
    return float(d.T @ np.linalg.inv(Sigma) @ d - R_s ** 2)
```

```
    def sdf_at_point(p):
        """інтерполяція SDF (білайнейр) для неповних координат"""
        x, y = p
        # clamp
        x = np.clip(x, 0, GRID_SIZE - 1)
        y = np.clip(y, 0, GRID_SIZE - 1)
        x0 = int(np.floor(x)); x1 = min(x0 + 1, GRID_SIZE - 1)
        y0 = int(np.floor(y)); y1 = min(y0 + 1, GRID_SIZE - 1)
        dx = x - x0; dy = y - y0
        v00 = SDF[x0, y0]; v10 = SDF[x1, y0]; v01 = SDF[x0, y1]; v11 = SDF[x1,
y1]
        v0 = v00 * (1 - dx) + v10 * dx
        v1 = v01 * (1 - dx) + v11 * dx
        v = v0 * (1 - dy) + v1 * dy
        return float(v)

    def grad_sdf(p, eps=1e-2):
        """чисельний градієнт SDF"""
        x, y = p
        sx = sdf_at_point((x + eps, y)) - sdf_at_point((x - eps, y))
        sy = sdf_at_point((x, y + eps)) - sdf_at_point((x, y - eps))
        g = np.array([sx, sy]) / (2 * eps)
        n = norm(g)
        return g / (n + 1e-9)

    def project_halfspace(v0, a, b):
        """
        Проекція вектора v0 на півпростір { v : a^T v >= b } за мінімумом ||v -
v0||^2.
        Якщо умова вже виконується, повертаємо v0.
```

```
    Інакше найменша зміна: v = v0 + ((b - a^T v0)/(||a||^2)) * a
    """
    a = np.asarray(a, dtype=float)
    denom = a @ a
    if denom < 1e-9:
        return v0
    gap = a @ v0 - b
    if gap >= 0:
        return v0
    # зміщуємо по напрямку a
    return v0 + ((b - (a @ v0)) / denom) * a

# Ініціалізація станів роботів
positions = starts.copy().astype(float)        # shape (N,2)
velocities = np.zeros_like(positions)          # поточні швидкості
reached = np.zeros(NUM_ROBOTS, dtype=bool)
arrival_time = np.full(NUM_ROBOTS, np.nan)

# Для логування
traj_logs = [ [positions[i].copy()] for i in range(NUM_ROBOTS) ]
speed_logs = [ [] for _ in range(NUM_ROBOTS) ]
dist_obst_logs = [ [] for _ in range(NUM_ROBOTS) ]
dist_human_logs = [ [] for _ in range(NUM_ROBOTS) ]
consensus_error_logs = [ [] for _ in range(NUM_ROBOTS) ]

# Вспомогательные: знайти сусідів по радіусу (геометрично)
def neighbors_of(i, positions, R_comm):
    ni = []
    for j in range(len(positions)):
        if j == i: continue
```

```
            if norm(positions[j] - positions[i]) <= R_comm:
                ni.append(j)
        return ni

    # Симуляція крок за кроком
    time_axis = np.arange(0, T_TOTAL, DT)
    for step, t in enumerate(time_axis):
        # оновити модель людини (проста лінійна модель)
        human_pos = human_pos + human_vel * DT
        # clamp в межі карти
        human_pos = np.clip(human_pos, 0.0, GRID_SIZE - 1.0)
        human_speed = norm(human_vel)
        R_s = R_safe(human_speed)

        # для кожного робота: формуємо бажану швидкість (до цілі) + узгодження з сусідами
        desired_vs = np.zeros_like(velocities)
        for i in range(NUM_ROBOTS):
            if reached[i]:
                desired_vs[i] = np.array([0.0, 0.0])
                continue
            p = positions[i]
            goal = goals[i]
            to_goal = goal - p
            dist_to_goal = norm(to_goal)
            # якщо близько до цілі - зупинка та лог часу
            if dist_to_goal < 0.8:
                reached[i] = True
                arrival_time[i] = t
                desired_vs[i] = np.zeros(2)
```

```
            continue
        # бажана швидкість до цілі (пропорційна, але обмежена)
        v_goal = unit(to_goal) * min(MAX_SPEED, dist_to_goal / 0.5)  # пропорція для плавності
        # консенсус: усереднена швидкість сусідів (як ескіз траєкторії)
        neigh = neighbors_of(i, positions, COMM_RADIUS)
        if len(neigh) > 0:
            v_neighbors = np.mean(velocities[neigh], axis=0)
        else:
            v_neighbors = np.zeros(2)
        # комбінуємо (використовуємо LAMBDA_XI як вага для узгодження)
        desired_vs[i] = (1 - LAMBDA_XI) * v_goal + LAMBDA_XI * v_neighbors

    # для кожного робота формуємо набір лінійних обмежень A v >= b і проекцією знаходимо безпечну швидкість
    next_vs = np.zeros_like(velocities)
    for i in range(NUM_ROBOTS):
        v0 = desired_vs[i].copy()
        p = positions[i]

        # Збираємо обмеження (a, b)
        A_list = []
        b_list = []

        # 1) Межі по швидкості: ||v|| <= MAX_SPEED --> це не лінійне, реалізуємо як проєкцію круга:
        # Але ми применимо це після лінійних проекцій: просто обмежимо довжину вектора пізніше.
```

```
            # 2) CBF між роботами: h_ij = ||p - p_j||^2 - (r_i + r_j + d_m)^2 >= 0
            for j in range(NUM_ROBOTS):
                if j == i: continue
                pj = positions[j]
                rij = ROBOT_RADIUS + ROBOT_RADIUS + INTER_ROBOT_BUFFER
                dp = p - pj
                h_ij = dp @ dp - (rij) ** 2
                # ∇_p h = 2 (p - p_j)
                grad_h = 2.0 * dp
                # constraint: grad_h^T v >= -gamma * h
                a = grad_h
                b = -CBF_GAMMA * h_ij
                A_list.append(a)
                b_list.append(b)

            # 3) CBF для людини: h_iH = (p - pH)^T Sigma^{-1} (p - pH) - R_s^2 >= 0
            dpiH = p - human_pos
            h_iH = dpiH.T @ np.linalg.inv(human_Sigma) @ dpiH - R_s ** 2
            grad_hH = 2.0 * (np.linalg.inv(human_Sigma) @ dpiH)   # gradient wrt p
            aH = grad_hH
            bH = -CBF_GAMMA * h_iH
            A_list.append(aH)
            b_list.append(bH)

            # 4) Перешкоди (SDF)
            phi = sdf_at_point(p)   # відстань до найближчої перешкоди (позитивна поза)
            grad_phi = grad_sdf(p)
```

```
    # h_iO = phi - d_min >= 0  -> grad_phi^T v >= -gamma * h_iO
    d_min = ROBOT_RADIUS + 0.2
    h_iO = phi - d_min
    aO = grad_phi
    bO = -CBF_GAMMA * h_iO
    A_list.append(aO)
    b_list.append(bO)
    # Тепер послідовні проекції (POCS-like)
    v_safe = v0.copy()
    for _ in range(PROJECT_ITERS):
        for (a, b) in zip(A_list, b_list):
            v_safe = project_halfspace(v_safe, a, b)

    # обмеження по максимальній швидкості (норма)
    speed = norm(v_safe)
    if speed > MAX_SPEED:
        v_safe = v_safe * (MAX_SPEED / speed)

    next_vs[i] = v_safe

# оновлення стану (положення)
positions = positions + next_vs * DT
velocities = next_vs.copy()

# логування
for i in range(NUM_ROBOTS):
    traj_logs[i].append(positions[i].copy())
    speed_logs[i].append(norm(velocities[i]))
    # відстань до найближчої перешкоди (в точці p)
    dist_obst_logs[i].append(sdf_at_point(positions[i]))
```

```
            # відстань до людини
            dist_human_logs[i].append(norm(positions[i] - human_pos))
            # консенсус помилка: різниця бажаного vs фактичного з урахуванням сусідів
            neigh = neighbors_of(i, positions, COMM_RADIUS)
            if len(neigh) > 0:
                avg_v = np.mean(velocities[neigh], axis=0)
                consensus_error_logs[i].append(norm(velocities[i] - avg_v))
            else:
                consensus_error_logs[i].append(0.0)

        # перевірка завершення (всі досягли цілей)
        if np.all(reached):
            print(f"Усі роботи досягли цілей на часі t = {t:.2f} s")
            break

    # Після симуляції: підготовка графіків
    # Перетворимо логи в зручні масиви
    traj_arrays = [np.array(tr) for tr in traj_logs]
    spd_arrays = [np.array(s) for s in speed_logs]
    dist_obs_arrays = [np.array(d) for d in dist_obst_logs]
    dist_h_arrays = [np.array(d) for d in dist_human_logs]
    cons_err_arrays = [np.array(c) for c in consensus_error_logs]

    # 1) Траєкторії на фоні карти перешкод та людини (ризикове поле)
    plt.figure(figsize=(8, 8))
    # відобразимо SDF як контури
    X, Y = np.meshgrid(np.arange(GRID_SIZE), np.arange(GRID_SIZE), indexing='ij')
```

```
    plt.contourf(X, Y, np.clip(SDF, -5, 10), levels=40, cmap='coolwarm', alpha=0.6)
    # перешкоди
    for (ox, oy, r) in OBSTACLES:
        c = plt.Circle((oy, ox), r, color='k', alpha=0.6)
        plt.gca().add_patch(c)
    # людина: локація та еліпс (mah)
    from matplotlib.patches import Ellipse
    plt.scatter(human_pos[1], human_pos[0], c='yellow', edgecolors='k', s=120, label='Human')
    # еліптичний ризик (2 sigma)
    eigvals, eigvecs = np.linalg.eig(human_Sigma)
    angle = np.degrees(np.arctan2(eigvecs[0,1], eigvecs[0,0]))
    ell = Ellipse((human_pos[1], human_pos[0]), width=2*np.sqrt(eigvals[1])*2, height=2*np.sqrt(eigvals[0])*2,
                  angle=angle, edgecolor='yellow', facecolor='none', lw=1.5, ls='--')
    plt.gca().add_patch(ell)

    colors = ['r', 'g', 'b', 'm', 'c']
    for i, traj in enumerate(traj_arrays):
        plt.plot(traj[:,1], traj[:,0], color=colors[i], label=f'Robot {i+1}')
        plt.scatter(starts[i,1], starts[i,0], marker='o', color=colors[i])
        plt.scatter(goals[i,1], goals[i,0], marker='x', color=colors[i])
    plt.title('Траєкторії роботів на фоні SDF (контур) та перешкод')
    plt.xlim(-1, GRID_SIZE)
    plt.ylim(-1, GRID_SIZE)
    plt.gca().invert_yaxis()
    plt.legend()
    plt.xlabel('Y'); plt.ylabel('X')
    plt.grid(True)
```

```
# 2) Швидкість кожного робота в часі
plt.figure(figsize=(10, 6))
for i in range(NUM_ROBOTS):
    plt.plot(spd_arrays[i], label=f'Robot {i+1}')
plt.title('Швидкості роботів в часі')
plt.xlabel('Кроки (dt = {:.2f}s)'.format(DT)); plt.ylabel('Швидкість (од/с)')
plt.legend(); plt.grid(True)

# 3) Відстань до перешкод у часі (SDF)
plt.figure(figsize=(10, 6))
for i in range(NUM_ROBOTS):
    plt.plot(dist_obs_arrays[i], label=f'Robot {i+1}')
plt.title('Відстань до найближчої перешкоди (SDF) в часі')
plt.xlabel('Кроки'); plt.ylabel('SDF (одиниці карти)')
plt.legend(); plt.grid(True)

# 4) Відстань до людини у часі
plt.figure(figsize=(10, 6))
for i in range(NUM_ROBOTS):
    plt.plot(dist_h_arrays[i], label=f'Robot {i+1}')
plt.title('Відстань від робота до людини (в часі)')
plt.xlabel('Кроки'); plt.ylabel('Відстань (одиниці карти)')
plt.legend(); plt.grid(True)

# 5) Помилка консенсусу
plt.figure(figsize=(10, 6))
for i in range(NUM_ROBOTS):
    plt.plot(cons_err_arrays[i], label=f'Robot {i+1}')
plt.title('Помилка консенсусу (відхилення від середньої швидкості сусідів)')
```

```
plt.xlabel('Кроки'); plt.ylabel('Помилка (норми вектору)'); plt.legend(); plt.grid(True)

# 6) Бар-діаграма часу досягнення цілі
plt.figure(figsize=(8, 5))
arr_times = np.where(np.isnan(arrival_time), T_TOTAL, arrival_time)
plt.bar(np.arange(1, NUM_ROBOTS+1), arr_times, color=colors[:NUM_ROBOTS])
plt.title('Час досягнення цілі (якщо не досягнуто, показано T_TOTAL)')
plt.xlabel('Робот'); plt.ylabel('Час (s)'); plt.grid(axis='y')

plt.show()
# Вивід чисел по роботах
print("\nРезюме по роботах:")
for i in range(NUM_ROBOTS):
    total_path_len = traj_arrays[i].shape[0]
    avg_speed = np.mean(spd_arrays[i]) if spd_arrays[i].size>0 else 0.0
    t_arr = arrival_time[i] if not np.isnan(arrival_time[i]) else None
    print(f"Robot {i+1}: Path steps = {total_path_len}, Avg speed = {avg_speed:.3f}, Arrival time = {t_arr}")
```

ДОДАТОК Б

Апробація результатів кваліфікаційної роботи

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

IX Міжнародна Конференція
ВИРОБНИЦТВО
&
МЕХАТРОННІ СИСТЕМИ 2025

IX International Conference
MANUFACTURING
&
MECHATRONIC SYSTEMS 2025

M&MS
2025
IX International Conference
25-26 October
Kharkiv

УДК: 005:004.896:62-65:338.3

Виробництво & Мехатронні Системи 2025: матеріали IX-ої Міжнародної конференції, Харків, 25-26 жовтня 2025 р.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2025. – 115 с.

У збірник включені тези доповідей, які присвячені сучасним тенденціям розвитку технологій та засобів виробництва та мехатронних систем, передовому досвіду та впровадженню їх в галузях систем промислової автоматизації та керування виробництвом; системній інженерії; CAD/CAM/CAE системах; мехатроніці (електро-механічних системах, електронних інструментах систем керування, механічних CAD системах); робототехніці та засобахм інтелектуалізації; MEMS (сучасних матеріалів та технологіях виготовлення MEMS) та компонентах і технологіях автоматизації видобутку, переробки та транспортування нафти та газу.

Редакційна колегія: І.Ш. Невлюдов, В.В. Євсєєв.

Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2025. - 115 p.

The collection includes the thesises of reports on modern trends in the development of technologies and means of production and mechatronic systems, top experience and implementation of them in fields of: industrial automation and production management systems; systems engineering; CAD/CAM/CAE systems; mechatronics (electrical and mechanical systems, electronic control tools, mechanical CAD systems); robotics and intellectual toolls; MEMS (modern materials and manufacturing technologies MEMS) and components and technologies for the automation of oil, gas and oil extraction, processing and transportation.

Editorial board: Igor.Sh. Nevludov, Vladyslav.V. Yevsieiev

Development of a model for decentralized control of a group of collaborative robot manipulators

Svitlana Maksymova [1], Pavlo Shakhov[2]

1. CITAR Department, Kharkiv National University of Radio Electronics, UKRAINE, Kharkiv, Nauki Ave. 14., e-mail: svitlana.milyutina@nure.ua @nure.ua
2. CITAR Department, Kharkiv National University of Radio Electronics, UKRAINE, Kharkiv, Nauki Ave. 14., e-mail: pavlo.shakhov@nure.ua

***Abstract*:** The paper considers an approach to developing a model for decentralized control of a group of collaborative robot manipulators, focused on increasing autonomy, adaptability, and safety in a dynamic production environment. The proposed solution allows minimizing dependence on centralized computing resources, reducing the risks of system failures, and ensuring effective interaction of robots in the joint performance of complex manipulation tasks. The results of the study demonstrate the prospects of decentralized models for implementing the concepts of Industry 5.0.

***Keywords*:** decentralized control, collaborative robot manipulators, multi-agent systems, adaptability, Industry 5.0.

I. Introduction

The modern development of robotics is increasingly focused on the creation of collaborative systems capable of effectively interacting with humans and with each other in a shared working environment. The increasing complexity of production processes, the requirements for flexibility and increased productivity necessitate the development of new control models that ensure adaptability and reliability. Robot manipulators that perform complex manipulation tasks in conditions of limited space and high variability of actions attract particular attention. Traditional centralized control systems have significant limitations related to scalability, vulnerability to failures and high requirements for computing resources. In this context, decentralized control is a promising approach that allows reducing the risks of critical failures, increasing the level of agent autonomy and ensuring effective interaction in heterogeneous robot teams. Research in this area is becoming particularly relevant within the framework of the Industry 5.0 concept, which envisages harmonious cooperation between humans and robotic systems. At the same time, the lack of sufficiently developed models capable of taking into account the dynamics of the environment and the variability of teams determines the scientific and practical significance of the topic. Thus, the creation of a model of decentralized control of a group of collaborative robot manipulators is an important step in the development of effective, safe and adaptive robotic systems.

II. Development of a Mathematical Model of Decentralized Control of a Group of Collaborative Robot Manipulators

The decentralized control model of a group of robots was chosen to ensure scalability and fault tolerance of the system, since the absence of a single control center minimizes the risk of failures in the event of a single node failure. This approach allows each robot to make decisions based on local information and data exchange with neighboring agents, which reduces delays and improves adaptability in a dynamic environment. To formalize the interaction between agents, a communication graph is used, which reflects the structure of connections between nodes and guarantees consistency of actions through consensus algorithms. The communication graph makes it possible to determine which robots exchange data and take into account restrictions on the communication range. In the framework of the study, the communication graph is described as a directed or undirected graph that provides symmetry or asymmetry of data exchange.

$$\begin{gathered}\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t)),\ |\mathcal{V}| = N \\ \mathcal{N}_i(t) = \{j : (i,j) \in \mathcal{E}(t)\}\end{gathered} \tag{1}$$

Where: $\mathcal{V}$ - a set of vertices, where each vertex corresponds to a separate robot or agent of the system;

$\mathcal{E}(t)$ - set of oriented or unoriented edges at a point in time t, which determine the presence of active communication channels between robots;

$|\mathcal{V}| = N$ - the total number of robots (nodes) in the system, which determines the size of the network;

$\mathcal{N}_i(t)$ - set of neighbors of a node i at a point in time t, that is, those agents with which robot i can directly exchange data.

In general, $\mathcal{N}_i(t) = \{j : (i,j) \in \mathcal{E}(t)\}$ indicates that a connection exists if the edge from i to j belongs to the set $\mathcal{E}(t)$. Thus, these parameters allow us to dynamically describe the topology of connections in a group of robots and ensure the correct operation of decentralized control algorithms.

The safety conditions of Control Barrier Functions (CBF) in the end effector (EE) space are represented by three cases:

- inter-robot distances (prevents robot collisions by ensuring a minimum safe distance)

$$h_{ij}\left(p_i, p_j\right) = \left\|p_i - p_j\right\|^2 - (r_i + r_j + d_m)^2 \geq 0 \tag{2}$$

- from a person (determines an elliptical safety zone for a person, taking into account their predicted movement and uncertainty)

$$h_{iH}(p_i, p_H) = (p_i - p_H)^T \sum_H^{-1}(t)(p_i - p_H) - R_{safe}^2(t) \geq 0 \quad (3)$$

- from obstacles through SDF (the condition guarantees the avoidance of collisions with objects or walls, taking into account the changing environment):

$$h_{iO}(p_i) = \phi(p_i, t) - d_{min} \geq 0 \quad (4)$$

Where: p_i, p_j - positions of robots *i* and *j* in space;

$\|p_i - p_j\|^2$ - square of the distance between two robots;

r_i, r_j - radii of safe zones for each robot (taking into account their dimensions);

d_m - additional minimum safety buffer between robots;

p_H - position of a person in the work area;

$R_{safe}^2(t)$ - dynamic radius of the safe zone around a person;

$\phi(p_i, t)$ - distance from the robot to the nearest static or dynamic obstacle at a given time*t*;

d_{min} - minimum permissible distance to an obstacle.

The human model is represented as a dynamic obstacle, and consists of the following four basic parameters:

- human state

$$\mathcal{Y}_H = [p_H^T, \dot{p}_H^T]^T \in \mathbb{R}^6 \quad (5)$$

Where: $\mathcal{Y}_H$ - the vector of the human state in the robot's workspace, including both position and velocity;

$p_H \in \mathbb{R}^3$ - vector of spatial coordinates (position) of a person in three-dimensional space, usually in the coordinate system of the work area or the robot's base system;

$\dot{p}_H \in \mathbb{R}^3$ - vector of linear velocities of human movement (change in position over time);

p_H^T - transposed position vector, used to form the complete state into a single column;

$\dot{p}_H^T$ - transposed velocity vector;

$\mathbb{R}^6$ - state space dimension: 3 coordinates for position and 3 for velocity, total 6 components.

- trajectory forecast for the forecasting horizon in time (T_H):

$$T_H: \hat{p}_H(t + k) \quad (6)$$

Where: T_H - the time horizon, which determines the number of discrete steps in the future for which the person's position is predicted;

$\hat{p}_H(t + k)$ - predicted position of a person at a given moment in time $t + k$, де $k \in [1, T_H]$;

t - current point in time from which the forecast begins;

k - forecast step index within the horizon T_H, used for iterative trajectory prediction;

$\hat{p}_H$ - the symbol "^" indicates that this is not a real position, but an estimate or prediction obtained from a motion model (for example, a CV model - constant velocity or using a Kalman filter).

- a person's safety (comfort) zone:

$$R_{safe}(t) = R_0 + \alpha\|\dot{p}_H(t)\| \quad (7)$$

Where: $R_{safe}(t)$ - the radius of the human safety (comfort) zone at time *t*, which determines the minimum distance at which the robot can approach the human without disrupting comfortable interaction;

R_0 - basic (static) radius of the safety zone, which takes into account a person's personal space, even when they are stationary;

α - a proportionality coefficient that determines how much a person's speed affects the size of the safety zone; larger values of α increase the zone at high speeds;

$\|\dot{p}_H(t)\|$ - the norm of the person's velocity vector at time *t*, which reflects the intensity of his movement; the faster the person moves, the larger the safety zone.

Model 2.23 allows adaptively changing the size of the comfort zone depending on the person's behavior.

- risk field, this is a Gaussian risk field around the predicted position of the person, where the risk decreases with distance according to the Mahalanobis distance:

$$\mathcal{R}(x, y) = exp\left(-\frac{1}{2}(x - \hat{p}_H(t))^T \sum_T^{-1}(t)(x - \hat{p}_H(t))\right) \quad (8)$$

Where: $\mathcal{R}(x, y)$ - dimensionless risk intensity at a planar point $x = [x, y]^T$, i.e. the closer to the person, the higher the value (maximum 1 in the center);

$x = [x, y]^T$ - current 2D position in the work plane in which we assess the risk; used in planning EE/link trajectories;

$\hat{p}_H(t) \in \mathbb{R}^2$ - predicted position of the person at time *t*, obtained from the prediction/filter model;

$(x - \hat{p}_H(t))^T \sum_T^{-1}(t)(x - \hat{p}_H(t))$ - squared Mahalanobis distance, which measures "how many sigmas" a point *x* is from the center, taking into account directional uncertainty;

exp (·) – a set specifying a smooth risk decay; the absence of a normalizing factor means that this is a relative risk/cost field, suitable as an additional part of the objective function in QP/MPC or as a weight layer in CBF constraints.

The local Quadratic Programming/Model Predictive Control (QP/MPC) control problem at speeds is such that at each step for robot i, QP is solved:

$$\min_{\dot{q}_i} \|J_{p,j}\dot{q}_i - v_i^{task}\|_{Q_v}^2 + \|\dot{q}_i\|_{R_\tau}^2 + \lambda_\mu(\mu_* - \mu_i(q_i))_+^2 + \lambda_\xi\|J_{p,i}\dot{q}_i - \hat{v}_{N_i}\|^2 \quad (9)$$

Where: $\dot{q}_i$ - vector of joint velocities of robot i, the optimization variable to be found;

$J_{p,j}$ - positional part of the Jacobian matrix for robot i, reflecting the relationship between the joint velocity and the linear velocity of the end-effector;

v_i^{task} - target speed of the working body (end effector) to perform a specific task (e.g., following a trajectory);

$\left\|J_{p,j}\dot{q}_i - v_i^{task}\right\|_{Q_v}^2$ - the first term of the functional that minimizes the deviation from the desired end-effector speed with a weight matrix Q_v, what determines the importance of this task;

$\left\|\dot{q}_i\right\|_{R_\tau}^2$ - the second term that minimizes the energy or steering torque (due to the speeds), with a weight matrix R_τ, which provides regularization for smoothing motion;

$\lambda_\mu(\mu_* - \mu_i(q_i))_+^2$ - fine for violation of manipulability restrictions;

$\mu_i(q_i)$ - robot manipulability indicator i;

μ_* - minimum permissible value of manipulability;

$(\cdot)_+$ - means the positive part (the penalty is activated if $\mu_i < \mu_*$);

λ_μ - weighting factor for this penalty;

$\lambda_\xi\left\|J_{p,i}\dot{q}_i - \hat{v}_{\mathcal{N}_i}\right\|^2$ - term for matching the speed with the average speed of neighbors in the communication graph;

$\hat{v}_{\mathcal{N}_i}$ - predicted or average speed of neighboring robots N_i;

λ_ξ - the weight of this agreement, which determines the importance of coordination.

III. RESULTS OF NUMERICAL MODELING AND ANALYSIS OF THE OBTAINED RESULTS

To conduct a numerical simulation of the synchronization of physical and virtual collaborative robots, a time interval from 0 to 10 seconds was used, which was discretized into 500 points to ensure smooth graphs and accuracy of calculations. The trajectory of the physical robot was modeled as a sinusoidal function with an amplitude of 1 and the addition of random noise with an intensity of 0,05, which simulates real fluctuations in sensor measurements. The virtual robot reproduced the same trajectory, but with a time shift of 0,2 seconds, which allows us to assess the system's ability to compensate for the delay between the physical and digital models. To analyze the synchronization error, we used the difference in the states of the physical and virtual robots at each time point. The neural network received normalized time data in the range from 0 to 1 as input, which allowed us to avoid the influence of scale in the calculations. The network architecture consisted of one hidden layer with 10 neurons, random weights were generated based on a normal distribution, and the ReLU activation function was used to model nonlinearity. The output of the network approximated the sine target function, which made it possible to evaluate the effectiveness of artificial intelligence in reproducing the dynamics of the robot's movement. Thus, the selected numerical parameters ensured the reproduction of both the physical characteristics of the robot and the process of its virtual synchronization using a digital twin. The obtained results of the numerical simulation are presented in Figures 1-3.

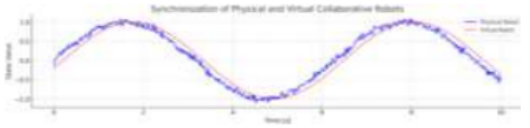

Figure 1. – Synchronization of Physical and Virtual Collaborative Robots Graph

Synchronization of Physical and Virtual Collaborative Robots (Fig. 1), shows the trajectories of physical and virtual collaborative robots, where their approximation and deviation during the synchronization process are visible.

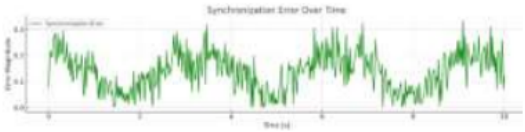

Figure 2. – Synchronization Error Over Time Graph

Synchronization Error Over Time (Fig. 2) reflects the change in synchronization error over time, which makes it possible to assess the stability and efficiency of the robot motion coordination process.

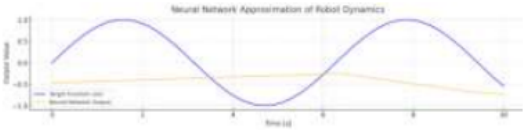

Figure 3. – Neural Network Approximation of Robot Dynamics Graph

Neural Network Approximation of Robot Dynamics (Fig. 3) demonstrates the operation of a simple neural network that approximates the dynamics of robot motion and allows modeling the behavior of the system in real time.

III. CONCLUSION

The study showed that the use of digital twins in combination with artificial intelligence creates an effective tool for synchronizing physical and virtual collaborative robots in dynamic conditions. Numerical modeling confirmed the possibility of accurately reproducing the movements of a physical robot in a virtual environment, taking into account time shifts and sensor errors, which significantly increases the accuracy and reliability of control. The use of neural networks provided the possibility of adaptive learning of the system and approximation of complex dynamic characteristics to real scenarios, which is key for work in Industry 5.0 conditions. The results demonstrated the ability to reduce the average synchronization error and improve the stability of the interaction between the real and digital environments. The proposed approaches can be applied to build integrated control systems capable of quickly responding to changes in external factors and interaction with a person. The use of

mathematical models made it possible to analyze the main parameters of the system and determine the optimal operating modes of digital twins. The results obtained create the basis for further research aimed at developing hybrid control architectures using artificial intelligence methods and multi-agent models.

REFERENCES

[1] Khudov, H., Khudov, R., Khizhnyak, I., Makoveichuk, O., & Khudov, V. (2025). Image Segmentation Methods for Kamikaze FPV Drones Targeting to Aid Critical Energy National Infrastructure Assets Protection. In Systems, Decision and Control in Energy VII: Volume I: Energy Informatics and Transport (pp. 139-151). Cham: Springer Nature Switzerland.

[2] Conejero, M. N., Montes, H., Bengochea-Guevara, J. M., Garrido-Rey, L., Andújar, D., & Ribeiro, A. (2025). A collaborative robotic fleet for yield mapping and manual fruit harvesting assistance. Computers and Electronics in Agriculture, 235, 110351.

[3] Attar, H., Abu-Jassar, A. T., Yevsieiev, V., Lyashenko, V., Nevliudov, I., & Luhach, A. K. (2022). Zoomorphic mobile robot development for vertical movement based on the geometrical family caterpillar. Computational intelligence and neuroscience, 2022(1), 3046116.

[4] Nevliudov, I., Yevsieiev, V., Baker, J. H., Ahmad, M. A., & Lyashenko, V. (2020). Development of a cyber design modeling declarative Language for cyber physical production systems. J. Math. Comput. Sci., 11(1), 520-542.

[5] Nevliudov, I., & et al.. (2020). Method of Algorithms for CyberPhysical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research, 8(10), 7465-7473.

[6] Lyashenko, V., Abu-Jassar, A. T., Yevsieiev, V., & Maksymova, S. (2023). Automated Monitoring and Visualization System in Production. International Research Journal of Multidisciplinary Technovation, 5(6), 9-18.

[7] Mustafa, S. K., Yevsieiev, V., Nevliudov, I., & Lyashenko, V. (2022). HMI Development Automation with GUI Elements for Object-Oriented Programming Languages Implementation. SSRG International Journal of Engineering Trends and Technology, 70(1), 139-145.

[8] Nevliudov, I., Yevsieiev, V., Lyashenko, V., & Ahmad, M. A. (2021). GUI Elements and Windows Form Formalization Parameters and Events Method to Automate the Process of Additive Cyber-Design CPPS Development. Advances in Dynamical Systems and Applications, 16(2), 441-455.

[9] Kragic, D., Gustafson, J., Karaoguz, H., Jensfelt, P., & Krug, R. (2018, July). Interactive, Collaborative Robots: Challenges and Opportunities. In IJCAI (pp. 18-25).

[10] Nevliudov, I., Yevsieiev, V., Maksymova, S., Gopejenko, V., & Kosenko, V. (2025). Development of mathematical support for adaptive control for the intelligent gripper of the collaborative robot manipulator. Advanced Information Systems, 9(3), 57-65.

[11] Maksymova, S., Yevsieiev, V., Chala, O., & Ababneh, J. (2025). DECISION-MAKING MODEL FOR CONTROLLING A COLLABORATIVE ROBOT-MANIPULATOR BASED ON THE SENSOR FUSION METHOD AND THE RULES OF RULE-BASED SYSTEMS. Multidisciplinary Journal of Science and Technology, 5(6), 526-538.

[12] Невлюдов, І. Ш., Євсєєв, В. В., & Гурін, Д. В. (2025). Model development of dynamic representation a model description parameters for the environment of a collaborative robot manipulator within the industry 5.0 framework. Системи управління, навігації та зв'язку. Збірник наукових праць, 1(79), 42-48.

[13] Yevsieiev, V., Abu-Jassar, A., Maksymova, S., & Demska, N. (2025). Development of a model for recognizing various objects and tools in a collaborative robot workspace. ACUMEN: International journal of multidisciplinary research, 2(1), 224-239.

[14] Yevsieiev V. Mobile Robots and Autonomous Vehicles in the Mobility as a Service (MAAS) Concept / V. Yevsieiev // Sustainable smart cities and communities: business and innovation solutions 2025 : Thesises of Reports of I st I International Conference, April 21, 2025. - Kharkiv, 2025. - P.7-8.

[15] Yevsieiev V. Using Multi-Agent Systems in the Management of Collaborative Robots / V. Yevsieiev // Computer-integrated technologies, automation and robotics 2025 : Thesises of Reports of II st All-Ukrainian Conference, May 16-17, 2025. - Kharkiv, 2025. - P. 13-17

[16] Yevsieiev, V., Maksymova, S., Gurin, D., & Alkhalaileh, A. (2024). HR data visualization of the distance to the object in the collaborative robot workspace based on hc-sr04 sensor. ACUMEN: International journal of multidisciplinary research, 1(4), 388-401.

[17] Yevsieiev, V., Maksymova, S., Abu-Jassar, A., & Ababneh, J. (2025). MATHEMATICAL MODEL OF LOCAL DECISION-MAKING FOR COLLABORATIVE ROBOTS USING EDGE COMPUTING. Multidisciplinary Journal of Science and Technology, 5(6), 34-46.

[18] Yevsieiev, V. Comparative Analysis of the Characteristics of Mobile Robots and Collaboration Robots Within INDUSTRY 5.0. / V. Yevsieiev, D. Gurin // Sectoral research XXI : characteristics and features : collection of scientific papers "SCIENTIA" with proceedings of the VI International Scientific and Theoretical Conference, September 8, 2023. - Chicago : European Scientific Platform, 2023. - P. 92-94.

[19] Yevsieiev, V., Ababneh, J., Maksymova, S., & Abu-Jassar, A. (2025). DEVELOPMENT OF A MATHEMATICAL MODEL FOR SIMULATING A DECENTRALIZED CONTROL SYSTEM FOR COLLABORATIVE ROBOT NETWORKS. Multidisciplinary Journal of Science and Technology, 5(5), 1187-1202.

[20] НЕВЛЮДОВ, І., ЄВСЄЄВ, В., & ГУРІН, Д. (2025). МАТЕМАТИЧНА МОДЕЛЬ БЛОЧНОГО ПРОЦЕСНОГО ПЛАНУВАННЯ В СИСТЕМАХ АЛОКАЦІЇ ЗАВДАНЬ МІЖ ЛЮДЬМИ ТА КАЛАБОРАТІВНИМИ РОБОТАМИ В РАМКАХ ІНДУСТРІЙ 5.0. Вісник Херсонського національного технічного університету, 1(1 (92)), 157-163.

ДОДАТОК В

Демонстраційний матеріал

№ документа	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 432361.015 ПЗ	Пояснювальна	А4, 93 с.
		записка	
		Додаткові матеріали	
2		Демонстраційний	А4, 14 с.
		матеріал у вигляді	
		презентації	

ГЮИК. 432361.015 ПЗ

Змін	Арк	№ документу	Підпис	Дата
Розробив		Шахов П.В.		
Перевірила		Максимова С.С.		
Норм. контроль		Стародубцев М.Г.		
Затвердив		Невлюдов І.Ш.		

Розробка програмного забезпечення децентралізованого керування групою колаборативних роботів	Літер	Аркуш	Аркуш
	Н	1	1
	Кафедра КІТАР ХНУРЕ		