COMPUTER NETWORKS LAB ASSIGNMENT-4

Name: Suman Khara

Class: BCSE-III

Section: A2

Roll Number: 002210501098

Assignment Number: 4

Problem Statement: Implement CDMA with Walsh code.

DESIGN

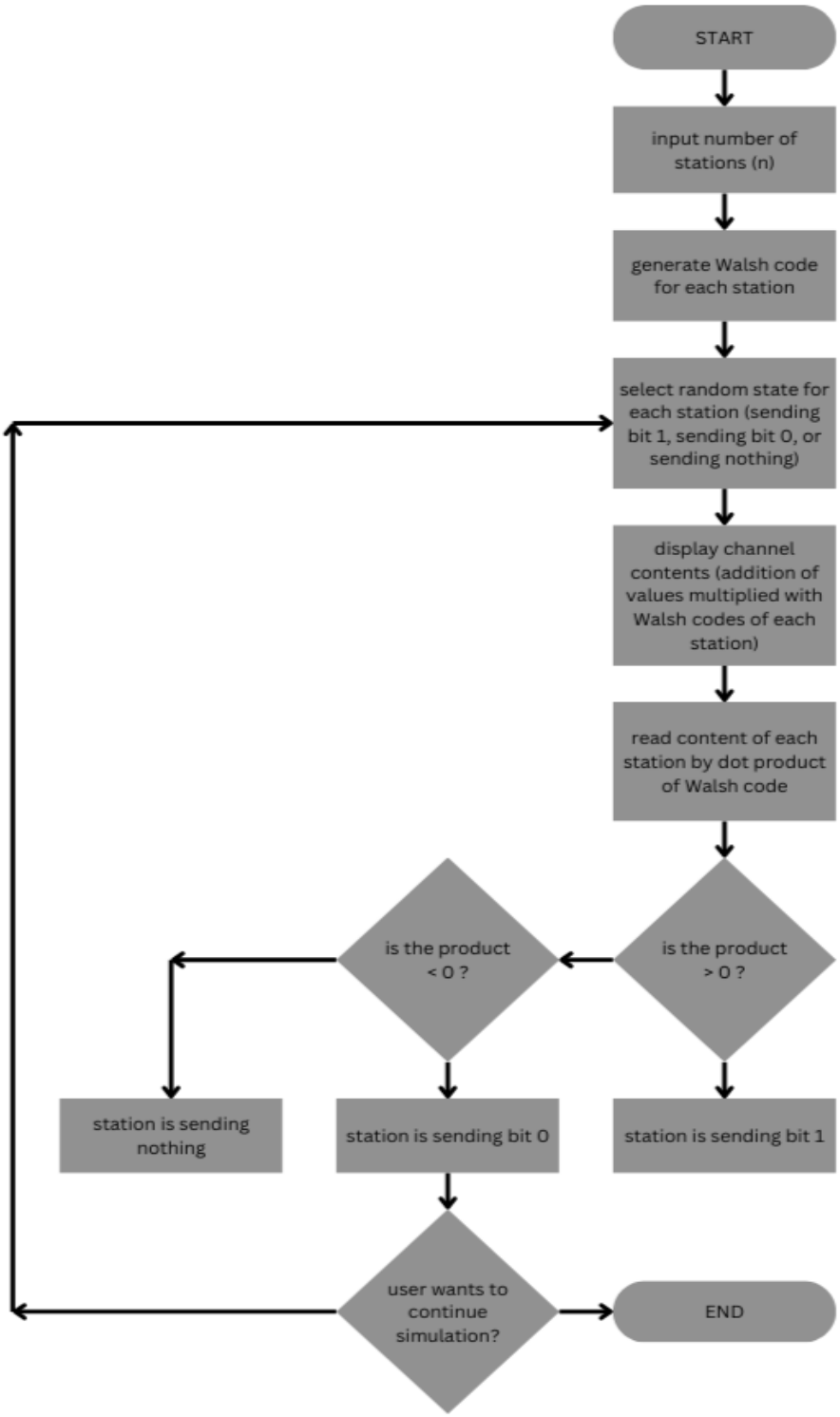

Fig 1: Flowchart of the CDMA simulation

The design of the **Simple_CDMA** project centers around simulating a CDMA-based communication system using Welsh codes to enable multiple stations to transmit simultaneously. Here's a breakdown of the key design aspects:

1. Welsh Code Generation

- **Purpose**: Welsh (Hadamard) codes are generated to ensure that each station has a unique, orthogonal code, which enables clear separation of signals on the shared channel.
- **Method**: The welsh_code_generator function recursively constructs a Hadamard matrix, which provides orthogonal codes for any power-of-2 number of stations.
- **Scalability**: The codes are generated based on the nearest power of two, making it possible to dynamically scale the simulation to any number of stations.

2. State Simulation of Stations

- **Purpose**: Each station can be in one of three states: sending 1, sending 0 (as -1), or sending nothing (0), allowing for more realistic behavior in a CDMA setting.
- **Randomized Transmission**: Each station's state is randomly determined in each iteration, simulating real-world CDMA behavior where stations do not constantly transmit.

3. Channel Content Calculation

- **Purpose**: The channel's combined signal content is calculated by summing the contributions of all active stations in each bit position.
- **Code Design**: This is achieved by a simple summation loop that uses the Welsh code and the state of each station to calculate the channel content vector.

4. Station Signal Decoding

- **Purpose**: Each station "reads" from the channel to determine what signal, if any, it can detect.
- **Signal Detection**: By computing the dot product of its Welsh code with the channel content and normalizing, each station determines whether it reads a 1, 0, or no signal.

5. Iterative User Interaction

- **Purpose**: Each iteration allows users to see individual states, channel content, and read values, adding a step-by-step visualization of how CDMA works.
- **Exit Option**: Users can exit or continue the simulation, making it flexible for testing or demonstration purposes.

This modular design, with distinct components for Welsh code generation, state setting, channel calculation, and reading, makes the project adaptable for further enhancements, such as more complex states or additional error-checking mechanisms.

IMPLEMENTATION

```
import random
import numpy as np

def welsh_code_generator(order):
    if order == 1:
        return np.array([[1]])
    prev = welsh_code_generator(order // 2)
    top = np.hstack((prev, prev))
    bottom = np.hstack((prev, -prev))
    return np.vstack((top, bottom))

def welsh_code_list(n):
    order = 1
    while order < n:
        order *= 2
    codes = welsh_code_generator(n)
    return codes[:n, :]

def simulate(n, codes):
    bits = len(codes[0])
    sent = []
    for i in range(n):
        send = random.choice([1, -1, 0])
        sent.append(send)
        if send == 0:
            x = "nothing"
        elif send == 1:
            x = "1"
```

```
            else:
                x = "0"
            print(f"station {i+1} sending {x}")
        channel = np.zeros(bits)
        for i in range(n):
            channel += sent[i] * codes[i]
        print(f"channel content: {channel}")
        for i in range(n):
            reading = np.dot(codes[i], channel) / bits
            if reading > 0:
                signal = "1"
            elif reading < 0:
                signal = "0"
            else:
                signal = "nothing"
            print(f"{signal} is read from station {i+1}")

n = int(input("number of stations: "))
codes = welsh_code_list(n)
for i in range(n):
    print(f"station {i+1} code: {codes[i]}")
while True:
    next = input("enter x to exit, any other to continue\n")
    if next == 'x':
        break
    simulate(n, codes)
```

Output:

number of stations: 8

station 1 code: [1 1 1 1 1 1 1 1]

station 2 code: [1 -1 1 -1 1 -1 1 -1]

station 3 code: [1 1 -1 -1 1 1 -1 -1]

station 4 code: [1 -1 -1 1 1 -1 -1 1]

station 5 code: [1 1 1 1 -1 -1 -1 -1]

station 6 code: [1 -1 1 -1 -1 1 -1 1]

station 7 code: [1 1 -1 -1 -1 -1 1 1]

station 8 code: [1 -1 -1 1 -1 1 1 -1]

enter x to exit, any other to continue

station 1 sending nothing

station 2 sending 1

station 3 sending 0

station 4 sending nothing

station 5 sending nothing

station 6 sending 1

station 7 sending 1

station 8 sending 0

channel content: [1. -1. 3. -3. -1. -3. 1. 3.]

nothing is read from station 1

1 is read from station 2

0 is read from station 3

nothing is read from station 4

nothing is read from station 5

1 is read from station 6

1 is read from station 7

0 is read from station 8

enter x to exit, any other to continue

x

Explanation:

In this output, the simulation iterates through one round of communication for **8 stations** using their assigned Welsh codes. Here's what happens step-by-step:

1. **Transmission**:
    - Each station randomly decides to send either 1, 0, or nothing:
        - **Sending 1**: The station transmits its Welsh code.

        - **Sending 0**: The station transmits the negated version of its Welsh code.
        - **Sending Nothing**: The station does not transmit, adding [0, 0, ...] to the channel.
    - In this iteration:
        - Stations 2, 6, and 7 send 1.
        - Stations 3 and 8 send 0.
        - Stations 1, 4, and 5 send nothing.
2. **Channel Content Calculation**:
    - The channel content reflects the combined contributions of all stations. Each station's signal (its Welsh code or a negated version, if sending 0) is added to the channel vector.
    - The result is [1, -1, 3, -3, -1, -3, 1, 3], representing the total signal at each bit position in the Welsh code.
3. **Decoding by Each Station**:
    - Each station reads the channel content to determine what signal it can detect.
        - If the dot product of its Welsh code with the channel is positive, it reads a 1.
        - If it's negative, it reads a 0.
        - If the result is zero, it reads "nothing."
    - The results:
        - Stations 2, 6, and 7 correctly read 1, indicating a positive result.
        - Stations 3 and 8 read 0.
        - Stations 1, 4, and 5, which did not transmit, read "nothing."

Thus, the output shows how each station can decode signals from the mixed channel content, demonstrating Welsh code-based CDMA's ability to separate signals from multiple simultaneous transmitters.

ANALYSIS

1. Assumptions and Simplifications

- **Ideal Channel Conditions**: The simulation assumes no noise or interference, unlike real-world CDMA, where factors like noise and fading affect signal quality.
- **No Flow Control or Error Correction**: Protocols for error handling are omitted, focusing purely on signal encoding and decoding.
- **Static Welsh Code Assignment**: Each station is assigned a fixed Welsh code, suitable for a static network but not ideal for dynamic networks.

2. Channel Content Calculation

- **Sum of Signal Contributions**: Each station sends either 1, 0, or nothing, with the channel content calculated as a sum of these signals.
- **Resource Intensity**: Matrix operations increase in complexity with more stations, requiring optimization for larger systems.
- **Orthogonal Code Property**: Welsh codes allow signal separation, demonstrating how CDMA enables multiple stations to transmit simultaneously without interference.

3. Signal Decoding by Each Station

- **Decoding Process**: Each station decodes its signal by taking the dot product with the channel content.
- **Accurate Signal Detection**: Orthogonality ensures each station can accurately detect its signal, with positive/negative results indicating 1 or 0.
- **Limitations in Signal Clarity**: Real-world applications may face minor cross-talk, but the simulation demonstrates ideal separation.

4. Resource Considerations

- **Output Management**: Real-time output can be resource-intensive; logging results to a file would be more efficient for large-scale simulations.

- **Real-Time User Interaction**: User control per iteration aids testing but continuous automation would be needed for real applications.

5. Practical Implications and Limitations

- **Comparison to Real CDMA Systems**: Real systems involve interference, dynamic codes, and error-checking, unlike this simplified model.
- **Accuracy vs. Complexity**: This project simplifies CDMA to show core functionality but does not cover real-world challenges.
- **Potential for Extensions**: Future enhancements could add noise, dynamic code assignment, and error correction for a more realistic model.

COMMENT

The assignment of implementing CDMA with Walsh code provided valuable insights into the fundamentals of Code Division Multiple Access (CDMA) communication. By simulating an environment where multiple stations use Welsh codes to share a common channel, this assignment highlighted the effectiveness of orthogonal codes for signal separation in multi-user scenarios. While the model simplifies real-world complexities—such as noise, interference, and error handling—it successfully demonstrates the core principles of CDMA.

This project underscores the importance of code orthogonality in enabling simultaneous communication and serves as a strong foundation for understanding CDMA's potential in practical applications. Future explorations could expand on this model to incorporate real-world conditions, enhancing our comprehension of CDMA's versatility and limitations in modern communication networks.
