Sparse Fourier Transform

Andrii Hlyvko, Wentao Zhu

Motivation

- The Fast Fourier Transform algorithm is one of the most important algorithms in signal processing. Computational complexity O(NlogN).
- With the rise of big data the size of the input signal becomes very large. Need to process terabytes of data.
- Many signals like images, video, GPS signals, etc. are sparse in the frequency domain. So we can approximate the FFT by using a subset of the input signal.
- Sparse Fourier Transform runs in O(KlogN), where K << N is the sparsity os the signal.

Applications

- Spectrum Sensing: One needs to scan a wide band frequency channel for frequencies that are in use. This needs to be done very fast. The Sparse FFT can be used to check which frequencies are in use in short amount of time.

- GPS synchronization: A satellite sends a code to a GPS receiver. The receiver gets the code corrupted by noise. The receiver has to tell where the code starts and ends by computing the FFT of the code and the signal to get the shift maximizing the correlation.

Conditions for Sparse Fourier Transform

For successful approximation we need to make some assumptions about the input signal.

- We assume that the input signal has K<<N large frequencies. So we can approximate the FFT from a subsampled input.
- From a lemma in the paper if N is a power of two and sigma is a uniformly random odd number in [N], then Pr[sigma ∈ [-C,C]] ≤ 4C/N. Bound on the probability that a non-zero frequency bins to the same bucket.
- The filter used is flat window concentrated in time and frequency. We used Dolph-Chebyshev function convolved with box-car. This limits the leakage of frequencies into other buckets.

Datasets

- For a given N we randomly generated K large frequencies and took the FFT to get the time signal. We generated 3 datasets.
- Dataset 1 consists of signals of various input sizes N with a sparsity of K=50.
- Dataset 2 consists of signals of length N=1048576 with various K.
- Dataset 3 consists of signals of length N=262144, K=50, and with various levels of noise added to the signal.

Algorithm Steps

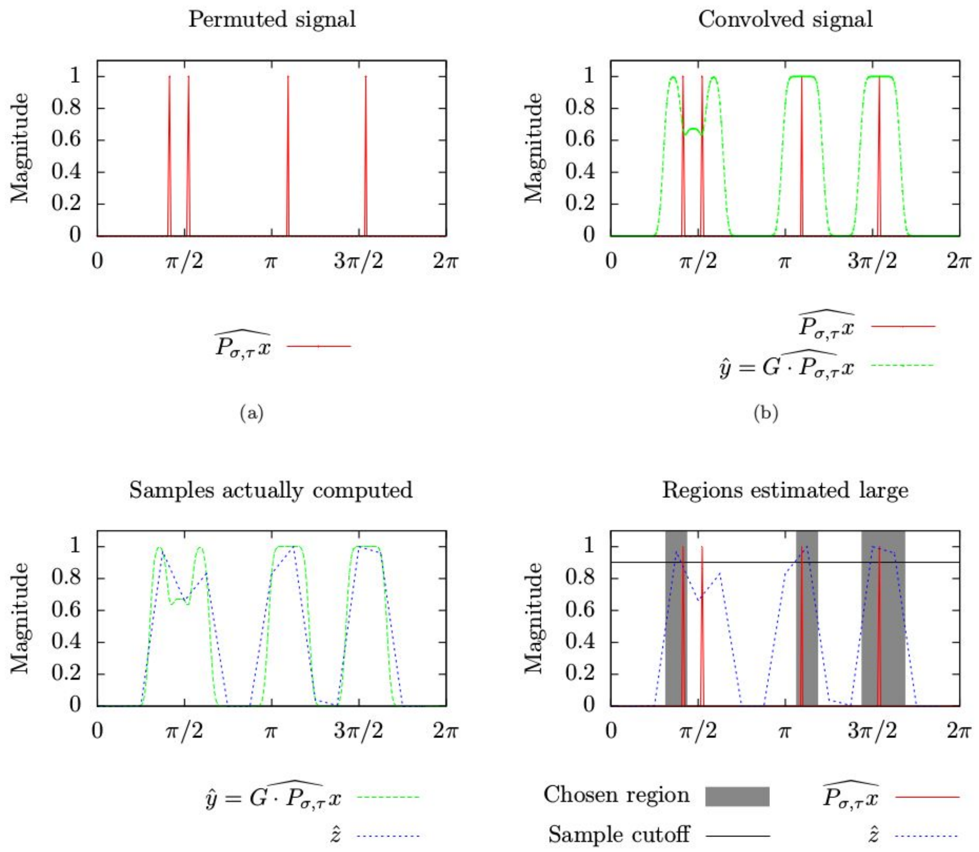
- Input: x[n], k sparse time domain signal.
- Output: $L = [(a_0, w_0), \dots (a_k, w_k)]$ coefficients for large frequencies.

1. Identify the large frequencies.
2. Estimate Coefficients
3. Remove the found large frequency contributions from the signal.
4. Repeat.

Frequency Identification

- First we downsample the signal using random permutation. Since there is probability of aliasing we do downsampling several times to avoid collisions.
- If $x' = x[\sigma i + \tau]$, then its Fourier transform $X' = x^*\omega^{-\tau i}$. So by randomly permuting the time signal we permute the spectra. This is to increase separation between large frequencies. This will result in subsampling the frequency spectrum.
- We then convolve with the flat window function to bin the frequencies. And threshold to get identify the set of largest frequency coefficients.

Identification Example from the paper

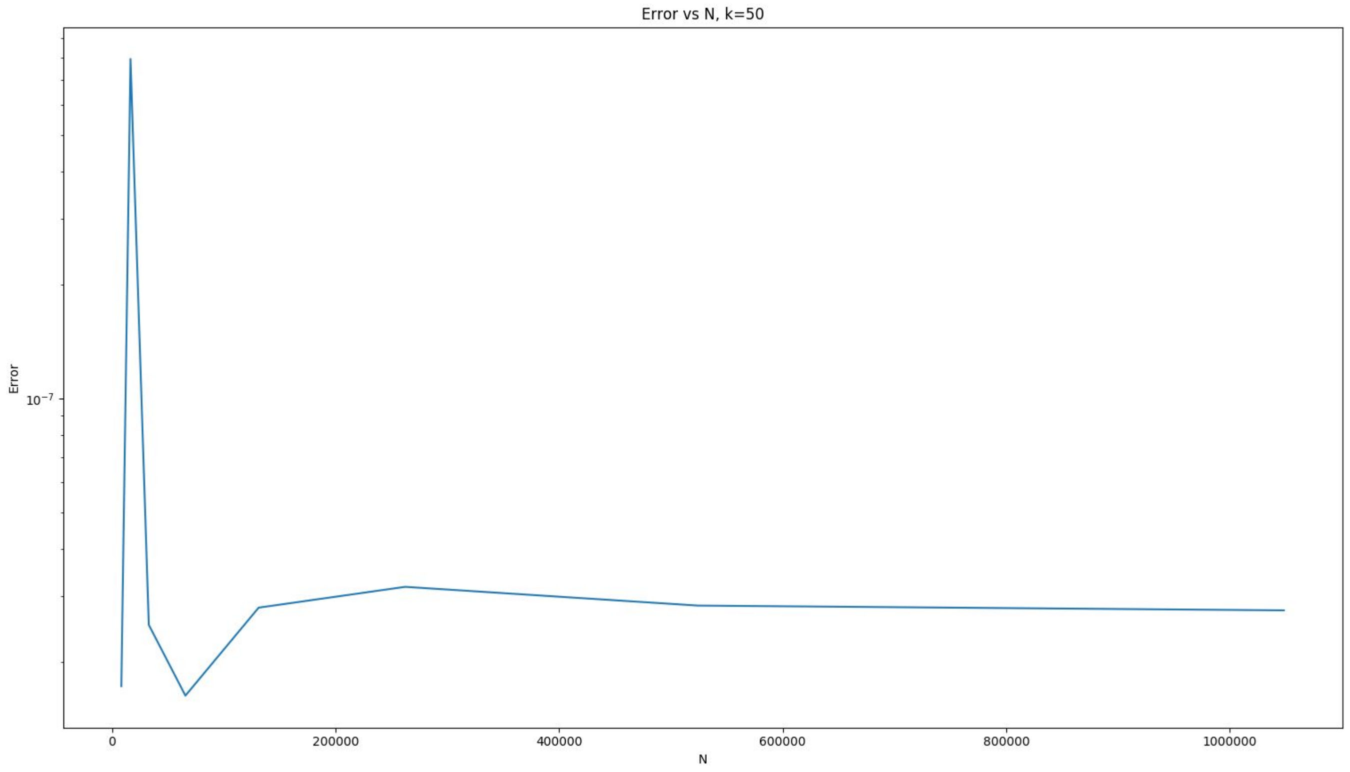


Estimating Coefficients

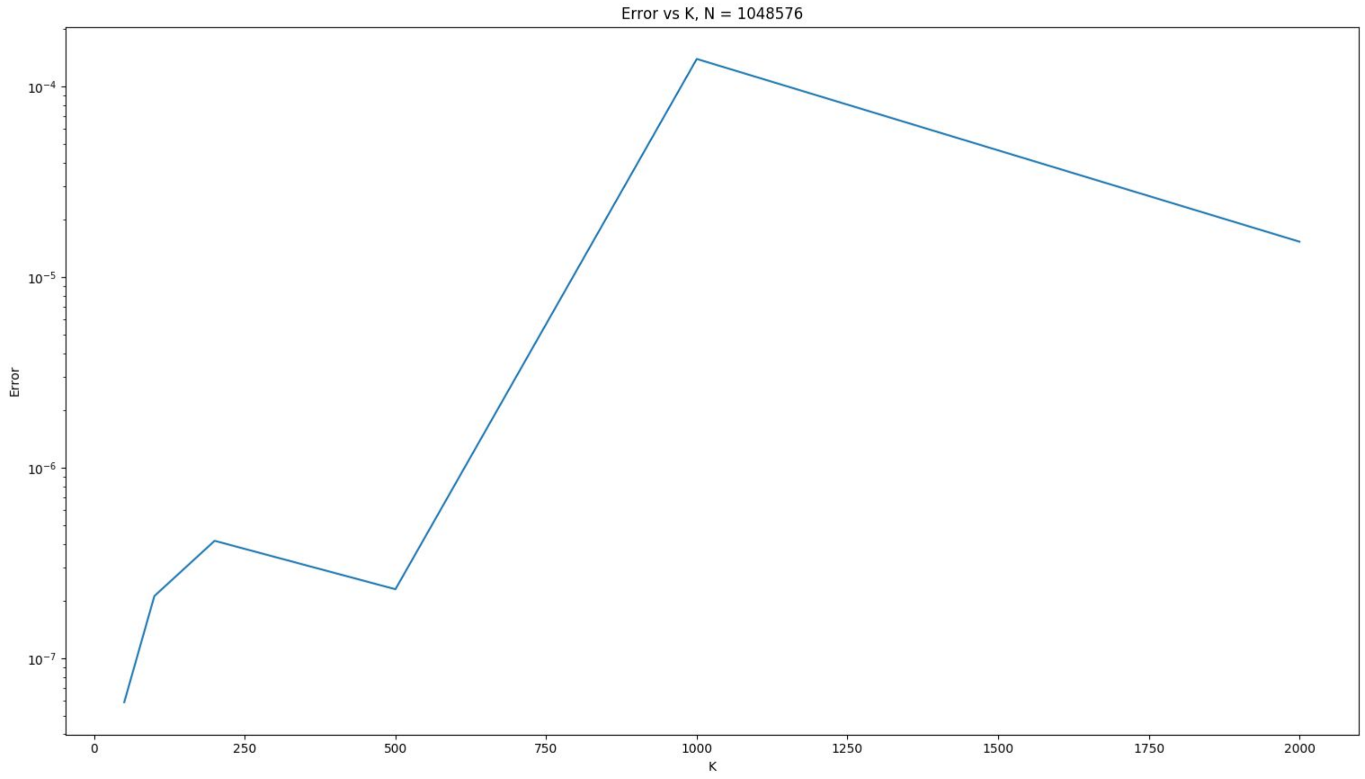
- We can estimate the coefficients obtained in stage 1 using L << N uniformly random samples of the signal.

$$\hat{f}'_\omega := \frac{1}{L} \cdot \sum_{l=1}^{L} f_{u_l} e^{-2\pi i \cdot \omega \cdot u_l / N}.$$

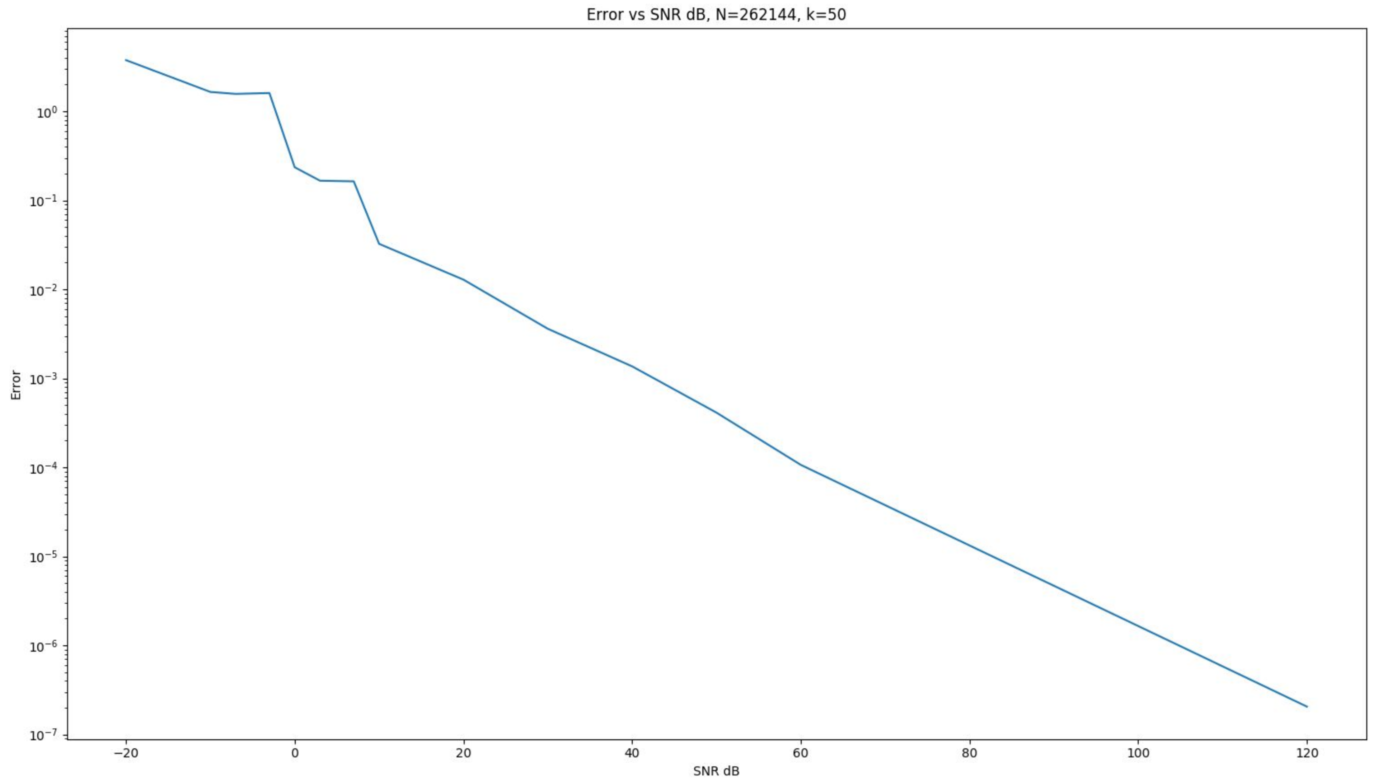
Sparse FFT error for different N



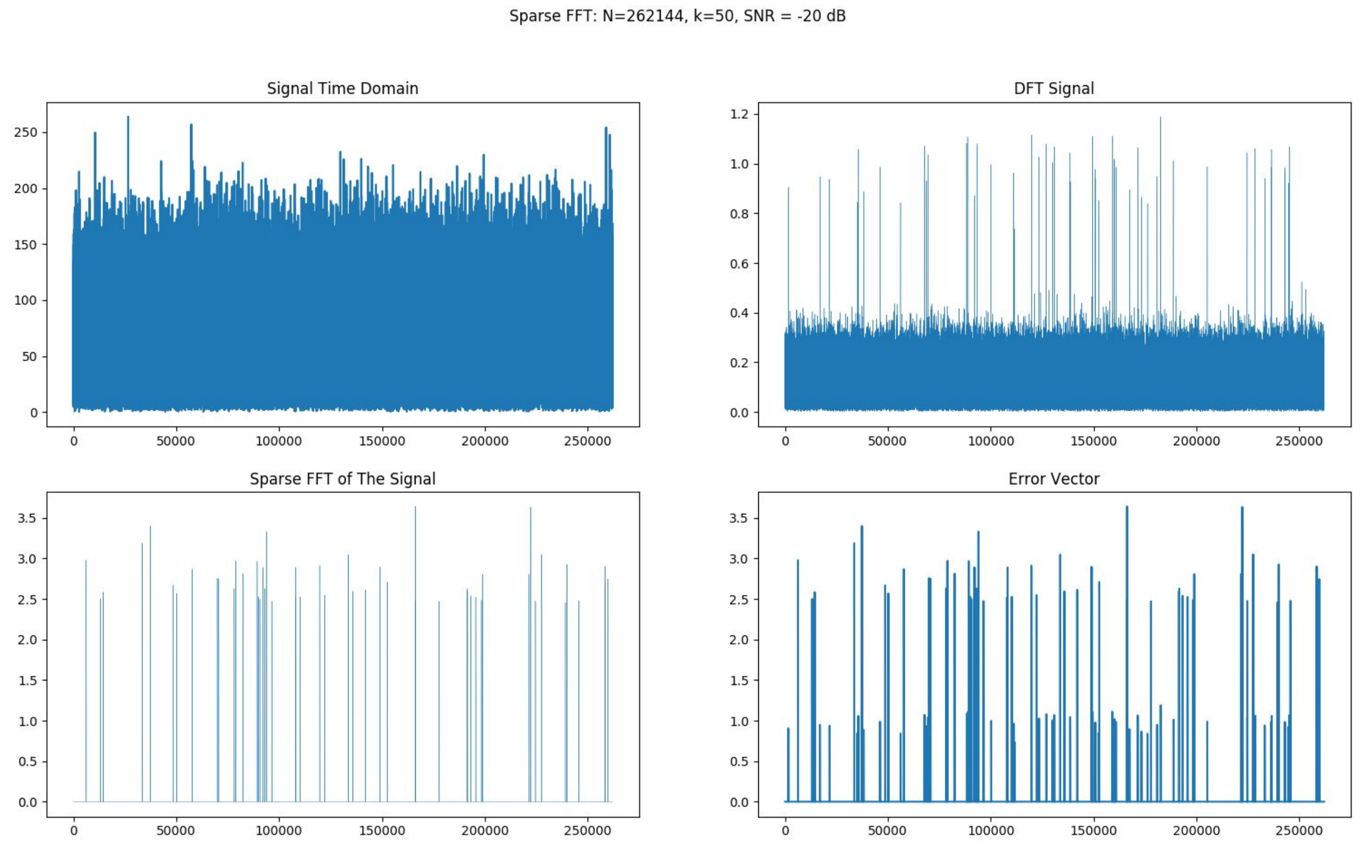
Sparse FFT error for different K



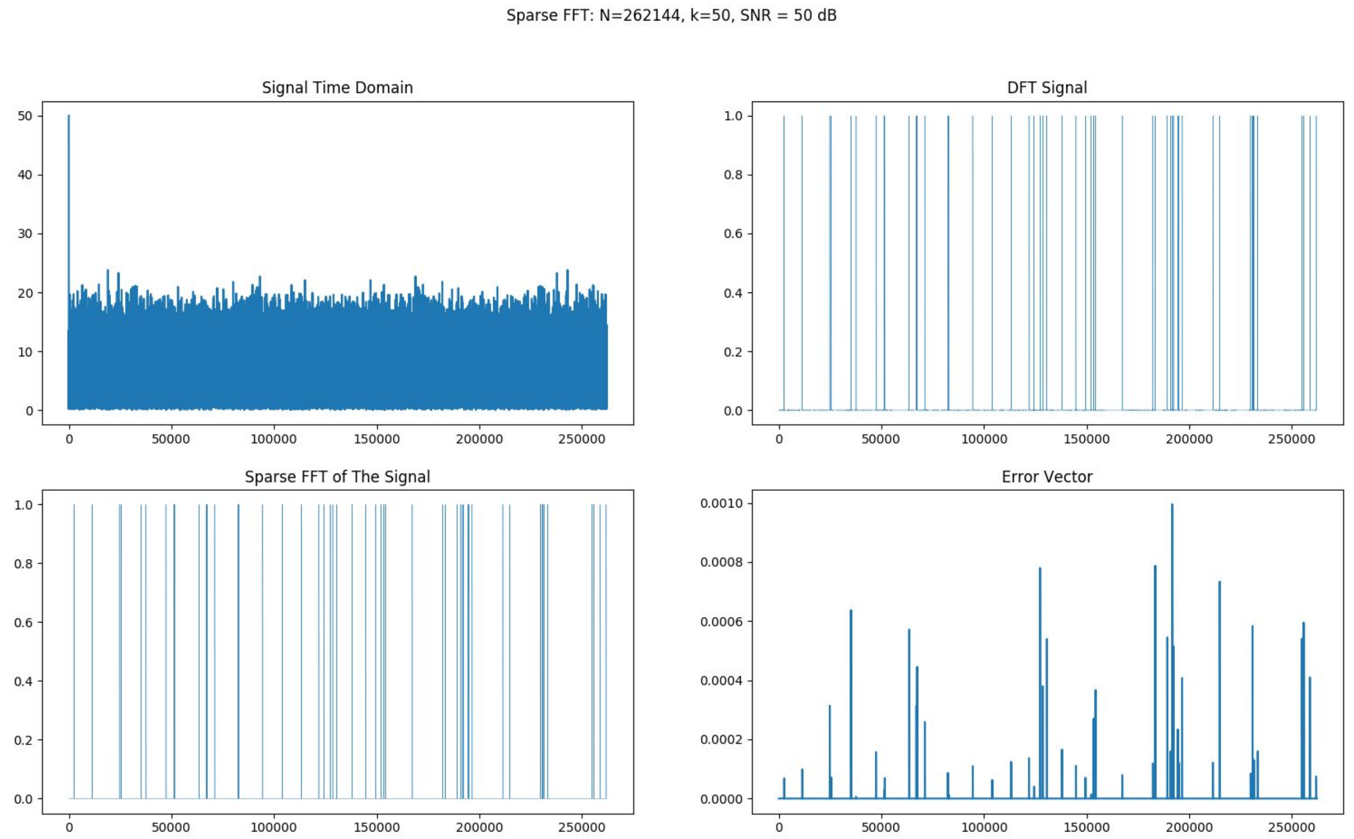
Sparse FFT error for different SNR



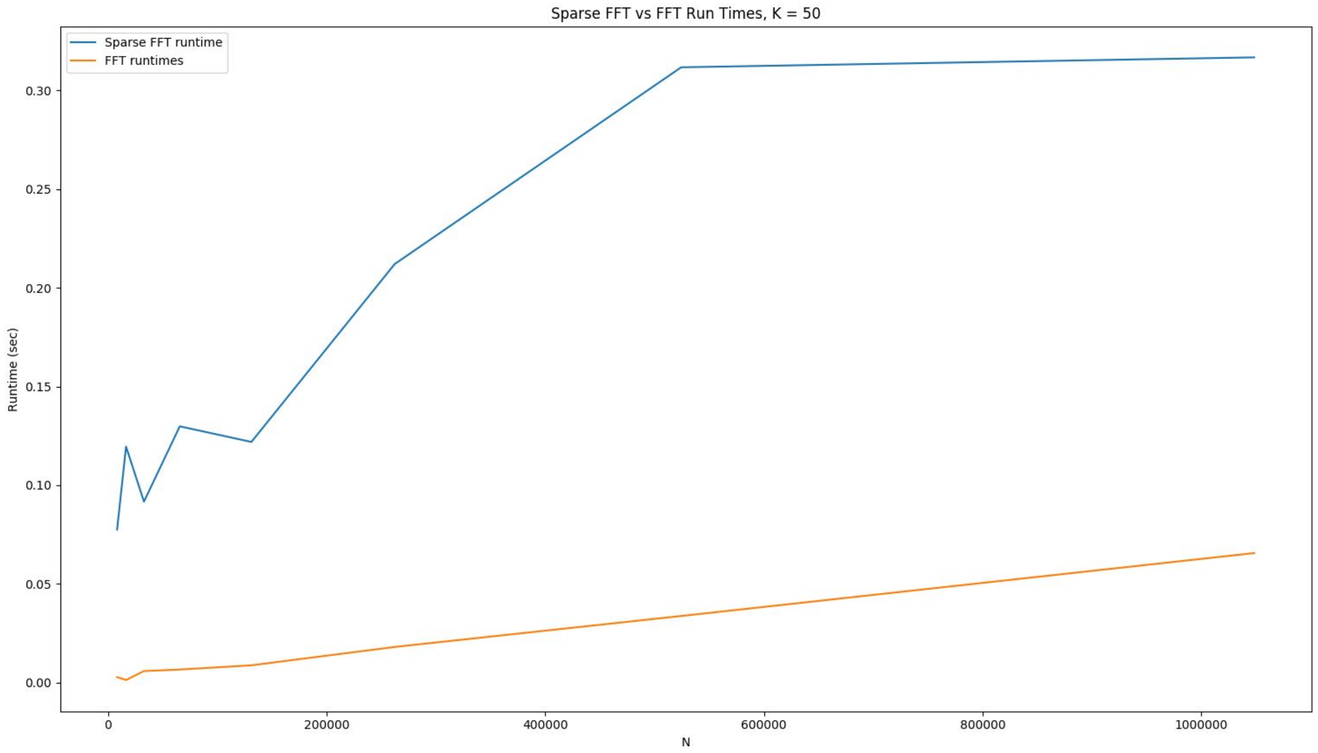
Sparse FFT error noisy signal



Sparse FFT error little noise



Sparse FFT python runtime



References

- Hassanieh, Haitham, et al. "Simple and practical algorithm for sparse Fourier transform." *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2012.
- Gilbert, Anna C., et al. "Recent developments in the sparse fourier transform: a compressed fourier transform for big data." *IEEE Signal Processing Magazine* 31.5 (2014): 91-100.