ДОДАТОК А
Графічні матеріали

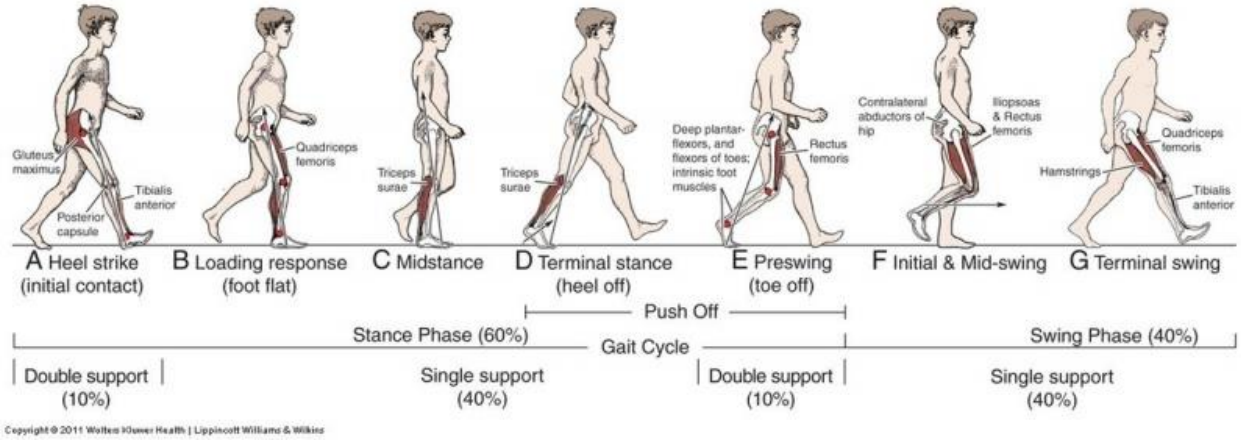


Рисунок А.1 – Фази ходи людини

Рисунок А.2 – Діаграма похибок та їх причина виникнення

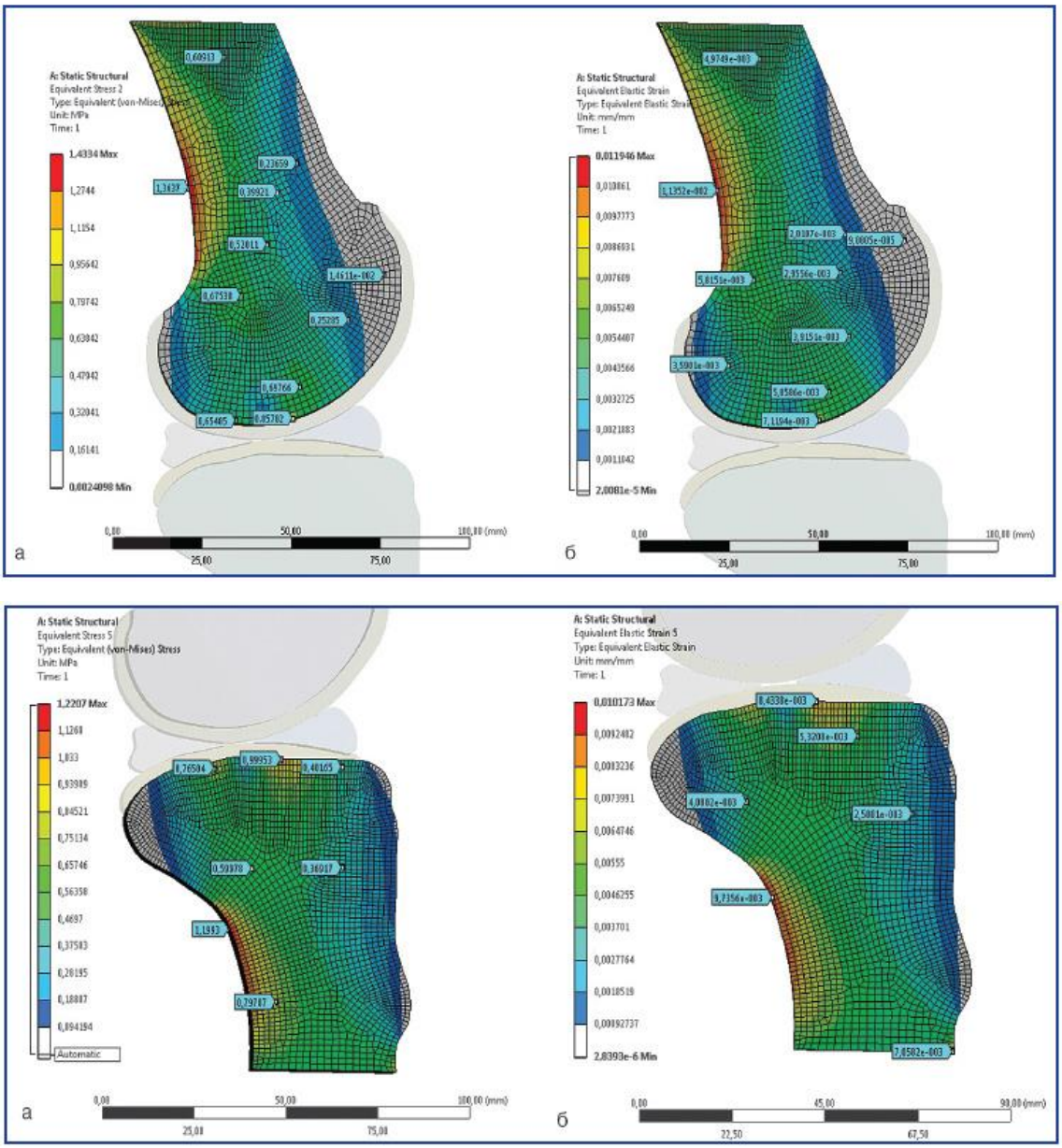


Рисунок А.3 – Біомеханічна модель

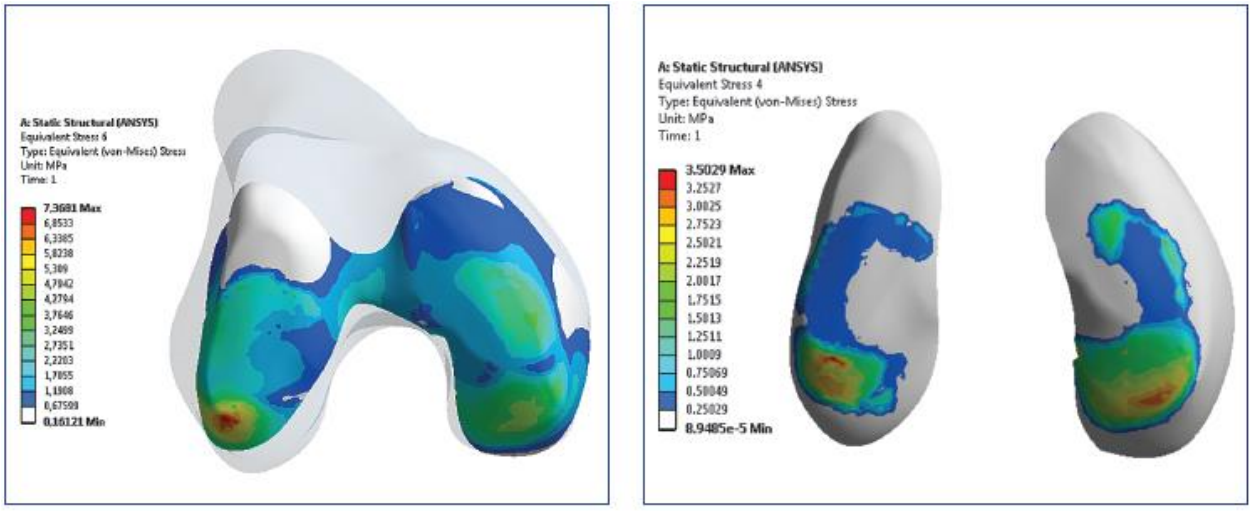


Рисунок А.4 – Математична модель

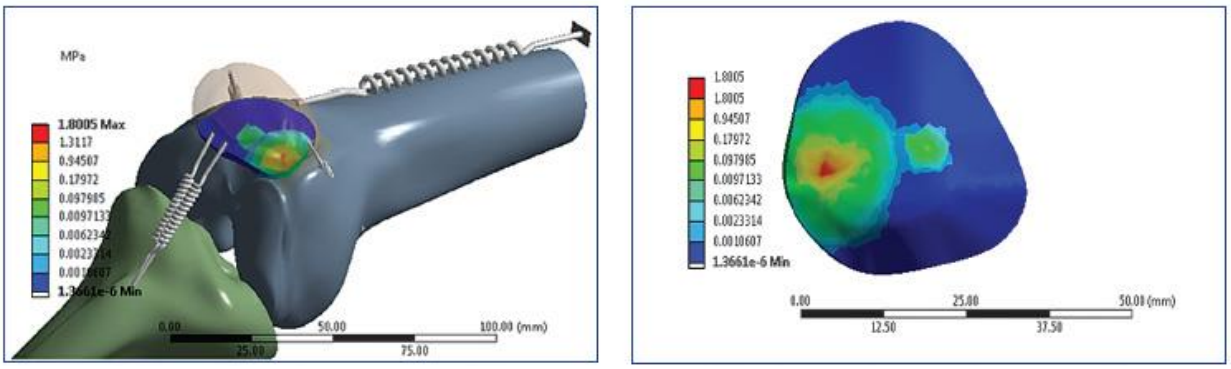


Рисунок А.5 – Імітаційне моделювання

Рисунок А.6 – Місця встановлення маркерів для запису рухів стопи

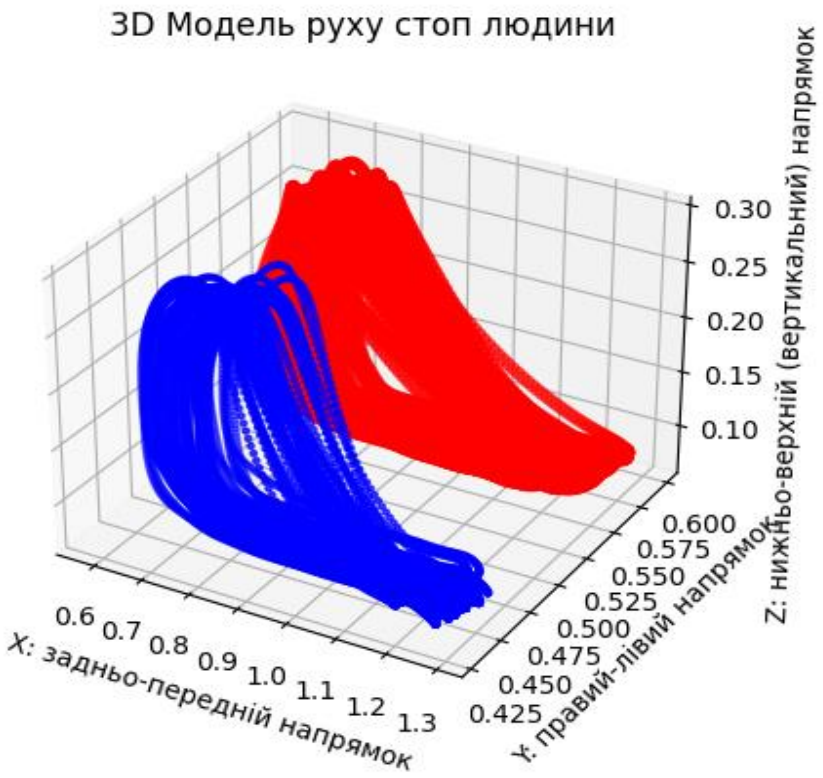


Рисунок А.7 – 3D модель руху стоп людини

group_symmetry_analizer.py | cluster_analysis_plot.png | group_analysis_plot.png | group_analysis.csv | all_asymmetry_speed_0.6.csv | data_normalization.py | constant

csv

	Marker	Axis	L_Mean	R_Mean	Symmetry Index (%)	File
1	FCC	y	0.5628520858325459	0.46873880492503506	18.24624116996531	GP10_0.6_marker.csv
2	FM2	y	0.5721648537504894	0.4469755349096561	24.567629785611494	GP10_0.6_marker.csv
3	FM5	y	0.6175704018851854	0.4032287341866586	41.99487639132198	GP10_0.6_marker.csv
4	FM1	z	0.06464575957856511	0.07248544527979807	-11.433846452862758	GP10_0.6_marker.csv
5	FCC	y	0.5649570012109084	0.45411855267461887	21.752744065675046	GP11_0.6_marker.csv
6	FM1	y	0.530054516553734	0.4785102559040718	10.221308944601	GP11_0.6_marker.csv
7	FM2	y	0.5696428174505495	0.4427803520118327	25.061154123148597	GP11_0.6_marker.csv
8	FM5	y	0.6154525189985777	0.40286231561723657	41.75333524656867	GP11_0.6_marker.csv
9	FM1	z	0.06563995018222203	0.07355388988603707	-11.37110622127263	GP11_0.6_marker.csv
10	FCC	y	0.5694842085368997	0.4581559516530474	21.66677815769182	GP12_0.6_marker.csv
11	FM1	y	0.5463353489880564	0.4593733962825945	17.293665410468144	GP12_0.6_marker.csv
12	FM2	y	0.586254116333466 2	0.4206116347370353	32.902595290448524	GP12_0.6_marker.csv
13	FM5	y	0.6262887579569838	0.3866343631119119	47.31936508511615	GP12_0.6_marker.csv
14	FCC	y	0.567958352328712	0.4013835418936284	34.368639471364034	GP13_0.6_marker.csv
15	FM1	y	0.5675705128495669	0.4319248169677477	27.14283735705143	GP13_0.6_marker.csv
16	FM2	y	0.6037669548173712	0.3930827855946918	42.269995302519696	GP13_0.6_marker.csv
17	FM5	y	0.653563199673656	0.34632364004870114	61.4548661747098	GP13_0.6_marker.csv

Рисунок А.8 – Результат отриманих даних файлу «all_asymmetry_speed_0.6»

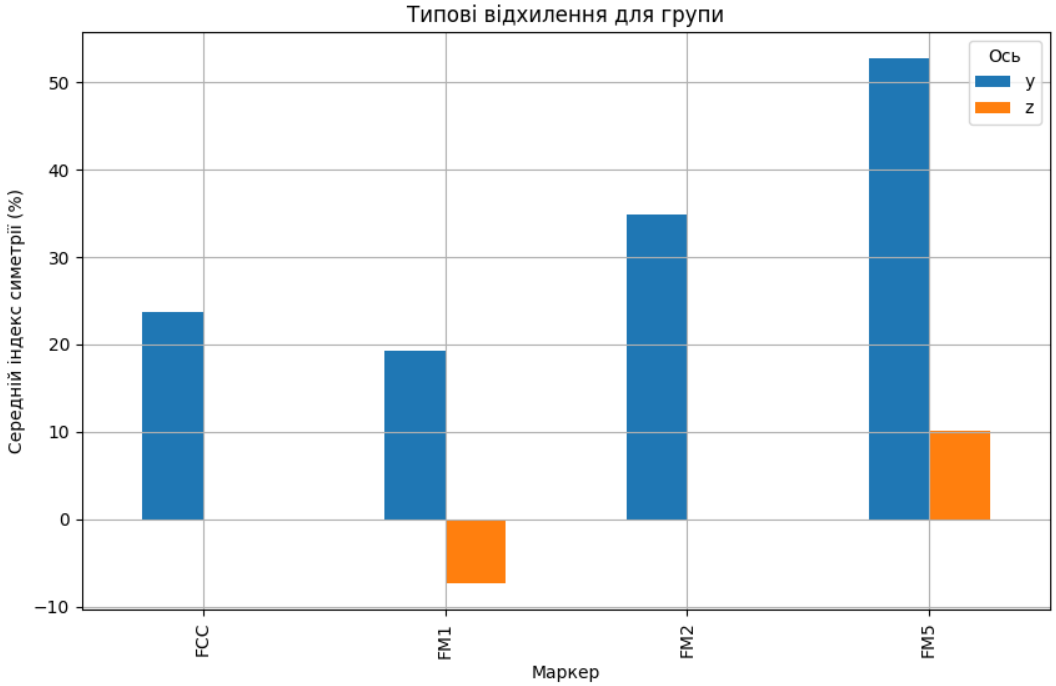


Рисунок А.9 – Типові відхилення групи

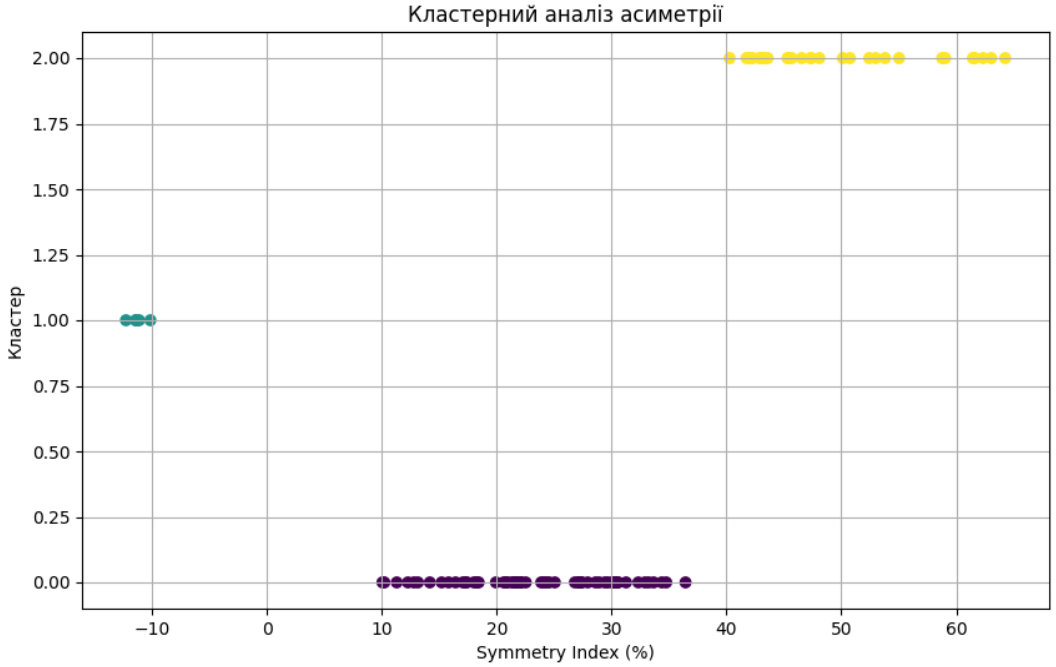


Рисунок А.10 – Графік кластерний аналіз асиметрії

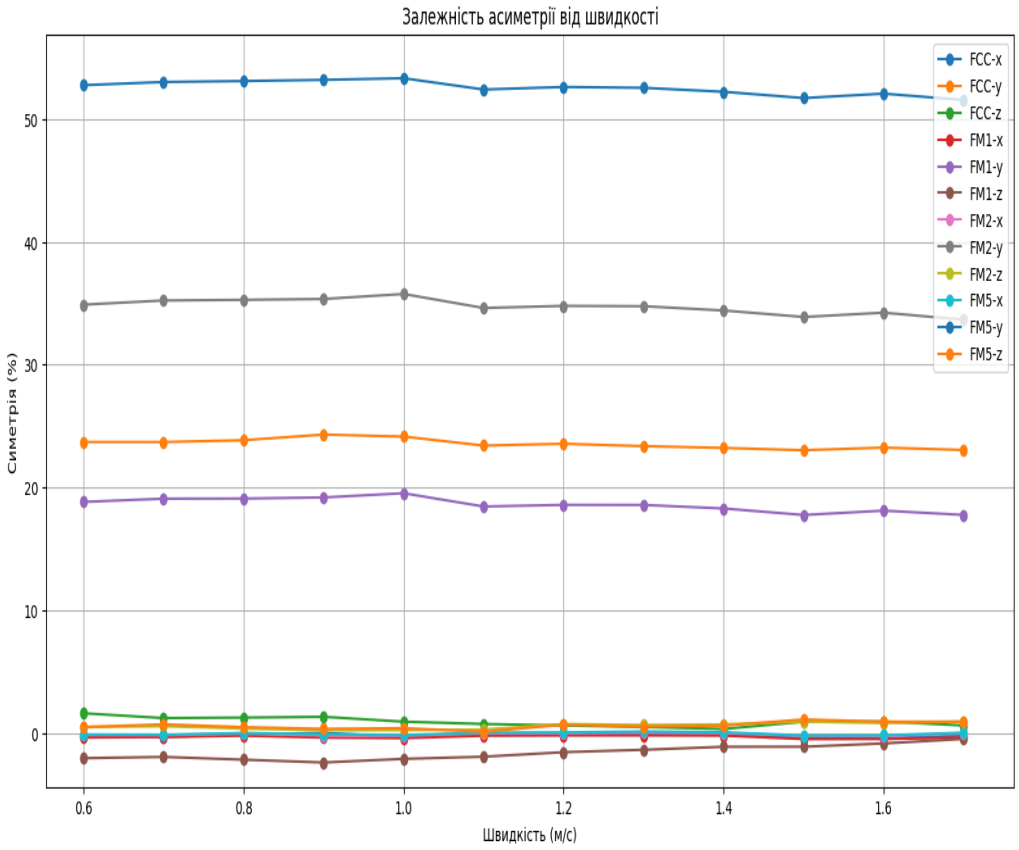


Рисунок А.11 – Залежність асиметрії від швидкості

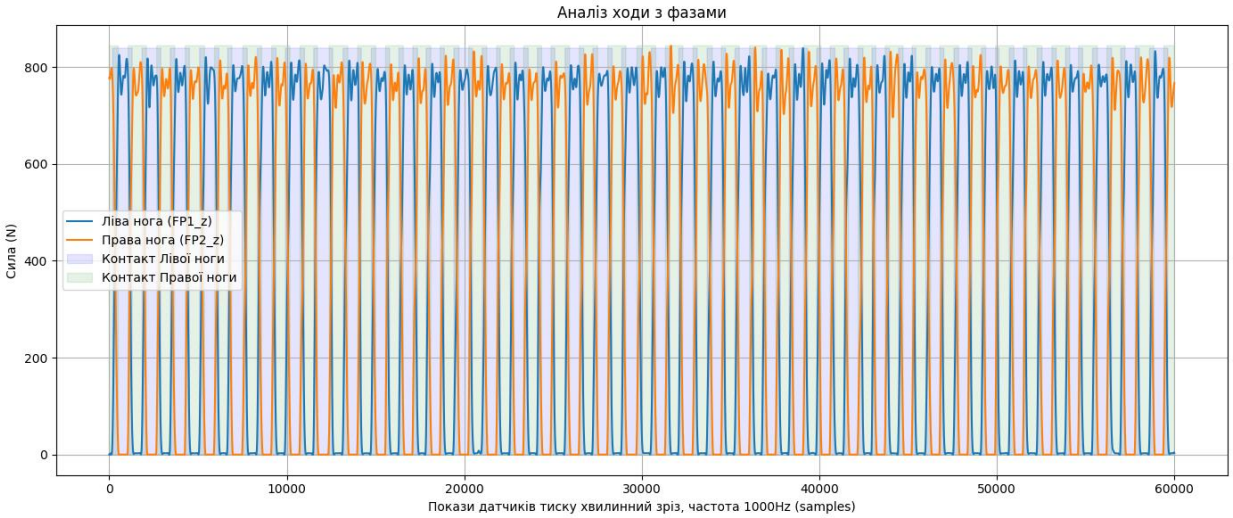


Рисунок А.12 – Графік аналіз фаз ходи людини

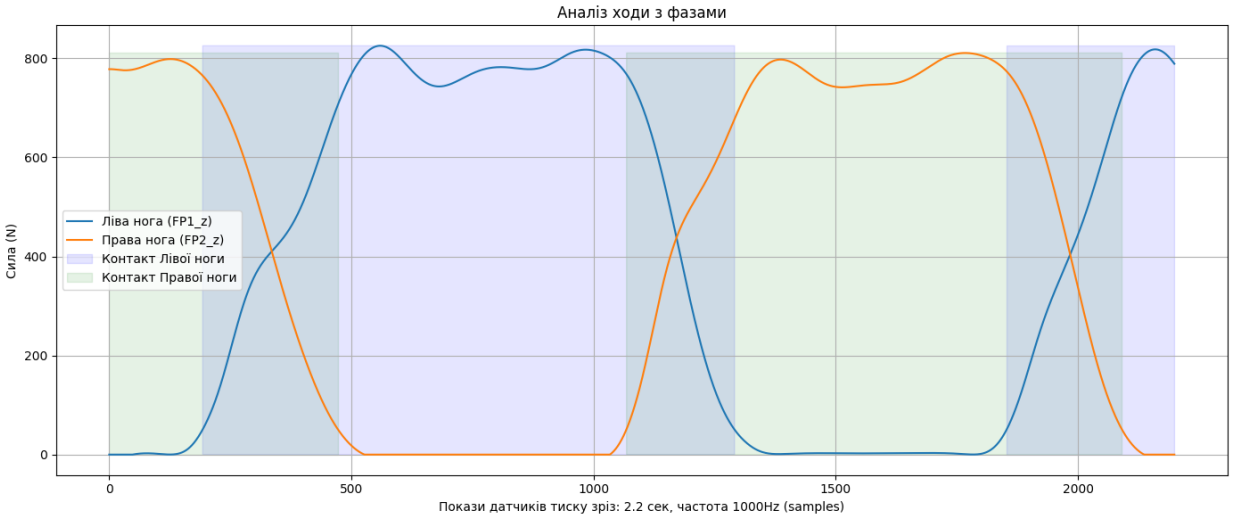


Рисунок А.13 – Аналіз фаз ходи людини за 2 кроки

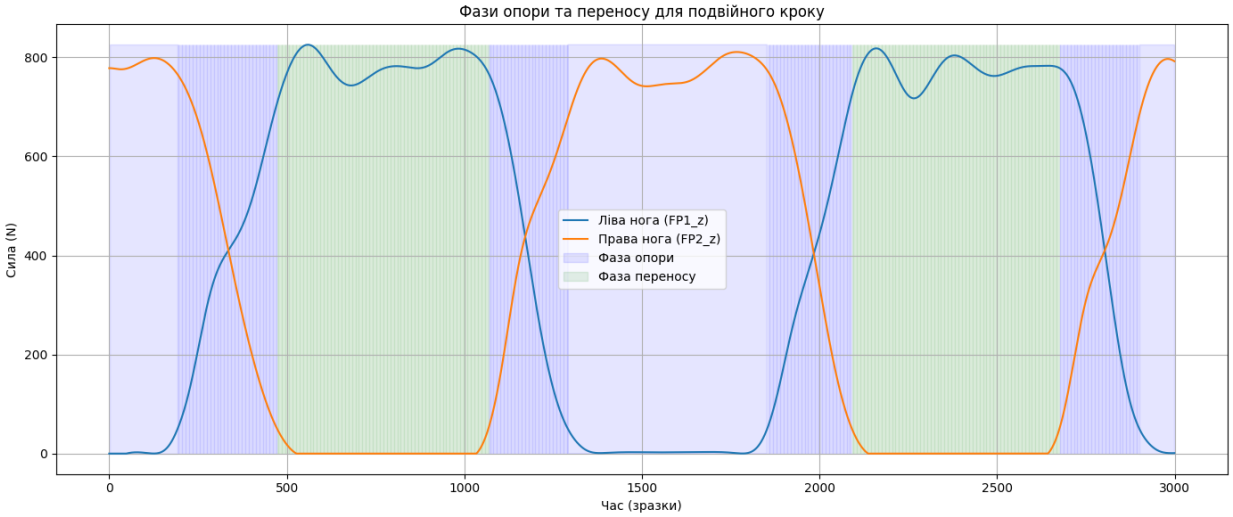


Рисунок А.14 – Фази опори та переносу для подвійного кроку

ДОДАТОК Б

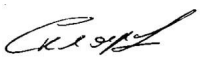
Текст програми

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

«ЗАТВЕРДЖУЮ»

Керівник кваліфікаційної роботи

 проф. Скляров В.В.

(підпис)

Моделі та методи метрологічного оцінювання особливостей ходи людини

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

РОЗРОБИВ:

ст. гр. ЗЯм-23-2

Потоцький М.В.

2024 р.

ЗАТВЕРДЖЕНО

Моделі та методи метрологічного оцінювання особливостей ходи людини

Текст програми

Листів 5

2024 р.

Програмний код застосунку:

```
import numpy as np
from filterpy.kalman import KalmanFilter


def apply_kalman_filter(data):
    # dim_x=1: Кількість змінних стану системи (одновимірний стан, 
наприклад, положення).
    # dim_z=1: Кількість вимірювань (одне вимірювання в кожен момент 
часу).
    kf = KalmanFilter(dim_x=1, dim_z=1)
    # Початковий стан,Наприклад, якщо data[0] = 10, то початковий стан 
— 10.
    kf.x = np.array([[data[0]]])
    # Модель переходу,F = 1 означає, що стан не змінюється (сталий 
стан).
    kf.F = np.array([[1]])
    # Модель спостереження, H = 1 означає, що ми безпосередньо 
вимірюємо стан (наприклад, положення).
    kf.H = np.array([[1]])
    # Початкова коваріація, P = 1 означає помірну впевненість у 
початковому стані.
    # Збільшення значення призведе до меншої впевненості та більшої 
адаптивності фільтра.
    kf.P *= 1
    # R = 0.1 вказує на низький шум (вимірювання досить точні)
    #  Більше значення змусить фільтр більше довіряти моделі, а не 
вимірюванням.
    kf.R *= 0.1  # Шум вимірювання
    # Процесний шум
    # Q = 0.01 передбачає майже стабільний процес із невеликими 
флуктуаціями.
    # Більше значення призведе до більшої гнучкості фільтра (краще 
адаптується до раптових змін).
    kf.Q *= 0.01

    # Фільтрація даних
    filtered_data = []
    for z in data:
        kf.predict()
        kf.update(z)
        filtered_data.append(kf.x[0, 0])
    return np.array(filtered_data)
from main.helpers.constant import DB_NAME

log = logging.getLogger(name)
```

```
def download_human_gait_db() -> None:
    log.info(f"Try to download {DB_NAME}")
    path = kagglehub.competition_download(DB_NAME)
    log.info(f"Successfully downloaded to the directory {path}")


if name == 'main':
    download_human_gait_db()
import pandas as pd
from pandas import DataFrame

from main.helpers.constant import PATH_TO_STORED_DB


def read_sensor_data(input_dataset: str) -> DataFrame:
    data = pd.read_csv(PATH_TO_STORED_DB.joinpath(input_dataset))
    return data

from pathlib import Path

PROJECT_MAIN_DIRECTORY = Path(file).resolve().parent.parent
PATH_TO_STORED_DB = PROJECT_MAIN_DIRECTORY.joinpath("resources/human-
gait-phase-dataset/versions/1/data")
DB_NAME = "dasmehdixtr/human-gait-phase-dataset"
это константы
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame

from main.helpers.constant import PATH_TO_STORED_DB


def read_sensor_data(input_dataset: str) -> DataFrame:
    data = pd.read_csv(PATH_TO_STORED_DB.joinpath(input_dataset))
    return data


def create_3d_model_of_foot_moving(input_dataset: str) -> None:
    data = read_sensor_data(input_dataset)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection="3d")
    ax.scatter(data["L_FCC_x"], data["L_FCC_y"], data["L_FCC_z"],
color="red", marker='o')
    ax.scatter(data["R_FCC_x"], data["R_FCC_y"], data["R_FCC_z"],
color="blue", marker=".")
    ax.set_xlabel("X: задньо-передній напрямок")
    ax.set_ylabel("Y: правий-лівий напрямок")
    ax.set_zlabel("Z: нижньо-верхній (вертикальний) напрямок")
    plt.title("3D Модель руху стоп людини")
    plt.show()
```

```
if __name__ == "__main__":
    create_3d_model_of_foot_moving("GP1_0.6_marker.csv")
import matplotlib
import pandas as pd
import matplotlib.pyplot as plt

from main.helpers.data_normalization import apply_kalman_filter
from main.helpers.file_helpers import read_sensor_data

# Set the backend for Matplotlib
matplotlib.use('TkAgg')  # Use 'TkAgg', 'WXAgg', or 'QtAgg' depending
on your environment

# Список осей та маркерів для аналізу
axes = ['x', 'y', 'z']
markers = ['FCC', 'FM1', 'FM2', 'FM5']
threshold = 10  # Допустимий поріг симетрії у %


def symmetry_index(left: float, right: float) -> float:
    """symmetry index calculation"""
    return (2 * (left - right) / (left + right)) * 100


def analyze_symmetry(file_name: str) -> pd.DataFrame:
    data = read_sensor_data(file_name)
    results = []
    # Аналіз кожної осі та кожного маркера
    for axis in axes:
        for marker in markers:
            # Координати для лівої і правої ноги
            L_data = apply_kalman_filter(data[f"L_{marker}_{axis}"])
def analyze_all_files(directory, speed):
    all_results = []
    all_asymmetry = []

    for file in os.listdir(directory):
        if file.endswith(f'{speed}_marker.csv'):
            file_path = os.path.join(directory, file)
            results, asymmetric = analyze_symmetry(file_path)

            # Додаємо результати для кожного файлу
            results['File'] = file
            asymmetric['File'] = file

            all_results.append(results)
            all_asymmetry.append(asymmetric)

    # Об'єднуємо результати
    all_results_df = pd.concat(all_results, ignore_index=True)
```

```
    all_asymmetry_df = pd.concat(all_asymmetry, ignore_index=True)

    # Зберігаємо результати у файли
    all_results_df.to_csv(f'all_results_speed_{speed}.csv',
index=False)
    all_asymmetry_df.to_csv(f'all_asymmetry_speed_{speed}.csv',
index=False)

    print(f"Загальні результати збережено в
'all_results_speed_{speed}.csv'")
    print(f"Асиметрії збережено в 'all_asymmetry_speed_{speed}.csv'")

    grouped = all_asymmetry_df.groupby(['Marker', 'Axis'])['Symmetry
Index (%)'].agg(['mean', 'std'])
    grouped.to_csv('group_analysis.csv')

    print("Груповий аналіз збережено в 'group_analysis.csv'")
# Побудова графіка аналізу типових відхилень
grouped.reset_index().pivot(index='Marker', columns='Axis',
values='mean').plot(kind='bar', figsize=(10, 6))
plt.title('Типові відхилення для групи')
plt.ylabel('Середній індекс симетрії (%)')
plt.xlabel('Маркер')
plt.grid(True)
plt.legend(title='Ось')
plt.savefig('group_analysis_plot.png')
plt.show()
# Кластерний аналіз асиметрії
scaler = StandardScaler()
scaled_data = scaler.fit_transform(all_asymmetry_df[['Symmetry Index
(%)']])
kmeans = KMeans(n_clusters=3, random_state=42)
all_asymmetry_df['Cluster'] = kmeans.fit_predict(scaled_data)

# Візуалізація кластерів
plt.figure(figsize=(10, 6))
plt.scatter(all_asymmetry_df['Symmetry Index (%)'],
all_asymmetry_df['Cluster'], c=all_asymmetry_df['Cluster'])
plt.title('Кластерний аналіз асиметрії')
plt.xlabel('Symmetry Index (%)')
plt.ylabel('Кластер')
plt.grid(True)
plt.savefig('cluster_analysis_plot.png')
plt.show()
import os

import matplotlib
import matplotlib.pyplot as plt
import pandas as pd

from main.helpers.constant import PATH_TO_STORED_DB
```

```
from main.measurements.symmetry_index import analyze_symmetry

# Set the backend for Matplotlib
matplotlib.use('TkAgg')  # Use 'TkAgg', 'WXAgg', or 'QtAgg' depending
on your environment


def analyze_speed_dependence(directory):
    all_results = []

    for file in os.listdir(directory):
        if file.endswith('_marker.csv'):
            speed = float(file.split('_')[1])  # Отримуємо швидкість
із назви файлу
            file_path = os.path.join(directory, file)
            results, asymmetric = analyze_symmetry(file_path)

            # Додаємо результати для кожної швидкості
            results['File'] = file
            results['Speed'] = speed
            all_results.append(results)

    # Об'єднання результатів
    all_results_df = pd.concat(all_results, ignore_index=True)
    # Аналіз залежності асиметрії від швидкості
    grouped = all_results_df.groupby(['Speed', 'Marker',
'Axis'])['Symmetry Index (%)'].mean().reset_index()
    grouped.to_csv('speed_analysis.csv', index=False)

    print("Аналіз швидкості збережено в 'speed_analysis.csv'")

    # Візуалізація залежності асиметрії від швидкості
    plt.figure(figsize=(12, 8))
    for marker in ['FCC', 'FM1', 'FM2', 'FM5']:
        marker_data = grouped[grouped['Marker'] == marker]
        for axis in ['x', 'y', 'z']:
            axis_data = marker_data[marker_data['Axis'] == axis]
            plt.plot(axis_data['Speed'], axis_data['Symmetry Index
(%)'], marker='o', label=f'{marker}-{axis}')

    plt.title('Залежність асиметрії від швидкості')
    plt.xlabel('Швидкість (м/с)')
    plt.ylabel('Симетрія (%)')
    plt.legend()
    plt.grid(True)
    plt.savefig('speed_analysis_plot.png')
    plt.show()

if __name__ == '__main__':
    file_path = PATH_TO_STORED_DB
    analyze_speed_dependence(file_path)
```