

УДК 004.27

Неласа Г.В.¹, Зайцева А.О.², Дубровін В.І.³, Ковальов І. Є.²

¹ канд., техн., наук, доц., доц. каф. ІБтаН НУ «Запорізька політехніка»

² студ. гр. БК-813М НУ «Запорізька політехніка»

³ канд., техн., наук, проф., проф. каф. ПЗ НУ «Запорізька політехніка»

КВАНТОВА РЕАЛІЗАЦІЯ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ ДОБЕШИ

Вейвлет є математичною функцією, яку використовують для поділу заданої функції або сигналу безперервного часу на компоненти масштабу. Вони є масштабованими та трансльованими копіями (відомими як «дочірні вейвлети») коливальної форми сигналу кінцевої довжини або швидкозгасаючих (відомих як «материнський вейвлет») [1].

Вейвлети Добеші є сімейством ортогональних вейвлетів, які визначають дискретне вейвлет-перетворення та мають максимальну кількість моментів зникнення для конкретно вибраної опори.

Дискретне вейвлет-перетворення сигналу x обчислюється, пропускаючи його через низку фільтрів. Спочатку зразки пропускають через фільтр низьких частот з заданою імпульсною характеристикою. Сигнал також розкладається одночасно за допомогою фільтра високих частот. Вихідні дані дають коефіцієнти деталізації (від фільтра високих частот) і коефіцієнти апроксимації (від фільтра низьких частот). Важливо, щоб два фільтри були пов'язані один з одним, і вони відомі як квадратурний дзеркальний фільтр. [2]

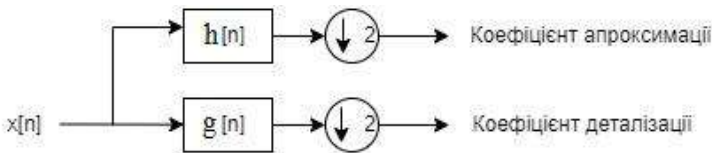


Рисунок 1 – Блок-схема аналізу фільтрів

Вейвлет-перетворення Добеші D4 має чотири коефіцієнти функції масштабування

$$h_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}}, h_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}}, h_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}. \quad (1)$$

де h_N – фільтр апроксимації.

Нижче наведено як вирахувати та чому дорівнюють значення коефіцієнтів вейвлет-функції:

$$g_k = (-1)^k h_{N-1-k}, \quad (2)$$

$$g_0 = h_3, g_1 = -h_2, g_2 = h_1, g_3 = -h_0. \quad (3)$$

де h_N – фільтр апроксимації, g_k – фільтр деталізації.

Реалізовувати вейвлет-перетворення Добеши D4 на квантах будемо за допомогою бібліотеки PennyLane та хмарного середовища для роботи з Jupyter Notebook – Google Colab.

Для початку треба встановити та підключити бібліотеки pennylane та numpy, присвоївши їм qml та np, відповідно.

Визначаємо коефіцієнти для перетворення Добеши D4. Якщо коротко, то коефіцієнти апроксимації (h) спрощують сигнал, зберігаючи при цьому основні ознаки. А от коефіцієнти деталізації (g) визначають, як виділити деталі сигналу, зокрема, зміни в амплітуді або частоті.

```
h0 = (1 + np.sqrt(3)) / (4 * np.sqrt(2))
h1 = (3 + np.sqrt(3)) / (4 * np.sqrt(2))
h2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
h3 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))
g0 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))
g1 = (-3 - np.sqrt(3)) / (4 * np.sqrt(2))
g2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
g3 = (-1 - np.sqrt(3)) / (4 * np.sqrt(2))
approximation_filter = np.array([h0, h1, h2, h3])
detail_filter = np.array([g0, g1, g2, g3])
```

Треба прописати кількість кубітів, яку будемо використовувати. Наприклад, `wavelet_qubits = 4`. Також задаємо довжину сигналу (2^n , n – кількість кубітів). Тобто, `signal_length = 16`.

Створюємо квантовий пристрій, на якому будемо реалізовувати вейвлет.

```
dev_d4 = qml.device("default.qubit", wires=wavelet_qubits)
```

Переходимо до квантової реалізації перетворення вейвлета Добеши D4.

```
@qml.qnode(dev_d4)
```

```
def d4_wavelet_transform(input_state):
```

```
    qml.MottonenStatePreparation(input_state, wires=range(wavelet_qubits))
```

Декоратор визначає функцію, яка буде описувати вейвлет-перетворення D4 для обробки вхідного стану `input_state`, який є вхідним параметром – масивом, що представляє собою класичний сигнал, який буде квантово підготовлений для обробки. Операція, яка прописана в третьому рядку, є в

самій бібліотеці `pennylane`. Після її застосування вхідний класичний сигнал буде перетворений на квантовий стан і збережений у `wavelet_qubits`.

А тепер найцікавіше. За допомогою циклів прописуємо перший етап – апроксимації та другий – деталізації.

```
# Перший етап: апроксимація
for i in range(0, wavelet_qubits - 1, 2): # Обробка парних індексів
    qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
    qml.CNOT(wires=[i, i + 1])
    qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)
for i in range(1, wavelet_qubits - 1, 2): # Обробка непарних індексів
    qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
    qml.CNOT(wires=[i, i + 1])
    qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)

# Другий етап: деталізація
for i in range(0, wavelet_qubits - 2, 2): #Обробка парних індексів
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])

for i in range(1, wavelet_qubits - 2, 2): # Обробка непарних індексів
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])
return qml.state()
```

Кожен парний/непарний кубіт обробляється через обертання, яке змінює його стан відповідно до фільтра для апроксимації. Потім між кожною парою кубітів (парні/непарні індекси та їхні наступні сусіди) виконується операція CNOT, щоб "передати" або коректувати квантову інформацію. Після цього, другий кубіт в парі (з індексом $i + 1$) також обертається згідно з другим коефіцієнтом фільтра для апроксимації.

На етапі деталізації робимо те саме, тільки спочатку перший кубіт у парі передає інформацію за допомогою операції CNOT до другого кубіта, після чого на другому кубіті виконується обертання за допомогою операції RY, яка залежить від коефіцієнта фільтра деталізації. Потім, виконується ще одна операція CNOT для корекції стану кубітів.

Тепер створюємо та нормалізуємо вхідний сигнал. А оскільки сигнал може мати довжину менше ніж 16 елементів, потрібно додати достатньо нулів, щоб досягти бажаної довжини.

```

padded_signal_length = 16
input_signal_padded = np.concatenate((input_signal_normalized,
np.zeros(padded_signal_length - len(input_signal_normalized))))

```

Після чого викликається функція, яку раніше прописували для сигналу `input_signal_padded`. Це той самий сигнал, який був нормалізований і доповнений нулями, щоб його довжина була ступенем двійки.

Тепер передостанній важливий блок. Потрібно створити функцію, яка буде аналізувати квантовий стан для того, щоб отримати коефіцієнти, які описують стан кубітів. Після чого розподіляються амплітуди квантового стану на коефіцієнти апроксимації та деталізації.

```

def extract_coefficients(quantum_state):
    amplitudes = np.abs(quantum_state) # Підрахунок амплітуд для
отримання коефіцієнтів

```

```

mid = len(amplitudes) // 2
approximation_coeffs = amplitudes[:mid]
detail_coeffs = amplitudes[mid:]
return approximation_coeffs, detail_coeffs

```

В кінці виводимо результат на екран.

```

print("Quantum State (D4 Wavelet):", quantum_state_d4)
print("Approximation Coefficients:", approximation_coeffs)
print("Detail Coefficients:", detail_coeffs)

```

```

Quantum State (D4 Wavelet): [-0.18151887+8.17480008e-18j -0.2575212 +1.19275975e-17j
-0.3929905 +1.11635249e-17j  0.15036484-5.67434333e-18j
 0.44763187-1.67789778e-17j  0.08147727-2.55938875e-18j
 0.10865971-4.28798240e-18j  0.62190698-2.73113115e-17j
-0.01573924+5.07352901e-18j -0.25104173+6.18044184e-17j
-0.01336068+1.04213696e-17j  0.02632096-3.95632418e-18j
 0.15882805-2.37694630e-17j  0.01458512-5.65417615e-18j
-0.08876239+2.07523966e-17j -0.14365144+3.29104707e-17j]
Approximation Coefficients: [0.18151887 0.2575212  0.3929905  0.15036484 0.44763187 0.08147727
 0.10865971 0.62190698]
Detail Coefficients: [0.01573924 0.25104173 0.01336068 0.02632096 0.15882805 0.01458512
 0.08876239 0.14365144]

```

Рисунок 2 – Результат виконання коду

В результаті, отримали квантовий стан, в якому можна побачити комплексні числа. Реальна частина сигналу відповідає за саму величину амплітуди, а уявна частина – за фазу цього сигналу. Оскільки всі уявні частини значень дуже малі (порівняно з реальними частинами), це свідчить, що більшість квантових станів насправді не мають значного зсуву фази, а є майже чистими реальними числами.

Першою половиною квантового стану є апроксимаційні коефіцієнти, які в свою чергу представляють низькочастотні компоненти сигналу, тобто основну частину інформації про сигнал, де знаходяться його глобальні характеристики. Друга половина – деталізаційні коефіцієнти, які відповідають за високочастотні компоненти сигналу (тобто дрібніші деталі, шуми або зміни в сигналі).

Також, варто дещо уточнити. З класичної точки зору, вейвлет-перетворення зазвичай зменшує розмірність сигналу вдвічі при кожному рівні перетворення. Однак, вихідні коефіцієнти квантового перетворення можуть не зменшуватися через природу квантових обчислень. Квантові стани можуть представляти суперпозицію (стан може одночасно мати багато значень), тому кількість значень у виході може відображати більш комплексну структуру сигналу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. «Wavelet» – Електронний ресурс – Режим доступу: <https://en.wikipedia.org/wiki/Wavelet>
2. «Discrete wavelet transform» – Електронний ресурс – Режим доступу: https://en.wikipedia.org/wiki/Discrete_wavelet_transform