Fast Fourier Sampling: A Tutorial

A. C. Gilbert, M. J. Strauss, and J. A. Tropp

I. When Fast Is Not Enough

Suppose that $\boldsymbol{x}$ is a discrete-time signal of length N that can be expressed with only m digital frequencies where $m \ll N$:

$$x[t] = \frac{1}{\sqrt{N}} \sum_{k=1}^{m} a_k \mathrm{e}^{2\pi \mathrm{i} \omega_k t/N}, \qquad t = 0, 1, 2, \ldots, N-1.$$

We study the problem of identifying the unknown frequencies $\omega_1, \ldots, \omega_m$ that participate and their coefficients $a_1, \ldots, a_m$. Conceptually, the easiest way is to perform an N-point Discrete Fourier Transform (DFT):

$$X[\omega] = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} x[t] \mathrm{e}^{-2\pi \mathrm{i} \omega t/N}, \qquad \omega = 0, 1, 2, \ldots, N-1.$$

Having obtained all N Fourier coefficients, it is straightforward to locate the m nonzero frequencies and their coefficients. Although you can compute the DFT efficiently by means of the Fast Fourier Transform (FFT), the fact remains that you must compute a very large number of zero coefficients when the signal involves few frequencies. This approach seems rather inefficient.

The Discrete Uncertainty Principle [DS89] suggests that it might be possible to use fewer samples from the signal. Indeed, if the spectrum of a length-N discrete-time signal contains only m nonzero frequencies, then the time domain has at least N/m nonzero positions. As a result, even if we sample the signal at relatively few points in time, the samples should carry significant information about the spectrum of the signal.

This article describes a computational method, called the *Fourier Sampling algorithm*, that exploits this insight [GMS05]. The algorithm takes a small number of (correlated) random samples from a signal and processes them efficiently to produce an approximation of the discrete Fourier transform of the signal. The algorithm offers provable guarantees on the number of samples, the

running time, and the amount of storage. As we will see, these requirements are exponentially better than the FFT for some cases of interest.

This article describes in detail how to implement a version of Fourier Sampling; it presents some evidence of its empirical performance; and it explains the theoretical ideas that underlie the analysis. Our hope is that this tutorial will allow engineers to apply Fourier Sampling to their own problems. We also hope that it will stimulate further research on practical implementations and extensions of the algorithm.

A. The Fourier Sampling Algorithm, a summary

We begin with a discussion of performance guarantees, so it clear what the Fourier Sampling algorithm can accomplish and what it cannot. The algorithm requires random access to the time domain of a signal $\boldsymbol{x}$ of length N. The input parameter m is the number of frequencies sought. As output, the algorithm produces a signal $\boldsymbol{y}$ that approximates $\boldsymbol{x}$ with only m frequencies:

$$y[t] = \frac{1}{\sqrt{N}} \sum\nolimits_{k=1}^{m} a_k \mathrm{e}^{2\pi \mathrm{i} \omega_k t/N}. \tag{I.1}$$

This approximation is represented by the set $\{(\omega_k, a_k) : k = 1, 2, \ldots, m\}$ of frequency/coefficient pairs. In a moment, we will see that the approximation error is comparable with the minimal error possible using an signal of the form (I.1).

The algorithm also involves several design parameters. The number $\varepsilon > 0$ determines the quality of the computed approximation in comparison with the best approximation. The number $\delta > 0$ is the probability that the algorithm fails with respect to the random choices it makes during its execution. Both these quantities can be controlled by taking additional samples from the signal. The following theorem shows how all the factors interact.

Theorem 1 (Gilbert et al. [GMS05]): Let $\boldsymbol{x}$ be an arbitrary signal of length N. The Fourier Sampling algorithm takes $m \operatorname{poly}(\varepsilon^{-1}, \log(\delta^{-1}), \log(N))$ random samples[1] of the signal. With probability at least $1 - \delta$, the algorithm returns an approximation $\boldsymbol{y}$ of the form (I.1) that satisfies the error bound

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq (1 + \varepsilon) \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{opt}}\|_2 + \varepsilon$$

[1]The term $\operatorname{poly}(\cdot)$ indicates an unspecified polynomial in its arguments.

where $\boldsymbol{x}_{\mathrm{opt}}$ is the best approximation to $\boldsymbol{x}$ of the form (I.1). It produces this approximation using time and storage $m\,\mathrm{poly}(\varepsilon^{-1}, \log(\delta^{-1}), \log(N))$.

Let us elaborate on the statement of this theorem. When $m \ll N$, the algorithm takes far fewer samples than the total length of the signal. We emphasize that the sample set depends on random choices, but it does not depend on the signal or the progress of the algorithm. Therefore, the sample locations can be established before execution. Moreover, the runtime and storage requirements of the algorithm are roughly proportional to the number of frequencies it outputs, rather than the signal length. All the resource requirements are logarithmic in N, the signal length, so Fourier Sampling has the potential to be exponentially faster than the FFT.

Second, let us discuss how to interpret the approximation guarantee. When the signal is well approximated by a set of m frequencies, the right-hand side of the error bound is small, so the algorithm produces an approximation that is competitive with the optimal m-frequency approximation. In contrast, when it takes many frequencies to approximate the signal, the algorithm will return a poor result. In this setting, Fourier Sampling is not an appropriate tool.

Even if the true signal consists of m frequencies contaminated with heavy noise, the algorithm may not return the m ideal frequencies. Indeed, the theorem only promises that the error in the output approximation is comparable to the amount of noise. Nevertheless, a careful analysis shows that the energy of the noise must be substantial compared with the signal energy before the algorithm delivers frequencies different from the ground truth.

A more familiar way to analyze the quality of the approximation $\boldsymbol{y}$ is to compute its reconstruction SNR. Suppose that our signal $\boldsymbol{x}$ consists of m frequencies plus an orthogonal noise vector $\boldsymbol{\nu}$. Then

$$\mathrm{SNR} = 10 \log_{10}\left(\frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x} - \boldsymbol{y}\|_2}\right) \geq 10 \log_{10}\left(\frac{\|\boldsymbol{x}\|_2}{(1+\epsilon)\,\|\boldsymbol{\nu}\|_2}\right).$$

Consequently, the SNR of the reconstructed signal is smaller than optimal by an additive term. We can reduce this loss by decreasing the design parameter ϵ, although this revision results in additional samples of the signal and increased computation time.

B. Rising to the Challenge

The fundamental challenge for the Fourier Sampling algorithm is to divine information about the frequency spectrum of a signal under severe constraints on the number of samples and arithmetic operations. To do so, the algorithm makes random choices to avoid worst-case scenarios. This means that the procedure has access to random bits separate from and in addition to its input. In its execution, the algorithm uses those random bits to guide its behavior. *For each input*, it succeeds with high probability with respect to the source of randomness. This idea is substantially different from the use of statistical signal models, so practitioners of signal processing may be less familiar with it. Here are the key observations:

1) Random time samples of a signal allows us to estimate certain characteristics, such as its zero-frequency coefficient and its energy.
2) Random permutation of the spectrum allows us to separate significant tones into different frequency bands. The tones can then be isolated with bandpass filters.

The algorithm also exploits many standard methods from the DSP toolbox:

1) Sampling in time corresponds to summing modulated Fourier coefficients.
2) Dilation in time corresponds to dilation of the spectrum.
3) Modulation in time corresponds to translation of the spectrum.
4) The FFT can be used to apply a filterbank, which multiplies the spectrum of the signal by a collection of transfer functions.
5) (Nonuniform) FFTs allow fast evaluation of exponential polynomials at multiple points.

The algorithm combines these ideas in a way that is complicated and—perhaps—unusual. In Section IV, we provide more detailed information about how these methods allow us to approximate the spectrum of an unknown signal.

C. Background and Related Work

The Fourier sampling algorithm differs from traditional spectral estimation techniques in a variety of ways. First, unlike Prony's method [dP95] and its more stable variations [SH00], the algorithm is not predicated upon evenly spaced samples—just the opposite. Second, the reconstruction algorithm uses the samples in a nonlinear fashion, unlike the procedures of [HB98]. It does not form a linear combination of the sample values. Third, the algorithm and its analysis

are inherently discrete. The samples are drawn from a discrete-time signal (rather than an underlying continous-time signal) and the output of the algorithm is an approximation to the discrete spectrum.

The Fourier sampling algorithm is related to the Compressive Sampling paradigm, but the two approaches focus on different resource constraints. Let us consider the case where signals of interest have few significant frequencies in comparison with their length. The primary concern of Compressive Sampling is to reconstruct the spectrum of the signal from as few samples as possible with extremely strong guarantees on the probability of success. Researchers have established that several different randomized sampling schemes are compatible with this goal [Don06], [CT06]. Most of the literature concentrates on reconstruction algorithms such as convex programming, but other methods are available [CM06], [GSTV07]. The Fourier sampling algorithm is closest in spirt to the algorithms in [KM91], [Man95].

II. Empirical Performance

The Fourier sampling algorithm has been implemented and tested in a variety of settings to assess its empirical performance. We discuss one particular implementation [IGS07], the AAFFT, and we provide evidence that it is both powerful and resource-efficient.

First, we consider a problem inspired by communication devices that use frequency-hopping modulation schemes. Suppose we wish to recover a synthetic signal consisting of two tones that change at regular intervals. These signals are contaminated with white Gaussian noise so the SNR is -17 dB. We apply the AAFFT implementation to identify the location of the two tones. Figure 1 exhibits the output using a *sparsogram*, which is a time–frequency plot that displays only the dominant frequencies in the signal. As a benchmark, we also computed the sparsogram with FFTW, a highly optimized implementation of the FFT. Both AAFFT and FFTW obtain the correct result in the same amount of time, but AAFFT samples only 3% of the signal—a factor $33\times$ *undersampling*.

This first experiment provides evidence that AAFFT uses far fewer samples than FFTW. The AAFFT implementation is also substantially faster than FFTW for long signals. To prove this point, we constructed (noiseless) signals of different lengths by selecting 60 frequencies at random and assigning them unit coefficients. We compared the running time for AAFFT to identify these

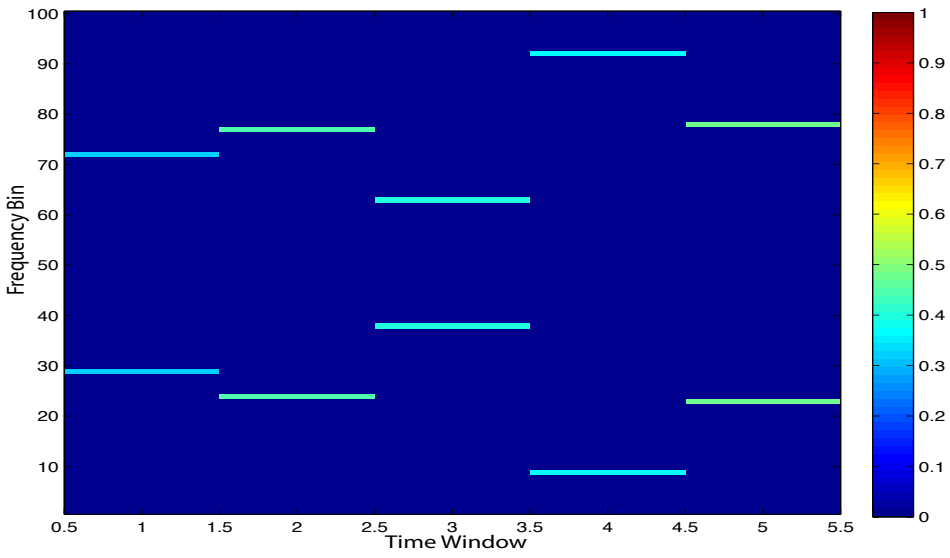

Fig. 1. The sparsogram for a synthetic frequency-hopping signal consisting of two tones in noise, as computed by the AAFFT.

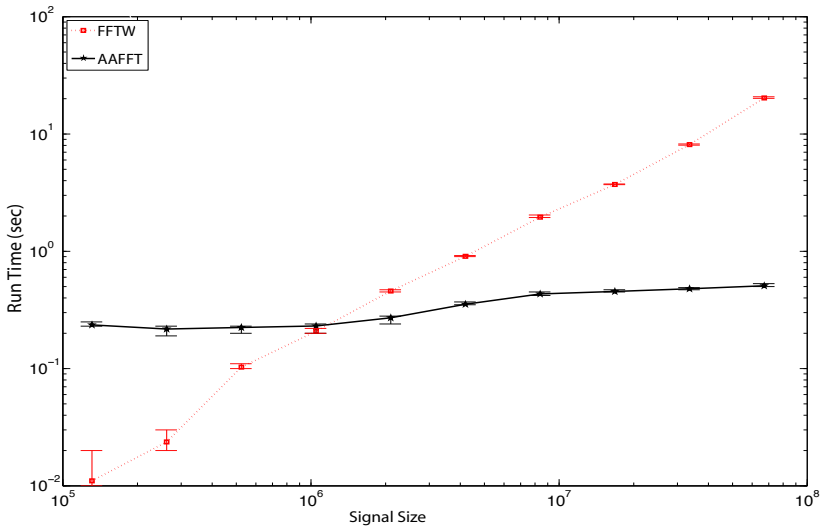

Fig. 2. The execution time of FFTW and AAFFT for recovering 60 frequencies without noise. The error bars indicate the minimum and maximum run times. The AAFFT runs faster than FFTW when the signal length exceeds 10^6.

60 frequencies with the running time for FFTW using a log–log scale. The result appears in Figure 2. Notice that the execution time of FFTW grows dramatically while the speed of the AAFFT remains virtually constant as the signal length varies across several orders of magnitude.

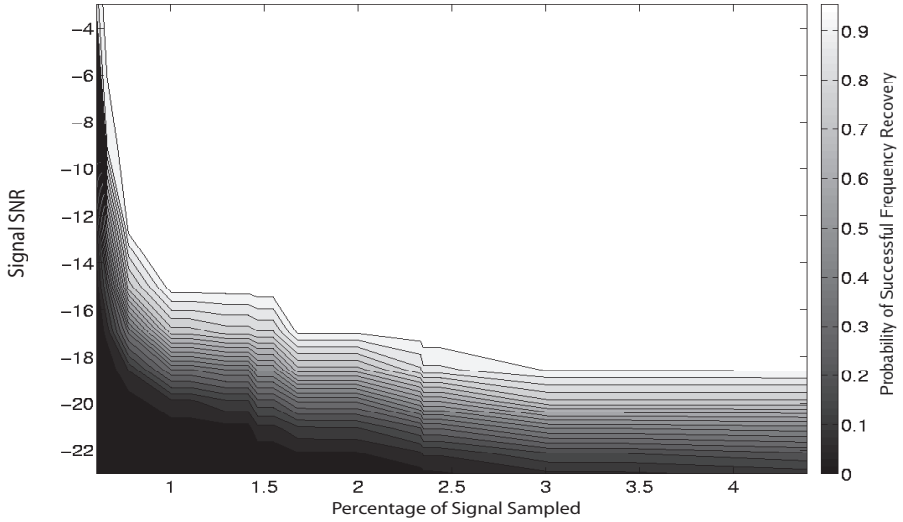

Fig. 3. Phase transition diagram for recovering one frequency in noise. The plot uses shades of gray to indicate the probability of successful recovery as a function of the SNR and the percentage of the signal that is sampled. The undersampling rate is the reciprocal of the sampling percentage.

At the beginning of the tutorial, we mentioned that the algorithm may fail completely to approximate the signal. The failure probability can be controlled by increasing the number of samples of the signal that we take. We constructed signals of length $N = 2^{22}$ (about four million) containing one tone in additive white Gaussian noise, and we attempted to locate the frequency with AAFFT. For each sampling rate, we performed 1000 trials and computed the fraction of those trials in which the tone was successfully identified. Figure 3 is a phase transition chart that indicates the probability of recovering a single frequency in heavy noise. We see, for example, that AAFFT can recover the tone 90% of the time at an SNR of -15 dB with $100\times$ undersampling. This rate is fully two orders of magnitude below Nyquist.

We also studied the number of samples necessary to recover a larger number of frequencies. We fixed the signal length at $N = 2^{22}$ and measured the number of samples necessary to recover m tones at least 99% of the time. Figure 4 displays the results. For example, if we sample 10% of the signal, the AAFFT implementation can recover 1000 tones over 99% of the time. If an application can tolerate a higher failure probability, then AAFFT can recover more tones with the same level of undersampling.

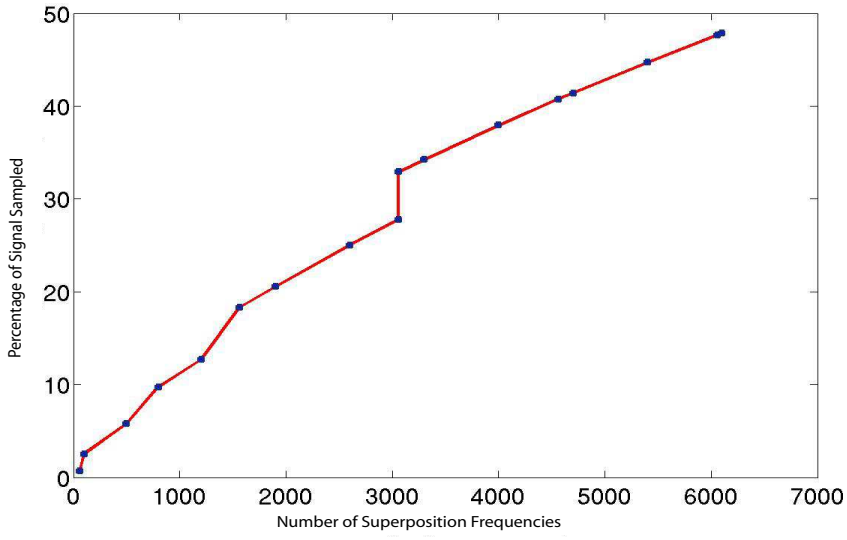

Fig. 4. The proportion of the signal that AAFFT samples to recover a fixed number of tones in a signal of length $N = 2^{22}$ at least 99% of the time.

III. Implementation

This section gives an overview of a simplified version of the AAFFT implementation, including explicit pseudocode (Algorithm III.1). This version assumes that N is a power of two, and it removes some failure-control mechanisms. The complete AAFFT algorithm is somewhat more complicated than the code here, but this basic implementation still works quite well.

Let us note that the upcoming description of the algorithm interleaves the sampling of the signal with the other actions of the algorithm. We have elected this description to make it clear precisely where samples are required. Nevertheless, we emphasize that the samples used by the algorithm are totally nonadaptive. In particular, it is possible to select the sample points and draw the samples from the signal prior to runtime.

The algorithm iteratively constructs an approximation $\boldsymbol{y}$ to the input signal. As it runs, the algorithm represents the approximation as a list Λ of at most K frequency/coefficient pairs: $\Lambda = \{(\omega_k, a_k) : k = 1, 2, \ldots, K\}$. The approximation $\boldsymbol{y}$ is implicitly determined via (I.1). The approximation also induces a *residual signal* $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{y}$. The most critical parameter in the algorithm is the size K of frequency list. In the pseudocode, we have chosen $K = 8m$. Increasing the factor eight improves accuracy at the cost of additional samples and arithmetic.

The algorithm often needs to determine the value of the residual signal at designated sampling locations. Since the approximation has the form (I.1), the SAMPLE-RESIDUAL subroutine is able to perform this computation efficiently with a nonuniform FFT. The literature describes several approaches to computing nonuniform FFTs, including [Bey95], [AD96]. Explicit pseudocode is also available [DR93, Alg. 1]. Alternatively, the exponential sums can be evaluated directly at somewhat higher cost.

At the highest level, the algorithm proceeds as follows. First, the approximation is set to zero, so the residual signal equals the input signal. The algorithm iteratively refines the approximation, as described in the next paragraph. After the iteration is complete, the algorithm reduces the list Λ by picking m frequencies with the largest coefficients.

The main loop consists of the three steps. First, the identification stage constructs a list Ω containing K frequencies that are likely to carry a significant amount of the energy from the residual. Second, the estimation stage finds estimates of the coefficients of the frequencies in Ω. Third, the algorithm adds the new approximation to the previous approximation to get a total of $2K$ terms (or fewer). Frequencies with small coefficients are likely to be spurious, so the list is reduced to K terms (or fewer) by retaining only the frequencies with the largest coefficients. Our experience suggests that the main loop should be repeated about five times.

The IDENTIFICATION subroutine employs a randomized filtering process to find up to K significant frequencies from the residual signal. Beginning with the least-significant bit, it determines each bit from all K frequencies in parallel. In the inner loop, the subroutine performs several repetitions to drive down the failure probability. Our experience suggests that 3–5 repetitions are adequate.

The ESTIMATION subroutine uses a related randomized filtering process to estimate simultaneously the coefficients of K given frequencies in the residual signal. This calculation involves the adjoint of the nonuniform FFT. Explicit pseudocode appears in [DR93, Alg. 2]. The sums can also be evaluated directly at higher cost. The subroutine takes the median of several copies of the estimator to improve robustness. In practice, 3–5 copies suffice.

IV. THE CONCEPTS BEHIND THE CODE

The Fourier sampling algorithm must perform computations on the frequency spectrum of a signal under severe constraints on the number of samples and arithmetic operations. It is possible

to achieve an economy of scale by attempting to find all the significant frequencies at once. The central design principle in the algorithm is to exploit this economy whenever possible by means of filterbanks, nonuniform FFTs, and random sampling. This section describes the intuitions behind the key steps in the algorithm. In the sequel, we assume that the signal length N is a power of two, that the signal takes complex values, and that all arithmetic on the indices is performed modulo N.

A. The Role of Randomness

In contrast with the field of statistical signal processing, we do not make any assumptions about the input signal. Instead, the algorithm makes random choices during its execution to enable it to succeed with high probability for any given input signal. In this section, we attempt to share the flavor of these techniques.

1) Random Sampling: Random sampling is a very efficient method for estimating some key characteristics of a signal. Let $\boldsymbol{x}$ be a signal of length N, and let T be the random variable that takes each value from $\{0, 1, 2, \dots, N-1\}$ with equal probability.

First, the squared magnitude of a random time sample gives a good estimate for the signal energy because $\|\boldsymbol{x}\|^2 = N \,\mathbb{E}\, |x[T]|^2$. Owing to Markov's inequality, it is unlikely that a random sample, suitably normalized, has magnitude much greater than the norm of the signal.

Second, consider a signal containing one large frequency plus noise: $x[t] = a\, \mathrm{e}^{2\pi \mathrm{i} \omega t/N}/\sqrt{N} + v[t]$. A short argument involving the triangle inequality and Jensen's inequality yields

$$|a| - \|\boldsymbol{v}\|_2 \leq \sqrt{N}\, \mathbb{E}\, |x[T]| \,.$$

That is, we can approximate the magnitude of the tone by random sampling. Therefore, we can find the location of a tone that dominates a signal.

Finally, the scaled expectation of a random sample equals the zero-frequency component of a signal. This point follows from the simple fact $X[0] = \sqrt{N}\, \mathbb{E}\, x[T]$. The algorithm uses this fact to estimate the coefficient of a specified frequency.

2) Random Spectral Permutation: A major difficulty is that significant tones in the spectrum of a signal can be clustered together or spread out. One of the central innovations in the algorithm is a randomized technique for isolating significant tones from each other so we can perform

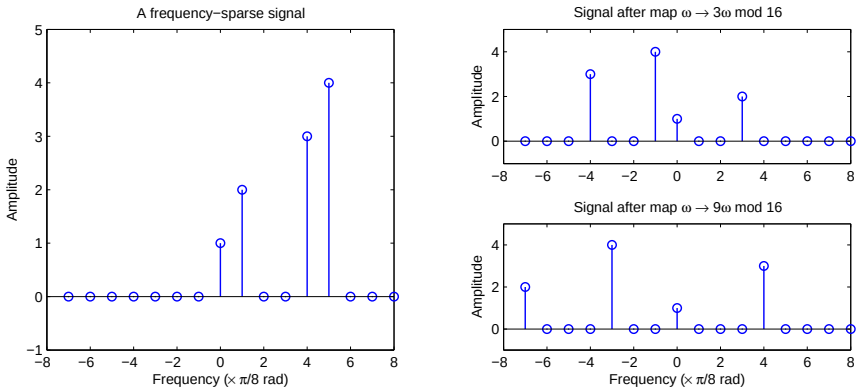

Fig. 5. The impact of dilations on the spectrum of a frequency-sparse signal. The top-right panel is generated by the time dilation $t \mapsto 11t \bmod N$ (equivalently, by a frequency dilation $\omega \mapsto 3\omega \bmod N$); the bottom-right panel by $t \mapsto 9t \bmod N$ (or $\omega \mapsto 9\omega \bmod N$). Note that the zero frequency always remains fixed.

spectral analysis using bandpass filters.

To explain, we need some basic number theory. Two numbers are *relatively prime* if they have no common integer factor except ± 1. Since N is a power of two, the numbers relatively prime to N are precisely the odd integers. Given an odd number σ, the Euclidean division algorithm furnishes a number σ^{-1}, called its *multiplicative inverse*, that satisfies $\sigma \cdot \sigma^{-1} \equiv 1 \pmod N$.

Let σ be odd, and consider the dilation $d_\sigma : t \mapsto \sigma t \bmod N$. It is not hard to see that this map is a permutation on the set $\{0, 1, 2, \dots, N-1\}$ and that its inverse is the map $d_{\sigma^{-1}}$. In discrete Fourier analysis, these observations lead directly to the identity

$$y[t] = x[\sigma t] \quad \text{for all } t \quad \Longleftrightarrow \quad Y[\omega] = X[\sigma^{-1}\omega] \quad \text{for all } \omega.$$

Succinctly, time dilations generate frequency permutations. See Figure 5 for an illustration.

The key idea is to choose σ at random from the set $\{1, 3, 5, \dots, N-1\}$ of invertible numbers. Applying the dilation d_σ to the signal, we produce a random permutation of its spectrum. It is unlikely that a given pair of frequencies is mapped to the same part of the spectrum. Roughly speaking, random permutation of the spectrum isolates significant frequencies from each other.

B. Identification

The first stage in the Fourier sampling algorithm is to identify a collection of frequencies whose coefficients are large relative to the signal energy. The identification process consists of two conceptual steps, *shattering* and *bit testing*. Shattering generates a collection of signals, many

of which have a single dominant frequency. Then bit tests are applied to each signal to find the location of the dominant frequency, one bit at a time.

1) Shattering: A *shattering* of a signal $\boldsymbol{x}$ is a collection $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{K-1}\}$ of signals that are formed by a three-step filtering process. First, we randomly permute the spectrum of the signal to isolate significant frequencies from each other. Second, we apply a sub-band decomposition filterbank to create K signals that each carry a chunk of the permuted spectrum. Each significant tone in the orginal signal is likely to dominate one signal in the shattering. Finally, we invert the dilation to restore the frequencies to their original places in the spectrum. Figure 6 exhibits a block diagram, and Figure 7 illustrates the effects of shattering on a signal.

The design of the sub-band decomposition filterbank is simple. Let $\boldsymbol{h}$ be a low-pass filter with K taps whose cutoff frequency is about π/K radians. The filterbank consists of K frequency translates of this filter, spaced $2\pi/K$ radians apart. In the time domain, this amounts to convolution with $h_k[t] = \mathrm{e}^{-2\pi \mathrm{i} kt/K} h[t]$ for each $k = 0, 1, \dots, K-1$. The analysis in [GMS05] suggests that the ideal filter has minimal energy among all normalized filters with K taps. This observation recommends the boxcar filter $h[j] = \sqrt{N}/K$ for $j = 0, 1, \dots, K-1$. It is possible that more sophisticated low-pass filters or windows will sometimes yield better results [OSB99, Ch. 7].

Let us emphasize that the algorithm never forms a shattering explicitly. Instead, the filterbank is constructed so we can take one time sample from each element of the shattering by processing K samples from the input signal. If σ is the parameter of the random dilation, the kth signal in the shattering satisfies

$$x_k[t] = \sum\nolimits_{j=0}^{K-1} h[j]\, x[t - \sigma j]\, \mathrm{e}^{-2\pi \mathrm{i} jk/K}.$$

Given a point t, we can simultaneously calculate $x_0[t], x_1[t], \dots, x_{K-1}[t]$ by extracting an arithmetic progression from the input signal, multiplying it with the filter, and applying an FFT. The subroutine SAMPLE-SHATTERING performs these actions.

2) Bit Testing: Shattering generates a collection of signals, some of which contain a single dominant frequency. The bit-test process is designed to locate the dominant frequency in such a signal. (The bit tests are likely to return spurious frequencies for other elements of the shattering.)

Suppose that $\boldsymbol{x}$ is a length-N signal in which a single frequency carries most of the energy. We find the bits of the frequency sequentially, beginning with the least significant bit $b = 0$. Assuming we already know the least-significant $(b-1)$ bits, we can demodulate the signal so

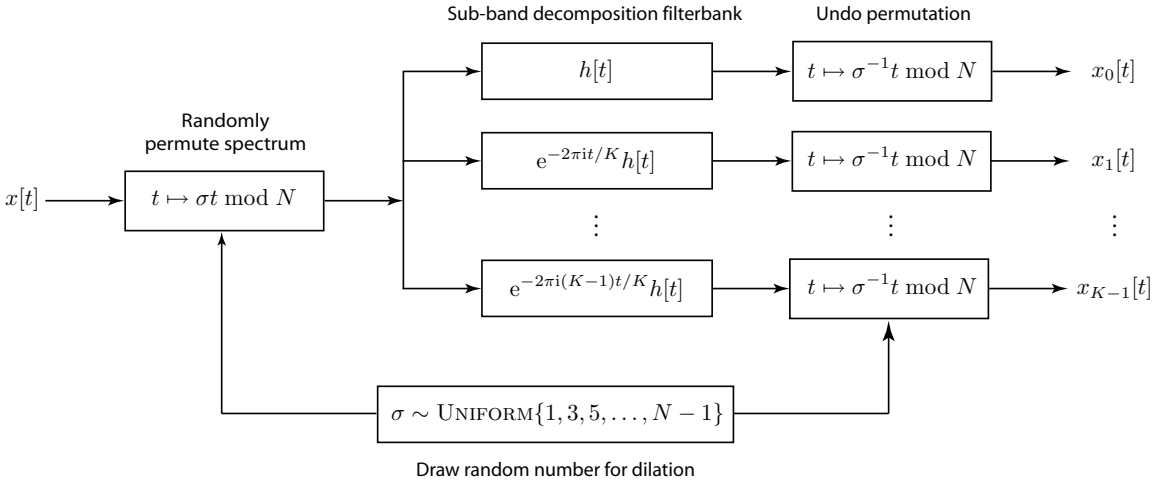

Fig. 6. A conceptual block diagram for the shattering process. A shattering of $\boldsymbol{x}$ contains K elements $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{K-1}$.

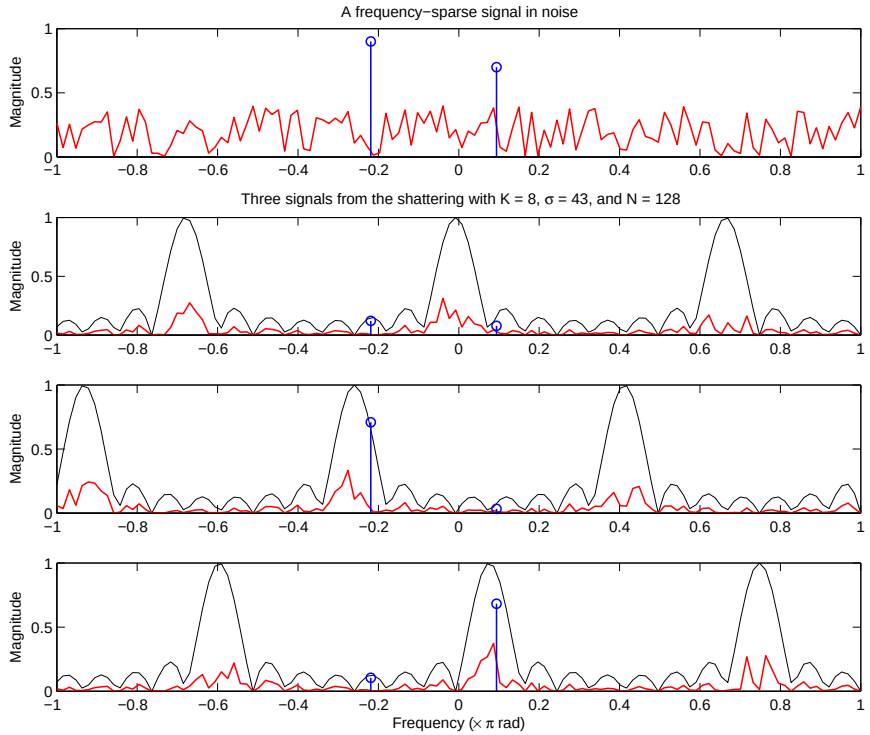

Fig. 7. A signal consisting of two tones in noise, along with three elements from a shattering. In the second panel, both the tones are attenuated so neither is recoverable. The third and fourth panels show elements where one tone is preserved and the other is attenuated. In each plot, the transfer function is traced in black.

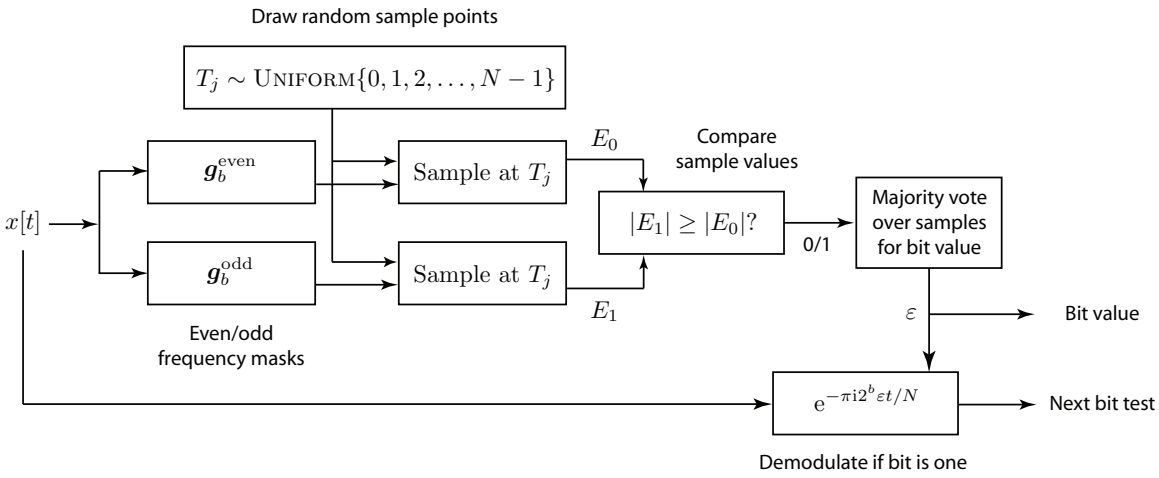

Fig. 8. A conceptual block diagram for the bth bit test. The test yields a 0/1 bit value ε and a demodulated copy of the input signal for the next bit test.

that the binary expansion of the dominant frequency ends in $10\ldots0$ or in $00\ldots0$. We apply the frequency mask filters

$$\boldsymbol{g}_b^{\text{even}} = \frac{1}{2}\left(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_{N/2^{b+1}}\right) \quad \text{and} \quad \boldsymbol{g}_b^{\text{odd}} = \frac{1}{2}\left(\boldsymbol{\delta}_0 - \boldsymbol{\delta}_{N/2^{b+1}}\right).$$

The filter $\boldsymbol{g}_b^{\text{even}}$ passes the even frequencies and zeros the odd frequencies (mod 2^b). In a similar fashion, the filter $\boldsymbol{g}_b^{\text{odd}}$ passes the odd frequencies and zeros the even ones (mod 2^b). If E_0 is a random sample from the output of the even filter and E_1 is a random sample from the odd filter, then the inequality $|E_1| \geq |E_0|$ is evidence that the least-significant bit is one.

Therefore, the bit test compares the magnitude of a random sample from each of the two filtered signals. It repeats the comparison several times, and it takes a majority vote to reduce the failure probability. See Figure 8 for a block diagram of the bth bit test. Note that bit testing is computationally efficient since each filter has only two taps.

3) Implementation: We separate the *concepts* of shattering and bit testing, but the code must intertwine them for efficiency. Recall that we can simultaneously compute one sample from each of the K elements in the shattering. To exploit this fact, we simultaneously test the bth bit of the dominant frequency in each element of the shattering using two correlated samples, demodulated by the first $(b-1)$ bits of that frequency. Details appear in the IDENTIFICATION subroutine.

C. Estimation of Fourier Coefficients

The identification phase of the algorithm returns a list of K or fewer frequencies, but it does not provide sufficient information about their coefficients. The next stage of the algorithm estimates the coefficients using a randomized filtering technique.

Suppose we want to estimate the coefficient of a significant frequency ω in a signal $\boldsymbol{x}$. First, we demodulate the signal by ω so we can estimate the zero frequency instead. Next, we randomly dilate the signal. This operation fixes the zero frequency and shuffles the other significant frequencies around (probably away from zero). Third, we apply a filter to pass the zero frequency and attenuate the rest of the spectrum. Then, we invert the dilation. At this point, the zero frequency is likely to dominate the signal. We estimate its coefficient by taking a random sample and scaling appropriately. See Figure 9 for a block diagram.

As in shattering, we cannot afford to use a filter with more than K taps. It turns out that the boxcar filter $h[k] = \sqrt{N}/K$ for $k = 0, 1, 2, \ldots, K-1$ remains a good choice in this setting.

We can write down the cumulative effect of this random filtering process. Let σ be the parameter of the random dilation, and let t be the sample location. Then the coefficient estimate c_ω is

$$c_\omega = \sqrt{N} \mathrm{e}^{-2\pi \mathrm{i} \omega t/N} \sum_{j=0}^{K-1} h[j]\, s[t - \sigma j]\, \mathrm{e}^{-2\pi \mathrm{i} (\omega \sigma K/N)\, j/K}.$$

One should view this expression as the discrete Fourier transform of a sequence of length K, evaluated at the nonintegral frequency $2\pi\omega\sigma K/N$ radians, then demodulated and scaled. Therefore, we can simultaneously estimate the coefficients of a collection of K frequencies (or fewer) using the adjoint nonuniform FFT. Afterward, we can demodulate each coefficient individually. Finally, we make each coefficient estimator more robust by taking the median of several copies. (The medians of the real and imaginary parts are performed separately.)

D. Iteration

The *recoverable energy* in a signal is the energy carried by the largest m frequencies. It is impossible to collect more since our approximation contains only m frequencies. By performing identification and estimation once, the algorithm finds a constant proportion of the recoverable energy in the residual. Therefore, after a constant number of iterations, the algorithm can find a fixed proportion of the recoverable energy in the original signal.

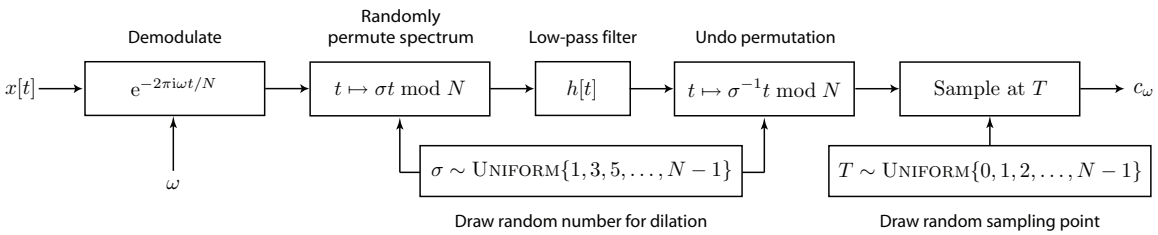

Fig. 9. A conceptual block diagram for the coefficient estimator. The system returns an estimate for the coefficient c_ω of the frequency ω in the signal $\boldsymbol{x}$. Note that the estimate must also be scaled by $\sqrt{N}$.

V. Extensions and Improvements

Although the Fourier sampling algorithm is designed for discrete-time signals, we can use it in certain analog settings with some modifications. We can acquire a few random structured samples of a wide-band continuous-time signal that has a few significant tones and recover quickly those tones present. To build a practical system, we must analyze carefully the required minimum sample spacing as it is costly to acquire signal samples close in time. We must also increase the flexibility of the output representation so as to improve the reconstruction SNR; instead of returning exactly m, we return a (tunable) multiple of m. All of these modifications are possible while still preserving the structure and quality guarantees of the algorithm. We must be realistic, however, in assessing the quality of our output. The algorithm returns a compressed or approximate representation of the discrete spectrum of an inherently analog signal. It only approximates the significant portions of the discrete spectrum, which, itself, is an approximation to the true spectrum.

Acknowledgments

The authors would like to thank Mark Iwen for his implementation and empirical analysis of AAFFT. AAFFT is publicly available and can be downloaded from http://www-personal.umich.edu/~markiwen/WebPage/AAFFT.tgz. We would also like to thank Sudipto Guha, Piotr Indyk, and Muthu Muthukrishnan for their collaboration on early versions of the Fourier sampling algorithm and its analysis [GGI$^+$02]. We especially thank the editors of this special issue and the anonymous referees for helping us improve the exposition of this article.

REFERENCES

[AD96] C. Anderson and M. D. Dahleh. Rapid computation of the discrete Fourier transform. *SIAM J. Sci. Comput.*, 17:913–919, 1996.

[Bey95] G. Beylkin. On the Fast Fourier Transform of functions with singularities. *Appl. Comp. Harmonic Anal.*, 2(4):363–381, 1995.

[CM06] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Proc. 2006 IEEE Int. Conf. Information Sciences and Systems*, Princeton, Apr. 2006.

[CT06] E. J. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Info. Theory*, 52(12):5406–5425, Dec. 2006.

[Don06] D. L. Donoho. Compressed sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, Apr. 2006.

[dP95] G. R. de Prony. Essai éxperimental et analytique: sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l'alkoo, à différentes températures. *Journal de l'École Polytechnique*, 1(22):24–76, 1795.

[DR93] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[DS89] D. L. Donoho and P. B. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49(3):906–931, 1989.

[GGI+02] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. J. Strauss. Near-optimal sparse Fourier representations via sampling. In *ACM Symposium on Theoretical Computer Science*, 2002.

[GMS05] A. C. Gilbert, S. Muthukrishnan, and M. J. Strauss. Improved time bounds for near-optimal sparse Fourier representations. In M. Papdakis, A. F. Laine, and M. A. Unser, editors, *Proceedings of SPIE Wavelets XI*, pages 59141A.1–15, San Diego, August 2005. SPIE.

[GSTV07] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: Fast algorithms for Compressed Sensing. In *Proc. 39th ACM Symposium on Theory of Computing*, San Diego, Jun. 2007. To appear.

[HB98] G. Harikumar and Y. Bresler. FIR perfect signal reconstruction from multiple convolutions: minimum deconvolver orders. *IEEE Trans. Signal Processing*, 1998.

[IGS07] M. Iwen, A. C. Gilbert, and M. J. Strauss. Empirical evaluation of a sub-linear time sparse DFT algorithm. *Communications in Mathematical Sciences*, 5(4), 2007.

[KM91] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proc. of the ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[Man95] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Scientific Computing*, 24(2):357–368, 1995.

[OSB99] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice-Hall, 1999.

[SH00] G. K. Smyth and D. M. Hawkins. Robust frequency estimation using elemental sets. *Journal of Computational and Graphics Statistics*, (9):196–214, 2000.

ALGORITHM III.1: Simplified Fourier sampling Algorithm.

FOURIERSAMPLING($\boldsymbol{x}$, m)
Input: Input signal $\boldsymbol{x}$ of length $N = 2^\alpha$ and number m of frequencies to find
Output: Set $\Lambda = \{(\omega, a_\omega)\}$ containing $\mathrm{O}(m)$ frequency/coefficient pairs

$K \leftarrow 8m$ and $\Lambda \leftarrow \emptyset$
for $j = 1$ **to** 5
  $\Omega \leftarrow$ IDENTIFICATION$(\boldsymbol{x}, \Lambda, K)$ { Identify K frequencies in the residual }
  $\boldsymbol{c} \leftarrow$ ESTIMATION$(\boldsymbol{x}, \Lambda, \Omega)$ { Estimate Fourier coefficients }
  foreach frequency $\omega \in \Omega$ and corresponding coefficient c_ω
    if $(\omega, a_\omega) \in \Lambda$ for some a_ω **then** replace the pair with $(\omega, a_\omega + c_\omega)$
      else add the new pair (ω, c_ω) to Λ
  Retain K pairs from Λ whose coefficients have greatest magnitude
Retain m pairs from Λ whose coefficients have greatest magnitude

---

SAMPLE-RESIDUAL($\boldsymbol{x}$, Λ, t, σ, K)
for $k = 1$ **to** K
  $u_k \leftarrow x[t + \sigma(k-1) \bmod N]$ { Arithmetic progression from signal }
  $v_k \leftarrow \sum_{(\omega, a_\omega) \in \Lambda} (a_\omega \mathrm{e}^{2\pi \mathrm{i} \omega t/N}) \, \mathrm{e}^{2\pi \mathrm{i}(\omega\sigma/N)(k-1)}$ { In parallel, via nonuniform FFT }
return $(\boldsymbol{u} - \boldsymbol{v})$ { Residual is signal minus approximation }

IDENTIFICATION($\boldsymbol{x}$, Λ, K)
reps $\leftarrow 5$ and $\omega_k \leftarrow 0$ **for** $k = 1, 2, \ldots, K$
Draw $\sigma \sim$ UNIFORM$\{1, 3, 5, \ldots, N-1\}$ { Random shattering parameter }
for $b = 0$ **to** $\log_2(N/2)$ { Loop from LSB to MSB }
  $\mathrm{vote}_k \leftarrow 0$ **for** $k = 1, 2, \ldots, K$
  for $j = 1$ **to** reps
    Draw $t \sim$ UNIFORM$\{0, 1, 2, \ldots, N-1\}$ { Random sample point }
    $\boldsymbol{u} \leftarrow$ SAMPLE-SHATTERING(t) { Samples correlated
    $\boldsymbol{v} \leftarrow$ SAMPLE-SHATTERING$(t + N/2^{b+1})$ for testing bth bit }
    for $k = 1$ **to** K
      $E_0 \leftarrow u_k + \mathrm{e}^{-\pi \mathrm{i} \omega_k / 2^b} \, v_k$ { Apply bit-test filters to
      $E_1 \leftarrow u_k - \mathrm{e}^{-\pi \mathrm{i} \omega_k / 2^b} \, v_k$ demodulated signal }
      if $|E_1| \geq |E_0|$ **then** $\mathrm{vote}_k \leftarrow \mathrm{vote}_k + 1$ { Vote when bit is one }
  for $k = 1$ **to** K
    if $\mathrm{vote}_k > \mathrm{reps}/2$ **then** $\omega_k \leftarrow \omega_k + 2^b$ { Majority vote for bit value }
return $\mathrm{Unique}(\omega_1, \omega_2, \ldots, \omega_K)$ { Remove duplicates }

SAMPLE-SHATTERING(p)
$\boldsymbol{z} \leftarrow$ SAMPLE-RESIDUAL$(\boldsymbol{x}, \Lambda, p, \sigma, K)$ { Get arithmetic progression of samples }
$\boldsymbol{z} \leftarrow \mathrm{FFT}(\boldsymbol{z})$ { Apply sub-band decomposition filterbank }
return $\boldsymbol{z}$

ESTIMATION($\boldsymbol{x}$, Λ, Ω)
reps $\leftarrow 5$
for $j = 1$ **to** reps
  Draw $\sigma \sim$ UNIFORM$\{1, 3, 5, \ldots, N-1\}$ and $t \sim$ UNIFORM$\{0, 1, 2, \ldots, N-1\}$
  $\boldsymbol{u} \leftarrow$ SAMPLE-RESIDUAL$(\boldsymbol{x}, \Lambda, t, \sigma, K)$
  for $\ell = 1$ **to** $|\Omega|$
    $c_\ell(j) \leftarrow \sum_{k=1}^{K} u_k \, \mathrm{e}^{2\pi \mathrm{i}(\omega_\ell \sigma/N)(k-1)}$ { In parallel, via nonuniform FFT }
    $c_\ell(j) \leftarrow (N/K) \, \mathrm{e}^{-2\pi \mathrm{i} \omega_\ell t/N} c_\ell(j)$ { Demodulate and scale estimates }
$c_\ell \leftarrow \mathrm{Median}\{c_\ell(j) : j = 1, 2, \ldots, \mathrm{reps}\}$ **for** $\ell = 1, 2, \ldots, |\Omega|$ { Do real, imaginary separately }
return $c_1, c_2, \ldots, c_{|\Omega|}$