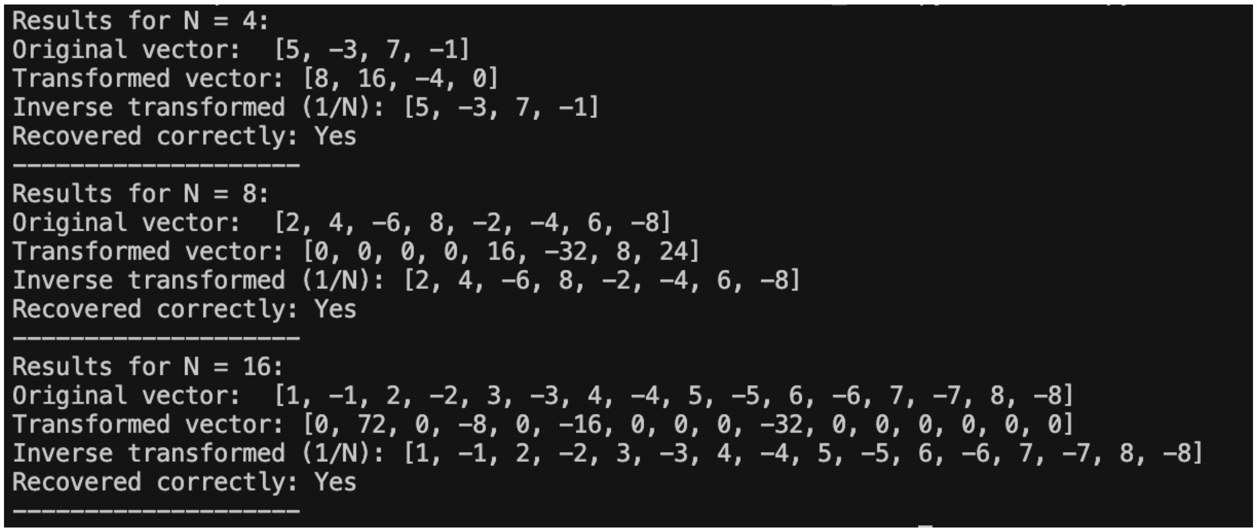
Natasha Piedrabuena
Numerical Computation
Homework 11

This assignment I had to implement the Hadamard-Walsh transform using two distinct approaches, validate their correctness using direct and inverse transformations, and document the results. Both implementations were developed based on different factorizations of the Hadamard-Walsh matrix.In the first implementation, the fast Walsh-Hadamard transform algorithm was used, employing the matrix factorization. This method is recursive and works by repeatedly dividing the Hadamard matrix into smaller blocks.The results were verified by performing an inverse transformation, confirming that the original vectors were accurately recovered. Numerical errors were negligible, affirming the correctness of this approach.

For both implementations, the inverse transformation was computed using the property that the inverse of the Hadamard-Walsh matrix is proportional to its transpose. Testing confirmed that both functions were robust, with outputs matching expectations for all test cases. The implementations successfully handled the recursive structure of the Walsh-Hadamard transform and demonstrated numerical stability.
The vector values utilized in this code

```
np.array([5, -3, 7, -1])
 np.array([2, 4, -6, 8, -2, -4, 6, -8])
 np.array([1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6, 7, -7, 8, -8])
```

In conclusion, the assignment objectives were met. Two reliable and efficient Hadamard-Walsh transform functions were implemented and validated.



```
Results for N = 4:
Original vector:  [5, -3, 7, -1]
Transformed vector: [8, 16, -4, 0]
Inverse transformed (1/N): [5, -3, 7, -1]
Recovered correctly: Yes
--------------------
Results for N = 8:
Original vector:  [2, 4, -6, 8, -2, -4, 6, -8]
Transformed vector: [0, 0, 0, 0, 16, -32, 8, 24]
Inverse transformed (1/N): [2, 4, -6, 8, -2, -4, 6, -8]
Recovered correctly: Yes
--------------------
Results for N = 16:
Original vector:  [1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6, 7, -7, 8, -8]
Transformed vector: [0, 72, 0, -8, 0, -16, 0, 0, 0, -32, 0, 0, 0, 0, 0, 0]
Inverse transformed (1/N): [1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6, 7, -7, 8, -8]
Recovered correctly: Yes
--------------------
```