The Sinefit Spectral Envelope Preprocessor
James Paris, Zachary Remscrim, Dr. Steven B. Leeb, Dr. Steven R. Shaw

Abstract—This paper presents a new spectral envelope preprocessor based on sinusoid fitting and the Discrete Fourier Transform (DFT). This preprocessor is well-suited for non-intrusive condition monitoring and diagnostics due to its high noise resiliency and flexibility. It reduces data storage, transfer, and processing requirements by extracting only relevant harmonic signatures. This paper analyzes the resolution and accuracy benefits of spectral envelopes including the effects of additive white Gaussian noise and the presence of higher frequency spectral harmonics.

I. INTRODUCTION
For many electrical systems driven by ac sources, the “spectral envelope” representation of observed current and voltage signals has proven to be a widely useful and powerful metric for classification, diagnostics, and power quality measurement [1]–[5]. Spectral envelopes describe the harmonic content of the measured signals at integer multiples of the ac line frequency driving the monitored loads. Such loads exhibit behaviors that are synchronous with the line frequency. By extracting spectral envelopes, the preprocessor facilitates physically based analysis of power and current consumption.

The spectral envelope preprocessor has two primary tasks: phase and frequency estimation, and harmonic coefficient calculation. Existing versions of the spectral envelope preprocessor vary in their implementations. For phase and frequency estimation, common techniques include phase-locked loops [4], weighted least-squares estimators, and Kalman filters [2]. To reduce computation load, existing implementations have utilized techniques such as analog multipliers and precomputed basis vectors [2], [4].

This paper presents the Sinefit spectral envelope preprocessor based on non-linear least-squares sinusoid fitting combined with the DFT. These techniques focus on accuracy and implementation flexibility, reflecting the growing availability of high performance computing resources. The preprocessor is implemented within the NilmDB framework [6], allowing reuse and replacement of computation components for related or optimized calculations. Sinefit solves problems with existing preprocessors by providing more robust phase and frequency detection, accurate timestamping of spectral envelopes, and improved and quantifiable accuracy.

II. SPECTRAL ENVELOPES
Spectral envelopes \(a_k(t)\) and \(b_k(t)\) are short-term averages of harmonic content, calculated over sliding windows of an input signal. Fig. 1 demonstrates spectral envelopes as computed for an ac load. The first plot is the raw sampled input from a data acquisition board. The second shows the in-phase and quadrature components of the first harmonic (60 Hz) envelopes.

A. Definition
For the NILM, we assume that the voltage and current signals \(v(t)\) and \(i(t)\) are locally periodic over one ac line cycle. For a discrete input \(i[n]\) sampled at rate \(f_s\), one period is of length \(N = f_s/f_0\) samples, and we can compute harmonic coefficients as:

\[
a_k = \frac{2}{N} \sum_{n=0}^{N-1} i[n] \cdot \sin(k(2\pi n/N)) \quad (1)
\]

\[
b_k = \frac{2}{N} \sum_{n=0}^{N-1} i[n] \cdot \cos(k(2\pi n/N)) \quad (2)
\]

Here, \(k\) denotes the multiple of the line frequency to which a particular coefficient corresponds; for example, \(k = 1\) corresponds to the 60 Hz component and \(k = 3\) to the 180 Hz component.

Harmonic coefficients can also be calculated in complex form using the DFT, which is defined as:

\[
X_k = \mathcal{F}(x[n]) = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi nk/N} \quad (3)
\]

Using Euler’s formula, we can extract \(a_k\) and \(b_k\) in terms of the DFT as:

\[
e^{-jw} = \cos(\omega t) - j \sin(\omega t) \quad (4)
\]

\[
a_k = -\frac{2}{N} \text{imag}(X_k) \quad (5)
\]

\[
b_k = \frac{2}{N} \text{real}(X_k) \quad (6)
\]

where, as before, \(N = f_s/f_0\).

Fig. 1: Raw ac voltage and current measurements (top) and computed spectral envelopes (bottom).
The values of these coefficients are calculated for successive or sliding windows of the input signal, and the resulting time-varying harmonics $a_k[n]$ and $b_k[n]$ are the spectral envelopes. Spectral envelopes can be extracted from any periodic or quasiperiodic input signal, and are typically computed for the current, using the voltage as the phase and frequency reference. When computed separately for voltage $v[n]$ and current $i[n]$, we denote the coefficients as $a_{vk}, b_{vk}, a_{ik},$ and $b_{ik}$. Then, first harmonic real and reactive power are:

$$P_1 = a_{v1} \cdot a_{i1} \quad Q_1 = b_{v1} \cdot b_{i1}$$  \hspace{1cm} (7)

For relatively “stiff” and harmonic-free utility voltage, $a_{v1}$ and $b_{v1}$ can be assumed to be constant, in which case

$$P_1 \propto a_{i1} \quad Q_1 \propto -b_{i1}$$  \hspace{1cm} (8)

**B. Phase Rotation**

The coefficients $a_k$ and $b_k$ in (5) and (6) are calculated with sliding or successive windows over the input data. These windows must be phase-aligned with a reference, typically the utility voltage, such that $n = 0$ corresponds to, for example, the zero crossing. Then, $a_k$ and $b_k$ will refer to the “in-phase” and “quadrature” spectral components, respectively.

In the more general case, the spectral coefficients can be computed over any window $[n, n + N]$, where the reference phase corresponding to sample $n$ is $\phi[n] = \phi_0 \neq 0$. Then, a correcting rotation of $-k\phi_0$ can be applied to the complex DFT coefficient $X_k$:

$$X'_k = X_k \cdot e^{j\phi_0 \cdot k}$$  \hspace{1cm} (9)

There are four common cases that require this phase rotation:

1) When calculating harmonic coefficients with a sliding window that is shifted by a non-integer number of periods, the start of each successive window will have a different phase $\phi_0$, which must be accounted for by a rotation of $X_k$ on a per-window basis.

2) For three-phase ac systems, it is common to use a single voltage $V_{p,A}$ as a phase reference. When computing spectral envelopes corresponding to $I_{pA}, I_{pB},$ and $I_{pC}$ using this reference, phase rotations of $0^\circ, 120^\circ,$ and $240^\circ$ should be applied, respectively.

3) Current transformers or transducers may introduce a fixed phase offset in their measurement, typically 0-1°. Correcting this offset with phase rotation of the preprocessor output can significantly reduce error at higher harmonics.

4) Multi-channel data acquisition cards that sample sequentially rather than simultaneously will introduce a similar phase offset between samples. For example, evenly-spaced, 6 channel, 8 KHz sampling of a 60 Hz signal inserts a phase rotation of 0.45° between each channel.

In subsequent discussion, the first case is referred to as $\phi_{\text{shift}}$, and the others are collectively referred to as $\phi_{\text{extra}}$.

![Fig. 2: Block diagram of signal flow in the Sinefit spectral envelope preprocessor.](image)

**III. IMPLEMENTATION**

Existing preprocessors such as the Kalman-filter spectral envelope preprocessor, described in [2], have been heavily used in non-intrusive load monitoring and diagnostics [3], [7]–[9]. However, a number of practical shortcomings of this preprocessor have been identified [6]. To address these, a new approach to spectral envelope preprocessing has been developed within the NilmDB framework, a unified system for managing and processing NILM data [6].

The new preprocessor has a modular, transparent design that allows for easy replacement and reuse of components, such as the phase alignment or harmonic coefficient calculation. It also fully supports the capabilities of NilmDB by utilizing its stream and metadata metaphors, and by incorporating accurate timestamping for all data. The preprocessor uses a phase and frequency estimation algorithm that is robust when presented with highly variable or truncated voltage waveforms. Finally, it focuses on correctness and implementation simplicity by taking advantage of the significant advancements in computing power since the existing preprocessors were developed.

Here, the spectral envelope preprocessor is split into two independent components. The first, an algorithm and code module called “nilm-sinefit”, performs least-squares fits of sinusoids to successive windows of the input waveform in order to mark zero crossings, frequency, and amplitude. The second, “nilm-prep”, performs spectral envelope extraction using (5), (6) and (9) over sliding windows of the input and sine fit data. The signal flow in the new spectral envelope preprocessor is shown in Fig. 2, and the implementations of these components are detailed below.

**A. 4-Parameter Sinusoid Fitting**

In order to accurately estimate the unknown frequency and phase angle from a voltage waveform, the preprocessor finds the best fit of a sinusoid to the data, using the method identified in IEEE Std 1241-2010 Annex B.2 [10]. Fig. 3 demonstrates this optimal fit for two representative waveforms.

Mathematically, given an arbitrary waveform vector $v$ of length $N$ sampled at frequency $f_s$, we wish to calculate the
conditions as:

The more accurate interpolated DFT described in [11], which can be used; for example, the implementation in §III-B uses within the resolution of the DFT. Other estimation techniques of \( v \) is found by calculating the DFT

such that the least-squares residual \( \sum (v[n] - v'[n])^2 \) is minimized. Note that this system is non-linear.

The algorithm that estimates these parameters is as follows. First, an initial estimate of \( f_0 \) is found by calculating the DFT of \( v \), locating the DFT index \( L \) with the largest magnitude, and computing \( f_{0,\text{est}} = L \cdot f_s / N \). This frequency is accurate to within the resolution of the DFT. Other estimation techniques can be used; for example, the implementation in §III-B uses the more accurate interpolated DFT described in [11], which may improve convergence of the sinusoid fit for small \( N \).

Now, perform the iterative fit of (10). Define the starting conditions as:

For each iteration \( i = \{0, 1, 2, \ldots, m\} \), perform a least-squares fit on a linearization of the system. Given an assumed small deviation from \( \omega_i \) of \( \Delta \omega_i \), the Taylor series expansion of (10) around the current estimated parameters gives the linear equation:

\[
\begin{bmatrix} A \ t \ \ B \ t \ \ C \ t \ \ \Delta \omega_i \end{bmatrix}^T
\]

where

\[
A_0 = B_0 = C_0 = 0
\]

\[
\omega_0 = 2\pi \cdot f_{0,\text{est}}
\]

The least-squares solution to this system gives the updated estimates for the next iteration as:

\[
\begin{bmatrix} A_{i+1} \\
B_{i+1} \\
C_{i+1} \\
\Delta \omega_{i+1} \end{bmatrix} = \left( \mathbf{D}_i^T \mathbf{D}_i \right)^{-1} \mathbf{D}_i^T \mathbf{v}
\]

\[
\omega_{i+1} = \omega_i + \Delta \omega_{i+1}
\]

Note that the least-squares fit in (17) can be computed with a more numerically-stable method such as Q-R decomposition [10]. The solution converges rapidly, and the preprocessor stops after a fixed number \( m = 7 \) iterations. The fitted parameters for (10) are:

\[
A' = A_m \\
B' = B_m \\
C = C_m \\
f_0 = \frac{\omega_m}{2\pi}
\]

Finally, we convert this fit into the equivalent polar form:

\[
v[n] = A \cdot \sin \left( 2\pi n f_0 / f_s + \phi_0 \right) + C
\]

by computing:

\[
A = \sqrt{A_m^2 + B_m^2}
\]

\[
f_0 = \frac{\omega_m}{2\pi}
\]

\[
\phi_0 = \arctan(A_m/B_m)
\]

\[
C = C_m
\]

This form is preferable because the parameters \([A, f_0, \phi_0, C]\) are more directly applicable for computing spectral envelopes.

B. Implementation of nilm-sinefit

The nilm-sinefit tool, implemented in Python, uses successive 4-parameter sine wave fits to find and mark every positive zero crossing (\( \phi = 0 \)) of an input voltage waveform. Each mark includes the amplitude \( A \), frequency \( f_0 \), and offset \( C \) of the following period. Given the expected line frequency \( f_{\exp} \) and sampling rate \( f_s \), the fits are calculated over sliding windows of

\[
N = 3.5 \cdot f_s / f_{\exp}
\]

samples of voltage. This corresponds to approximately 3.5 periods, although the actual number will vary with \( f_0 \). If the fitted \( f_0 \) falls outside predetermined bounds, or the amplitude \( A \) is too low, the window is skipped to avoid spurious marks.

Fig. 4 demonstrates the fit and marking over a window of length \( N \). To avoid potentially double-marking a zero crossing that occurs near window boundaries, the point \( N_s \) is calculated as the last point within \([0, N]\) with phase angle \( \frac{\pi}{2} \). Only zero crossings prior to \( N_s \) are marked, and the next window is shifted to \([N_s, N_s + N]\). This ensures that any particular zero crossing will only fall squarely within one window, even as sine fit parameters change.

The nilm-sinefit processing tool takes as input a single NilmDB stream, and marks zero crossings using data from a user-specified column in the input stream. Output marks are written to a new stream consisting of a timestamp \( t \) and three
extra phase rotation to apply, to correct for known phase offsets.

Table 1: Parameters for nilm-prep.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{harm}}$</td>
<td>4</td>
<td>Number of odd harmonics to store.</td>
</tr>
<tr>
<td>$N_{\text{shift}}$</td>
<td>1</td>
<td>Number of shifted windows for which to compute coefficients, per zero crossing.</td>
</tr>
<tr>
<td>$\phi_{\text{extra}}$</td>
<td>0</td>
<td>Extra phase rotation to apply, to correct for known phase offsets.</td>
</tr>
</tbody>
</table>

Floating-point values $A$, $f_0$, and $C$. These values correspond to the parameters from (21), as calculated for the crossing detected at time $t$.

C. Implementation of nilm-prep

The nilm-prep tool, implemented in Python, reads two data streams: an input waveform $i[n]$, and the zero crossing and $f_0$ estimates for each period as marked by nilm-sinefit. Other parameters that control its behavior are shown in Table 1. For each identified zero crossing at time $t$ with frequency $f_0$, the spectral envelopes are calculated with the following algorithm:

1) Initialize $N = f_0 / f_0$ and $\phi_{\text{shift}} = 0$.
2) Extract $N$ samples starting at time $t + (\phi_{\text{shift}} / (2 \pi f_0))$.
3) Use the Fast Fourier Transform to calculate $X_k$ of $i[n]$, as defined in (3).
4) Apply rotation from (9) using $\phi_0 = \phi_{\text{extra}} - \phi_{\text{shift}}$ to obtain $X'_k$.
5) Calculate and store the first $N_{\text{harm}}$ odd harmonic coefficients from $X'_k$ using (5) and (6).
6) Increment $\phi_{\text{shift}}$ by $2 \pi / N_{\text{shift}}$.
7) Repeat from step 2 until $\phi_{\text{shift}} \geq 2 \pi$.

This results in $N_{\text{shift}}$ sets of coefficients per period of the input waveform, by applying successive overlapping DFTs. This sliding-window approach may be useful in some cases, as it can help retain additional information about energy content at harmonics that are not otherwise stored by this implementation, such as the even harmonics. For waveforms known to consist primarily of the low-numbered odd harmonics, $N_{\text{shift}} = 1$ is sufficient and results in the most space savings.

Output from nilm-prep is stored in a new NilmDB stream where each row contains a timestamp and $2 + N_{\text{harm}}$ floating-point values numbers, in order $\{a_1, b_1, a_3, b_3, \ldots\}$.

IV. COMPRESSION BENEFITS

Physical modeling of ac loads shows that there are often useful bounds on the number and types of harmonics that are present in the load current. By omitting these higher harmonics from stored data streams, the spectral envelope preprocessor performs a domain-specific form of compression that greatly reduces data storage and transfer requirements. A typical NILM data acquisition is 16-bit samples at a rate of 8 KHz. For a three-phase system, 6 channels are recorded. Assuming one 64-bit timestamp is also stored with each sample, the storage requirements for raw data are:

$$8 \text{ KHz} \cdot (64 + 16 \cdot 6) \text{ bits} = 13.8 \text{ GB / day}. \quad (27)$$

Preprocessor output is harmonics $a_k$ and $b_k$ for $k = 1, 3, 5, 7$ as 32-bit floating point values with a 64-bit timestamp, for each of the three phases. One set of coefficients is calculated per 60 Hz period, so the total storage requirements for pre-processed data are:

$$60 \text{ Hz} \cdot (64 + 32 \cdot 8) \text{ bits} \cdot 3 = 0.62 \text{ GB / day}. \quad (28)$$

Thus, the preprocessor reduces the data to only 4.5% of its original size, while preserving all information about three-phase power usage up to the 7th harmonic. Even doubling this and storing up to the 15th harmonic still uses only 8.1% of the original raw data space.

Other reductions of the data could instead be applied, for example, simply recording aggregate real power averaged over every period or second. However, such data would not reflect the detailed short term variations that would occur in real power, nor would it reflect any of the behavior of the higher harmonics. Time-varying spectral envelope coefficients strike a balance between the usefulness of the retained data and the storage and transmission requirements of that data.

V. RESOLUTION AND ACCURACY

The original digital samples of the current and voltage waveforms have limited precision due to quantization. As shown in Fig. 5, the continuous input signal is divided into $B$ discrete regions, and each region is mapped to a unique
digital code during the sampling process. This quantization can be stated explicitly as
\[ i[n] = \left\lfloor 2^B \cdot i(t) + 0.5 \right\rfloor \]
where \( i(t) \) is normalized to the range \([0, 1]\). The preprocessor input consists of \( N \) values of \( i[n] \) over one period of the line voltage.

For power estimation and load identification purposes, it is desirable to find the average power of the input signal, but the quantization error affects this measurement. Using raw data to directly estimate power is less accurate than using the quantization error.

A. Resolution of Power From Raw Data

A straightforward approach to estimating \( A \) from raw sampled data is to locate the peak of the waveform. However, as shown in Fig. 6, finding the amplitude in this manner leads to measurement error, because the single point at the peak is subject to the same quantization as any other point. Of the \( 2^B \) possible quantized sample values, the positive peak will reside in the upper half, resulting in a total of \( 2^{(B-1)} \) discernible amplitudes. This corresponds to an overall resolution of
\[ \log_2 2^{(B-1)} = B - 1 \]
bits of precision, slightly less than the original sampling precision of \( B \) bits.

B. Resolution of Power From Preprocessor Output

The preprocessor improves effective power resolution by averaging over time. Specifically, \texttt{nilm-prep} uses not only the quantized value at the peak, but all \( N \) sampled data points over one or more periods. For example, the preprocessor discards all but the low, odd-numbered harmonics; the sinusoid in (30) contains no harmonics other than for \( k = 1 \). Thus, the \texttt{nilm-prep} output encapsulates all information about the original signal. Since the analogous DFT is invertible, this means that every unique sampled input has a unique corresponding preprocessor output.

We can use this to determine how many discernible outputs the preprocessor will produce as the amplitude \( A \) of the input sinusoid changes. The sampling process that generates \( i[n] \) is limited in resolution, and so there are necessarily a finite number \( U \) of unique sampled \( i[n] \) waveforms. Because of the 1:1 relationship of the DFT, there are the same number \( U \) of unique preprocessor outputs, and we can therefore discern \( U \) different values of \( A \).

To calculate \( U \), consider sweeping \( A \) from 0 \( \rightarrow \) 1, as demonstrated in Fig. 7. By symmetry, let us consider only the first \( N/4 \) samples. At \( A = 0 \), all samples are zero. As \( A \) increases, there will be some transition where a single quantized sample \( i[n] \) will increase by one. More specifically, as \( A \) increases from 0 to 1, samples will monotonically increase according to the following pattern:

- The first sample at \( n = 0 \) never increases.
- The peak sample at \( n = N/4 \) increases \( 2^{(B-1)} - 1 \) times.
- For each sample between these extremes, the number of “increases”, or effective quantization steps, is given by:
\[ U_n = \left\lfloor (2^{(B-1)} - 1) \sin \left( \frac{2\pi n}{N} \right) + \frac{1}{2} \right\rfloor \]

where \( [x] \) denotes floor(\( x \)).

Each increase creates a new input to the preprocessor, which results in a new output. Thus, the total number of unique outputs from the preprocessor is one corresponding to the initial case (\( i[n] = 0 \)), plus one for each time a sample in \( i[n] \) increases.

\[ U = 1 + 0 + \left( \sum_{n=1}^{N-1} U_n \right) + (2^{(B-1)} - 1) \]
\[ = 2^{(B-1)} + \sum_{n=1}^{N/4-1} (2^{(B-1)} - 1) \sin \left( \frac{2\pi n}{N} \right) + \frac{1}{2} \]

Fig. 8 shows this number of unique outputs \( U \) as a function of the input samples \( N \) and input quantizer bits \( B \). The number of output bits are approximately linear with the number of input bits, and increase logarithmically with \( N \). As a representative example,
data point, sampling at values $N = 128$ and $B = 10$ gives an overall output resolution of about $B = 13$ bits.

Note however that the unique outputs of the preprocessor are not linearly distributed among the input amplitudes. For example, for amplitudes less than the first quantized bit level of $1/(2^B - 1)$, all quantized input samples are zero, and so there is only one unique preprocessor output. Fig. 9 shows the number of unique outputs as a function of the input amplitude, as enumerated computationally via binary search (described later in Algorithm 1). As the input amplitude is swept along the $x$-axis from left to right, the cumulative number of unique outputs is counted and plotted along the $y$-axis. Inflection points in the curve correspond to the quantization levels. This is because small changes in amplitude affect more samples when the peak of the sinusoid is near a quantization level, causing relatively more unique outputs in these areas.

From Fig. 9, a measure of effective resolution as a function of input amplitude can be developed. If a change of amplitude $A$ to $(A + \Delta A)$ causes the preprocessor to output a new unique value, then discerning these values corresponds to discriminating between input amplitudes with a resolution of $\beta = \log_2 (1/\Delta A)$ bits around operating point $A$. Fig. 10 shows this resolution as a function of input amplitude. Here, the input bits $B = 4$ and samples $N = 512$, and the output resolution varies between $\beta = 4$ and 18 bits, with an average around 10 bits.

C. Accuracy of Preprocessor Output

For load identification, the ability to simply discern inputs may be sufficient. As shown, the preprocessor can provide a high resolution estimate of input waveform amplitude. In some cases, such as in power level measurement and energy scorekeeping, high accuracy is also desired.

When the characteristics of the input waveform are known, the accuracy can often be directly determined. To continue the previous example, consider a single sinusoidal waveform at the fundamental line frequency and amplitude $A$, corresponding to power $P$. This waveform is sampled, quantized, and passed through the spectral envelope preprocessor. Fig. 11 shows a plot of the error between the preprocessor output $P_1$ and the actual power $P$, as a function of $P$. Like the resolution, the error varies with signal amplitude. Note that the maximum absolute error corresponds to approximately one part in $2^B$, where $B$ is the quantization bits. Thus, the plot shows that the error of the preprocessor output is always less than or equal to the quantization error of any individual sample.
D. Accuracy in the Presence of Noise

The previous analysis has been calculated and simulated based on ideal waveforms. In practice, there are many potential sources of noise or interference in non-intrusive load monitoring. This noise can be broadly split into two categories, correlated and uncorrelated signals.

Correlated interference refers to any introduced signal or distortion that is related to the loads being monitored or is otherwise nonrandom. Examples include magnetic coupling between adjacent current transducers, pickup of stray 60 Hz electric fields from other shielded wiring or lighting fixtures, and aliasing effects in the data acquisition process. This sort of coupling can potentially be complex and relate closely to specific installation details, which means that its effect on preprocessor accuracy is highly variable. In general, the net effect of correlated noise has been extremely low in observed systems [5].

Uncorrelated noise is statistically independent from the input waveforms. One such type of noise is additive white Gaussian noise (AWGN), which is normally-distributed around zero and might be expected to appear as a result of thermal noise in the sensors or data acquisition. This form of noise often sets the limit of sampling resolution in the monitoring system.

The preprocessor is particularly well-suited to handle such disturbances. To analyze the effect of AWGN on preprocessor accuracy, we again consider the example of a single sinusoid, with added noise $\mathcal{N}(t)$:

$$i(t) = A \sin(\omega t) + \mathcal{N}(t) \quad (35)$$

By sweeping $A$ from $0 \to 1$, we can generate data in the same manner as Fig. 11, simulating the sampling process and calculating the preprocessor power estimate $P_1(A)$ corresponding to each actual power input $A$. Define the overall root-mean-square error of this data as:

$$E_{RMS} = \sqrt{\int_0^1 (P_1(A) - A)^2 dA} \quad (36)$$

This RMS error will vary with the amount of noise $\mathcal{N}(t)$ that is being added to the sinusoid. We describe this amount using the signal-to-noise ratio (SNR), the ratio of power in the full-amplitude sinusoid to the power of the added white Gaussian noise. Note that the amount of noise is held constant as $A$ is swept.

Fig. 12 shows the calculated overall RMS error versus the SNR of injected noise. Two approaches to estimating power are shown: the preprocessor $P_1$ estimate, and the simple peak-estimation technique described in §V-A. As expected, the preprocessor estimate offers an improvement over the raw peak estimate in all cases, reducing overall error by more than half.

In some cases, the addition of noise will actually reduce error. Here, noise levels around 30 dB provide a slight accuracy improvement. This is due the dithering effect of the white Gaussian noise on the sampling quantization which, combined with the averaging effect of the spectral envelope calculation, serves to decouple the quantization error from the input waveform [12]. The preprocessor can therefore be seen as even more useful in the presence of noise.
The harmonics are held at a fixed ratio while the overall amount of harmonic content is increased. The effective bits of preprocessor resolution as relative consistency. Fig. 13 shows a plot of the calculated effective bits that the bits-per-amp ratio is fixed for all trials, to maintain maximum and minimum resolution from Fig. 13. In general, these results demonstrate that adding harmonic content increases the effective resolution, although some combinations of harmonics reduce it slightly due to “flat” regions that do not vary much with power scaling.

Algorithm 1: Binary search to enumerate unique preprocessor outputs as power $A$ varies, given waveform model $f(A)$.

```
outputs ← []
function ENUMERATE(f, Alow, Ahigh)
    Plow ← PREPROCESS(SAMPLE(f(Alow))))
    Phigh ← PREPROCESS(SAMPLE(f(Ahigh))))
    if Plow ≠ Phigh then
        A_mid ← (Ahigh + Alow)/2
        ENUMERATE(f, Alow, A_mid))   // Search 1st half
        ENUMERATE(f, A_mid, Ahigh))   // Search 2nd half
    else if Plow not in outputs then
        outputs ← outputs + Plow
    ENUMERATE(f, 0, 1)   // Add if unique
return SORT(outputs)
```

VI. CONCLUSIONS

The spectral envelope preprocessing has consistently extracted relevant harmonic information while providing data storage reduction. The applicability of the preprocessor to complex systems such as multi-phase systems and variable speed drives has been improved by the development of a more flexible and modular “site-fit” preprocessor design with improved performance and frequency estimation. The new preprocessor is designed to integrate with the NilmDB framework and builds upon its ability to correlate, manipulate, and retrieve interrelated data streams.

Acknowledgements

This research was funded by the BP-MIT Research Alliance, the ONR Structural Acoustics Program, and The Grainger Foundation.

References