Suppose \( \alpha(t) \) is a smooth function for \( t \in [0,1] \) with \( 0 \leq \alpha(t) < 1 \). For any integer \( N \), define the function

\[ c_N(j) = \lceil N \cdot \alpha(j/N) \rceil \]

for \( 0 \leq j \leq N - 1 \). Given \( f_0, \ldots, f_{N-1} \), the partial Fourier transform of size \( N \) computes \( u_0, \ldots, u_{N-1} \) given by

\[ u_j = \sum_{0 \leq k < c_N(j)} e^{2\pi i jk/N} f_k. \]

The goal of this small project is to design and implement (in Matlab) an algorithm that computes \( \{u_j\}_{0 \leq j \leq N-1} \) in \( O(N \log^2 N) \) time. The main difficulty comes from the \( j \)-dependent summation constraints \( 0 \leq k < c_N(j) \). If there were no summation constraints, this is simply a discrete Fourier transform. For simplicity, let us assume that \( N \) is an integer power of 2.

Hint: (1) Define the summation domain \( D = \{(j,k) | 0 \leq k < c_N(j)\} \). Decompose the domain \( D \) recursively into dyadic squares (see Figure 1).

![Figure 1: \( \alpha(t) \) is a Gaussian function. (a) \( D \) is the region below the curve. (b) \( D \) is partitioned hierarchically into dyadic squares.](image)

(2) For each square of size \( s \times s \) in the constructed decomposition, is there a fast algorithm that performs the computation associated with this square in \( O(s \log s) \) steps (see Problem 3 of the homework)? How many squares of size \( s \times s \) are there? Recall that the curve \( \alpha(t) \) is smooth. What is the number of steps that are used on all the squares of size \( s \times s \)?

(3) How many different values of \( s \) are there? What is the total number of steps?