

Finding CPU Fails: A Wavelet Approach

BY SIDNEY TAYLOR

What is a CPU failure and why do we care?

- CPUs are rigorously tested before leaving production factories
- CPU frequency is measured during “intense” conditions (games, renders, complex calculations)
- Most of the time, it “passes” testing and is good to go
- When it doesn’t pass, engineers would like to know how it failed, so they can figure out why
- Catching fails early means they can save money and save time if a customer has an issue

What's wrong with the current approach?

- Current methods for analyzing fails are slow and expensive
- Companies aim to save money, cutting corners to cut costs
- Fails are missed in screening, meaning more people get worse parts
- DSP can help to reduce dimensionality, everyone wins

Timestamp_s	Core1_Freq_GHz	Core1_Temp_C	Core1_Util_Percent	Core2_Freq_GHz	Core2_Temp_C	Core2_Util_Percent
0	4.402	58.5	87.3	4.365	51.4	76.3
0.1	4.36	55.2	81	4.385	48.9	70.9
0.2	4.417	54.7	85.9	4.399	47.4	88.9
0.3	4.376	52.9	89.9	4.337	42.3	76.9
0.4	4.358	50.4	87.7	4.402	49	61.1
0.5	4.375	55.6	77.7	4.389	51.6	85.6
0.6	4.409	54.7	79.2	4.362	49.4	88.9
0.7	4.414	56.7	74.2	4.388	46	89.9
0.8	4.402	51.2	94.7	4.358	52.5	86.4
0.9	4.41	59.6	84.6	4.329	51.3	81.8
1	4.418	57.2	78.5	4.369	49.1	83.7
1.1	4.382	50.5	91.9	4.388	50.2	100
1.2	4.402	58.7	80.2	4.325	49.2	82.6
1.3	4.38	53.4	89.9	4.317	49.4	78.5
1.4	4.411	56.5	100	4.358	51.7	95.8
1.5	4.441	56.7	95	4.333	47.9	93.3
1.6	4.439	53.4	84.2	4.303	53.3	98.7
1.7	4.474	48	84.7	4.354	45.8	91.1
1.8	4.435	57.6	100	4.324	47.8	86.9
1.9	4.393	54.3	84.4	4.31	46.1	95.1
2	4.473	54.8	89.3	4.31	46.5	79.4
2.1	4.466	57.7	89.6	4.307	43.5	78
2.2	4.431	53.8	95.7	4.334	52.3	72.7
2.3	4.451	58.2	88.7	4.289	48.3	98
2.4	4.465	56.9	90.4	4.317	48.1	100
2.5	4.461	53.4	90	4.324	48.8	90.5
2.6	4.47	59	100	4.284	45.8	74.3
2.7	4.456	57.7	95.8	4.306	47.1	68
2.8	4.46	60.1	78.4	4.265	50.2	70.9
2.9	4.464	53.7	75.1	4.308	48	80.4
3	4.45	53.5	71.2	4.269	49.1	66.8

This is just 3 seconds of data sampled at 10 Hz for 2 cores...

How can we fix this?

- Wavelet analysis to break down the signal
- Looking at different reconstructions to understand data
- Creating 'features' based on the reconstructions
- Using features to classify the frequency data into groups

Generate data

Filter data

Do wavelet analysis

Create features

Classify failures






Generating the data

Finding data online vs generating it programmatically

- Finding data online caused many issues in this project
 - Incomplete entries, undefined fails, raw data size
- Data is kept private; companies don't want to give free training data
- Generating data manually gives full control of the *fail modes*

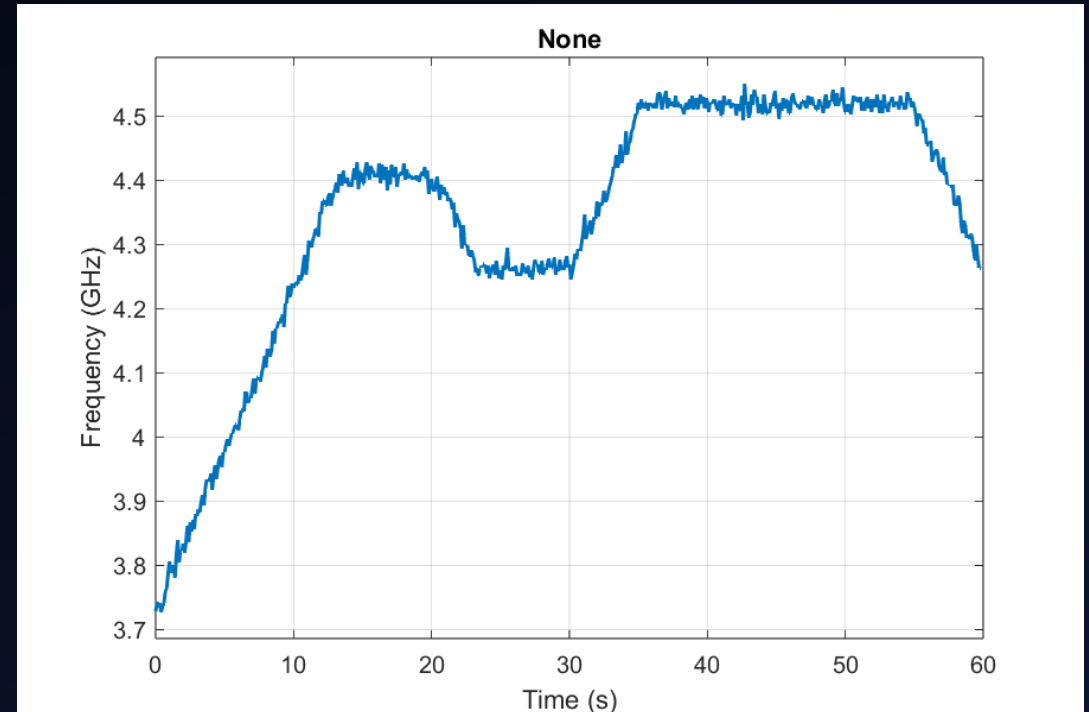


Time (s)	Correct, Freq, 10%	Correct, Temp, 1%	Correct, 10%, Percent
0	4.402	10.0	87.0
0.1	4.38	10.2	86
0.2	4.47	10.7	85.9
0.3	4.37	10.9	86.9
0.4	4.4	10.4	87.7
0.5	4.4	10.6	87.7
0.6	4.4	10.7	86.2
0.7	4.404	10.7	86.2
0.8	4.402	10.2	86.7
0.9	4.41	10.6	86.4

```
%generate data for each run
for run = 1:num_runs
    t = (0:num_samples-1) / fs;
    %get chosen failtype
    failure_type = failure_distribution(run);
    %generate frequency data
    freq_profile = generate_frequency_profile(num_samples, base_freq, boost_freq, fs);
    %inject fail
    if failure_type > 0
        freq_profile = inject_failure(freq_profile, num_samples, fs, failure_type);
    end
    %combine into single dataset
    run_data = [t(:), freq_profile(:), ones(num_samples, 1) * run, ones(num_samples, 1) * failure_type];
    %append
    all_data = [all_data; run_data];
end
```

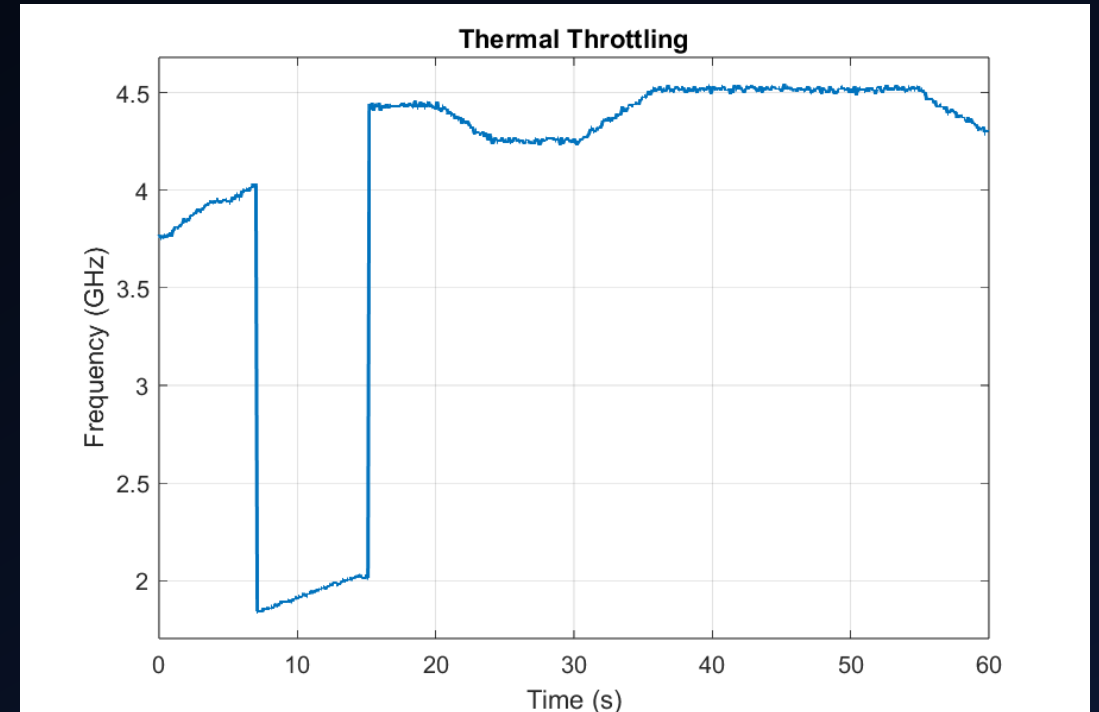
Mode 1 – No fail

- Ideal case for a working CPU – no odd behavior
- Frequency ramps up in first 10 seconds and is held
- Lighter workload after 20 seconds
- “Full stress” workload for 30 seconds
- Test ends in last 5 seconds



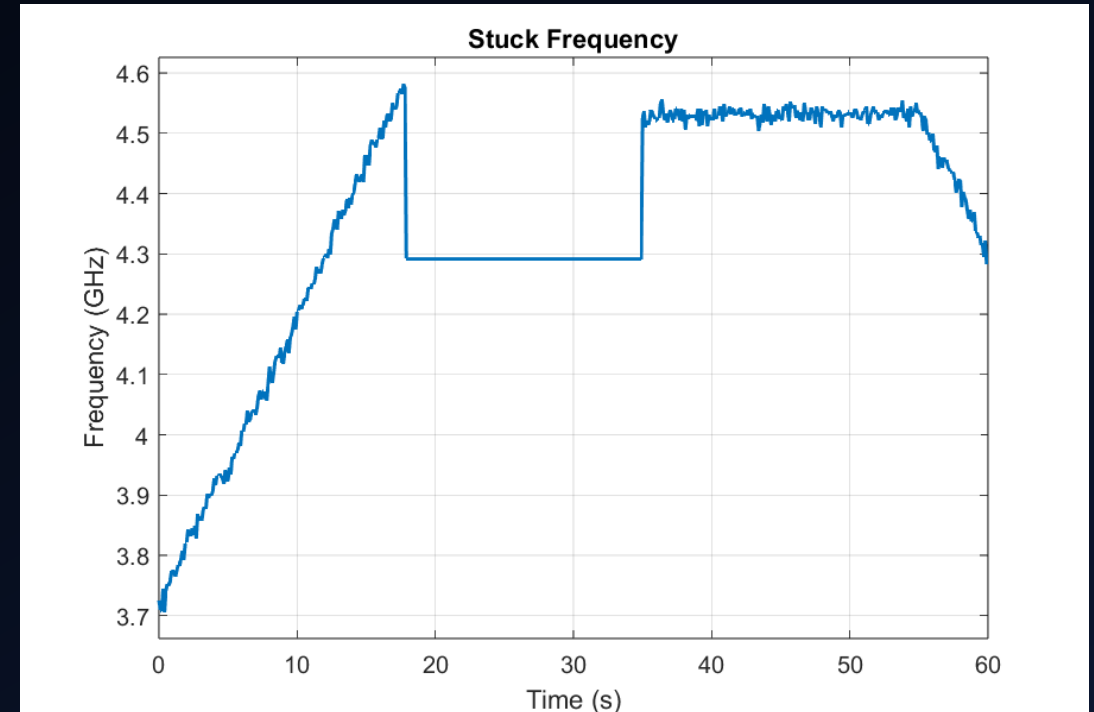
Mode 2 – Thermal throttling fail

- Occurs when the CPU exceeds a safe temperature threshold
- Almost always a cooling issue
- Safety mechanism forces the frequency to drop to reduce power, allowing CPU to cool
- Drops are usually significant, but last for any amount of time



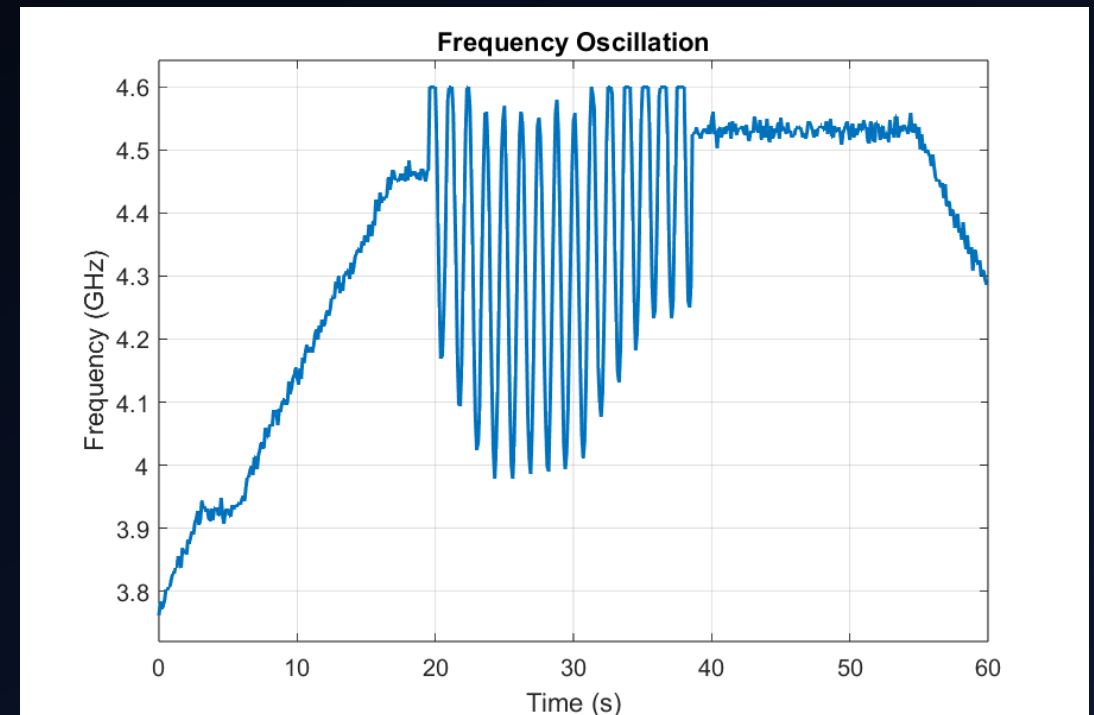
Mode 3 – Stuck frequency fail

- Happens when the CPU can't change frequency properly
- Usually caused by a firmware issue
- Instead of smooth behavior, sharp arbitrary drops happen
- Can last for a random amount of time, and at any value



Fail mode 4 – Frequency Oscillation

- Happens when there are power delivery or frequency control issues
- Caused by unstable power from VRMs or overclocking
- Frequency bounces up and down randomly from instability
- No center frequency, leads to increased power consumption





Filtering the data

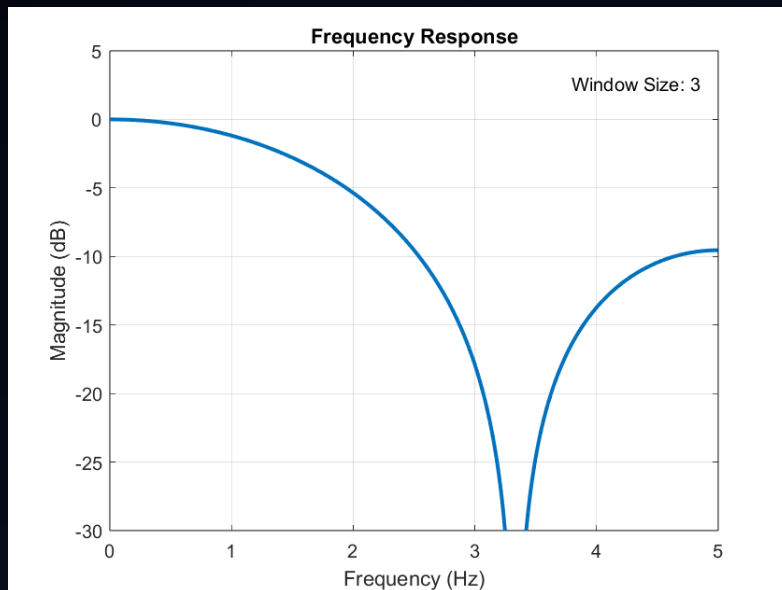
Why filter the data?

- Frequency is constantly changing as the CPU reacts to tiny changes
- Noise from the measurement of the frequency at a given sampling rate creates further instability
- For the best classification, we want to remove high frequency noise while retaining the underlying pattern

Preprocessing filters

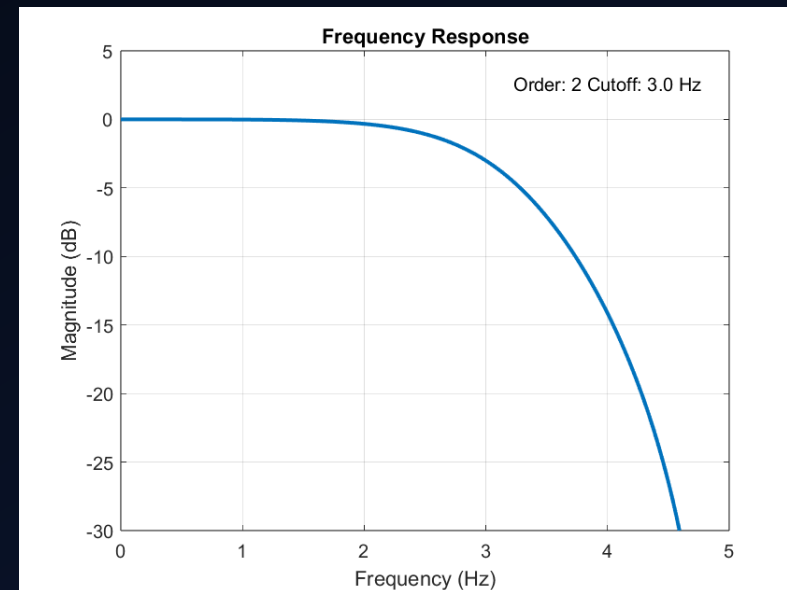
MOVING AVERAGE FILTER

Replaces each point with the average of itself and the points either side of it (noise reduction)



BUTTERWORTH FILTER

Advanced low-pass filter with a flatter frequency response, and smooth roll-off into blocked frequencies

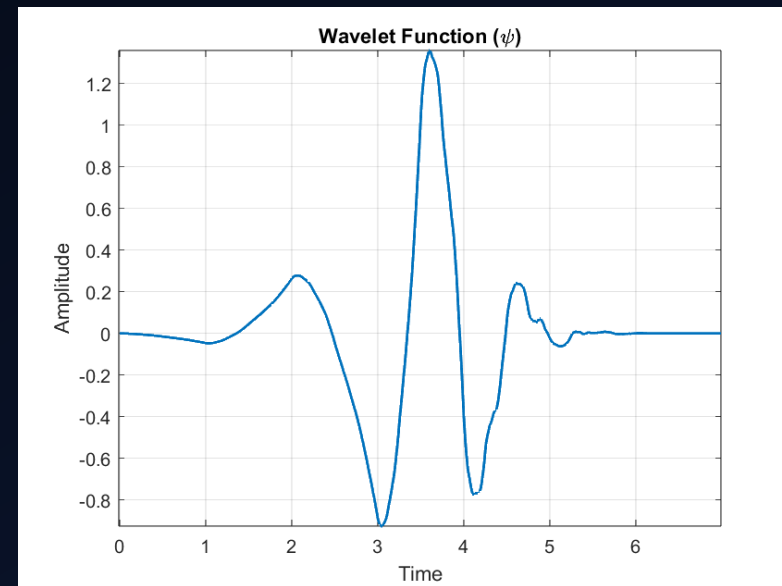
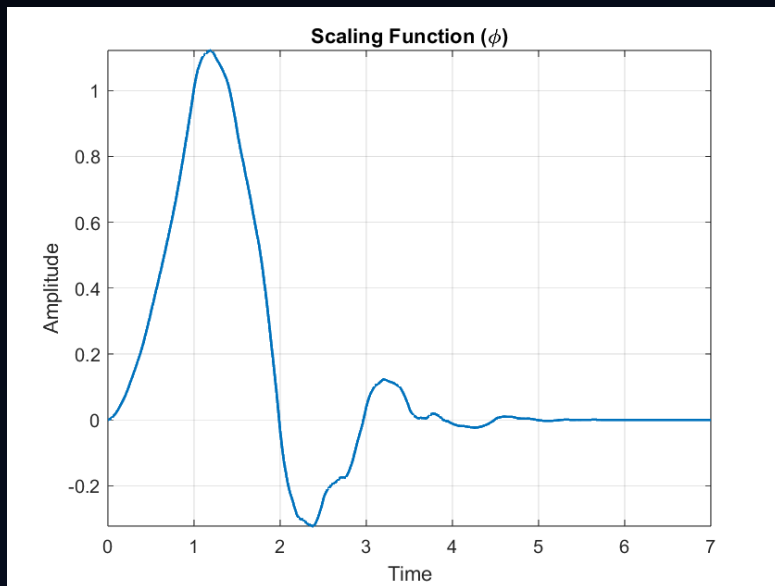




Wavelet analysis

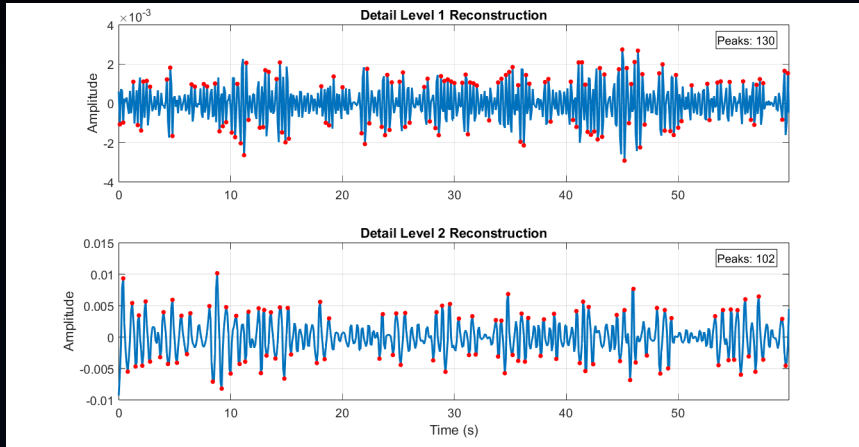
Finding the perfect wavelet

- Wavelet must have compact support and a few vanishing points
 - Compact support helps analyze events that happen at a specific time as the wavelet is finite
 - Vanishing points make the detail signal smaller when signal is not changing
- Db4 wavelet has 2 vanishing points and compact support



Interpreting the reconstructions

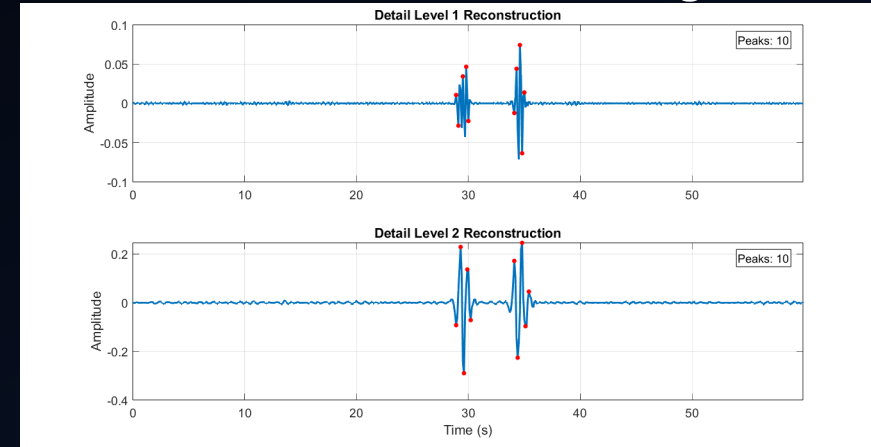
No fail



Magnitude scale is smaller as there are no high frequency changes during standard operation.

Reconstructions appear busy due to the tiny frequency adjustments throughout the run.

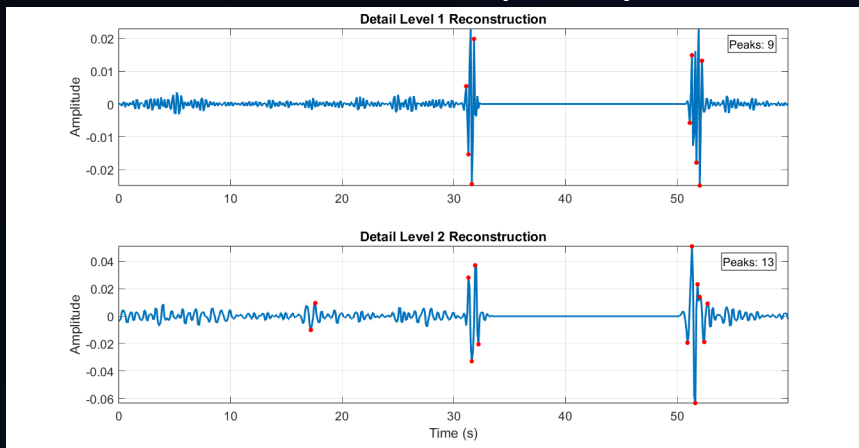
Thermal Throttling



Minimal high frequency change, most values are 0.

Sharp spikes represent the sudden changes in frequency are significantly higher energy.

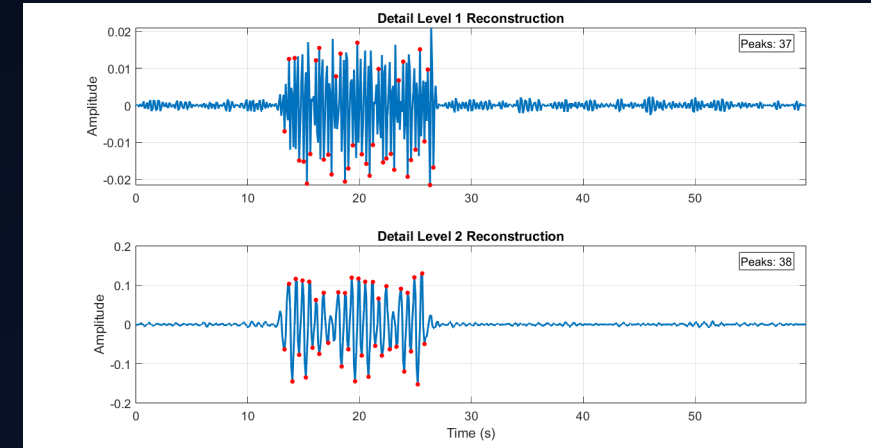
Stuck Frequency



The sharp spikes are much higher in relative magnitude than in thermal throttling, indicating larger relative change

While the frequency is locked, the amplitude is a perfect zero, only situation where this happens.

Frequency Oscillation



Most values are low as the frequency changes are relatively low energy throughout.

Sustained activity with significant amplitude variations occur when the frequency oscillation starts and finishes.



Feature creation

Why do we need features?

- Creating a metric based on the D1 signals means each run can be reduced from 600 numbers (samples) to just one number (feature)
- This makes digesting and classifying runs much easier and cheaper

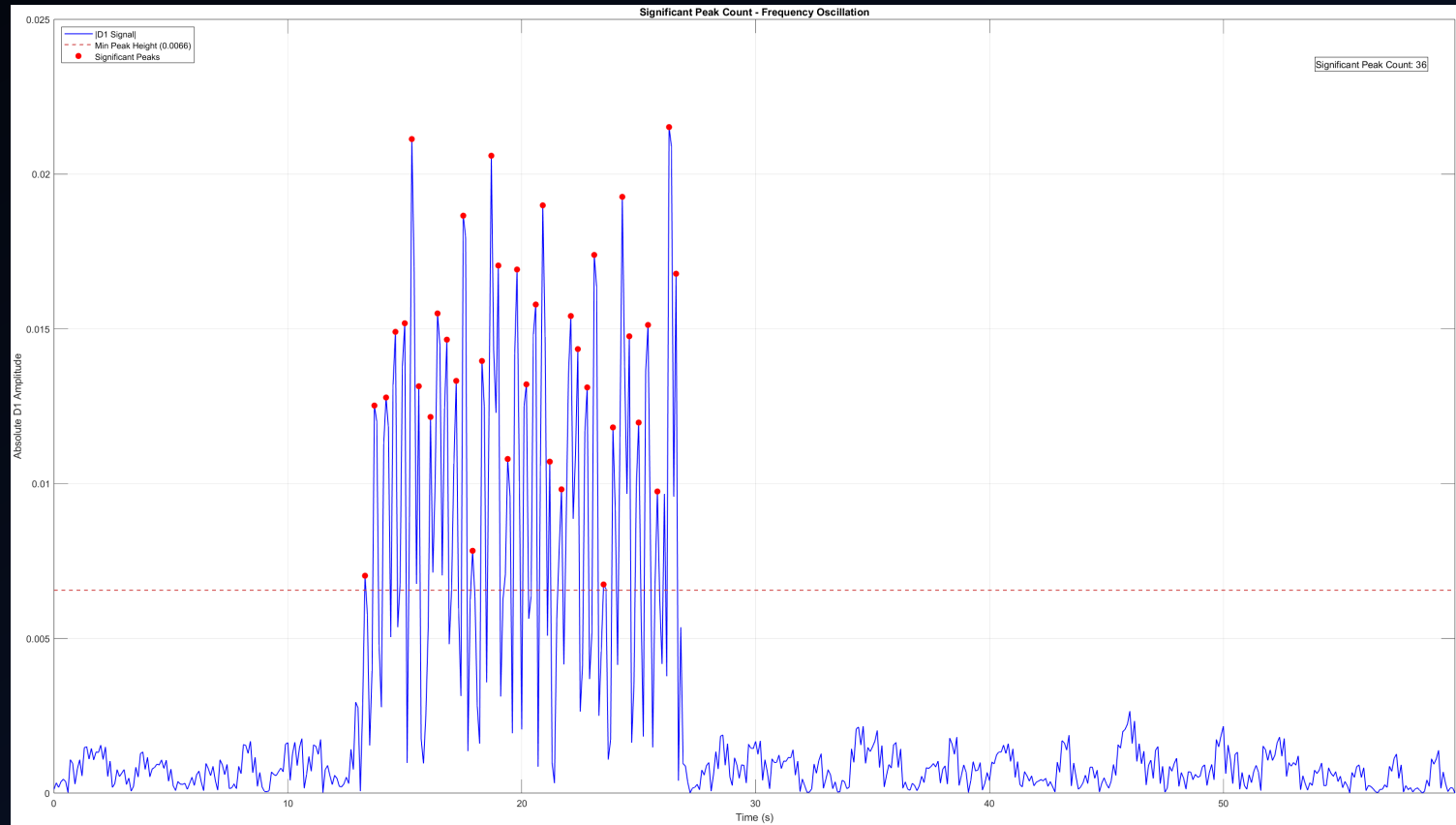
Time	Frequency_GHz
0	3.765581996
0.1	3.78611686
0.2	3.781347384
0.3	3.790379762
0.4	3.798135548
0.5	3.815354817
0.6	3.808825896
0.7	3.806020938
0.8	3.816074821
0.9	3.820405574
1	3.825194697
1.1	3.836659659
1.2	3.83846335
1.3	3.831575335
1.4	3.831639833
1.5	3.837534397
1.6	3.868391008
1.7	3.850053448
1.8	3.857235604
1.9	3.86299638
2	3.862143594
2.1	3.867552516
2.2	3.865380161
2.3	3.891585521
2.4	3.896646416
2.5	3.907600023
2.6	3.898046396
2.7	3.91580951
2.8	3.922789581
2.9	3.923549969
3	3.90997498
3.1	3.91831567
3.2	3.924587988
3.3	3.937290254
3.4	3.940296809
3.5	3.936267179
3.6	3.927576549
3.7	3.937755023
3.8	3.950259789
3.9	3.962825754
4	3.984658738
4.1	3.970473515
4.2	3.959196342
4.3	3.988055391
4.4	3.979752093
4.5	3.987655525
4.6	3.977904239
4.7	3.9906973
4.8	3.99483448
4.9	3.993373387
5	3.970791728



Feature 1: Number of significant peaks

What it is: Count of high-frequency spikes in the signal's detail level 1 component

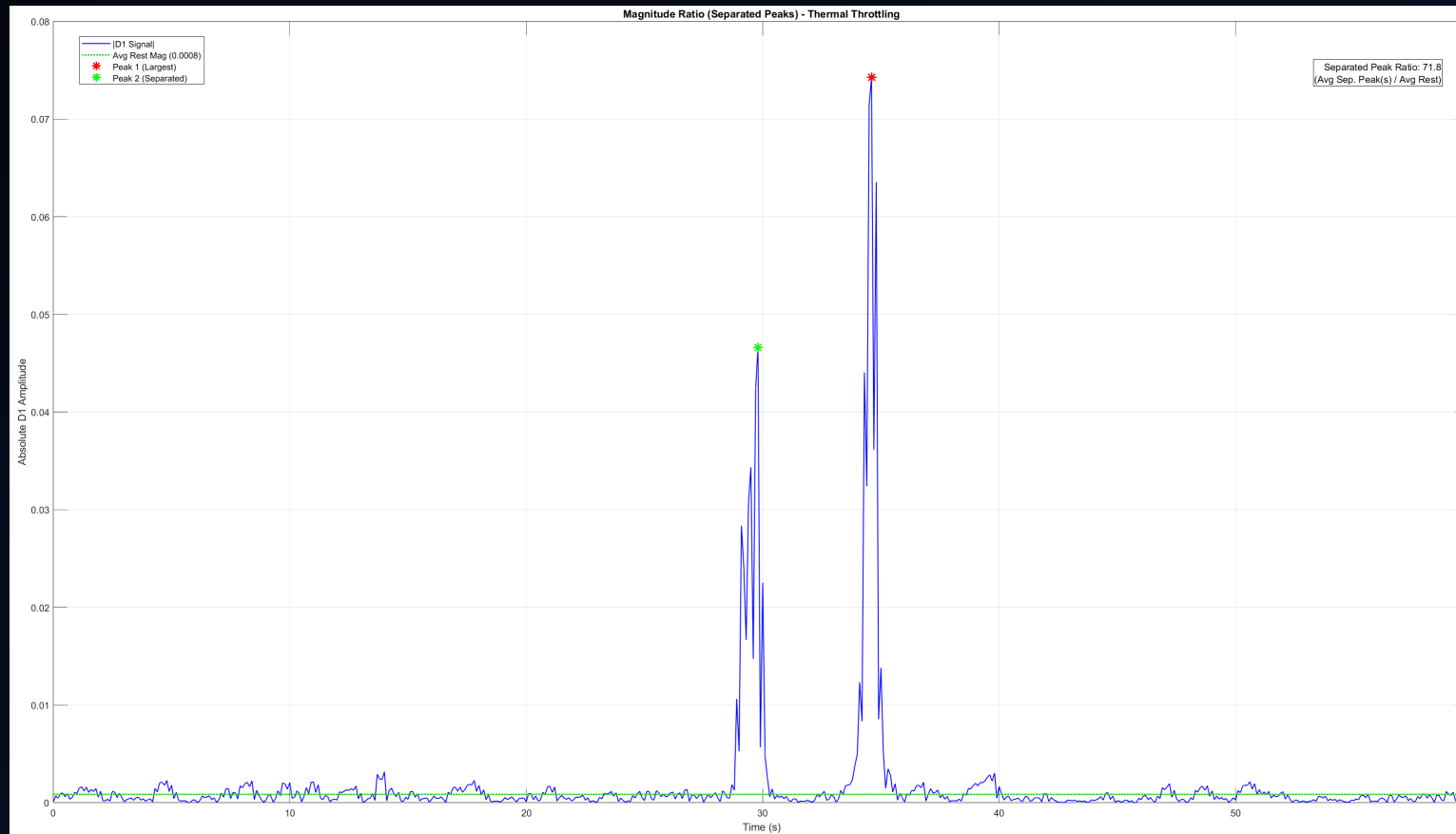
Calculation: Take the number of peaks above 1.5x the std dev of signal



Feature 2: Top 2 Peaks Magnitude Ratio

What it is: A measure of how much the largest peaks dominate the overall signal.

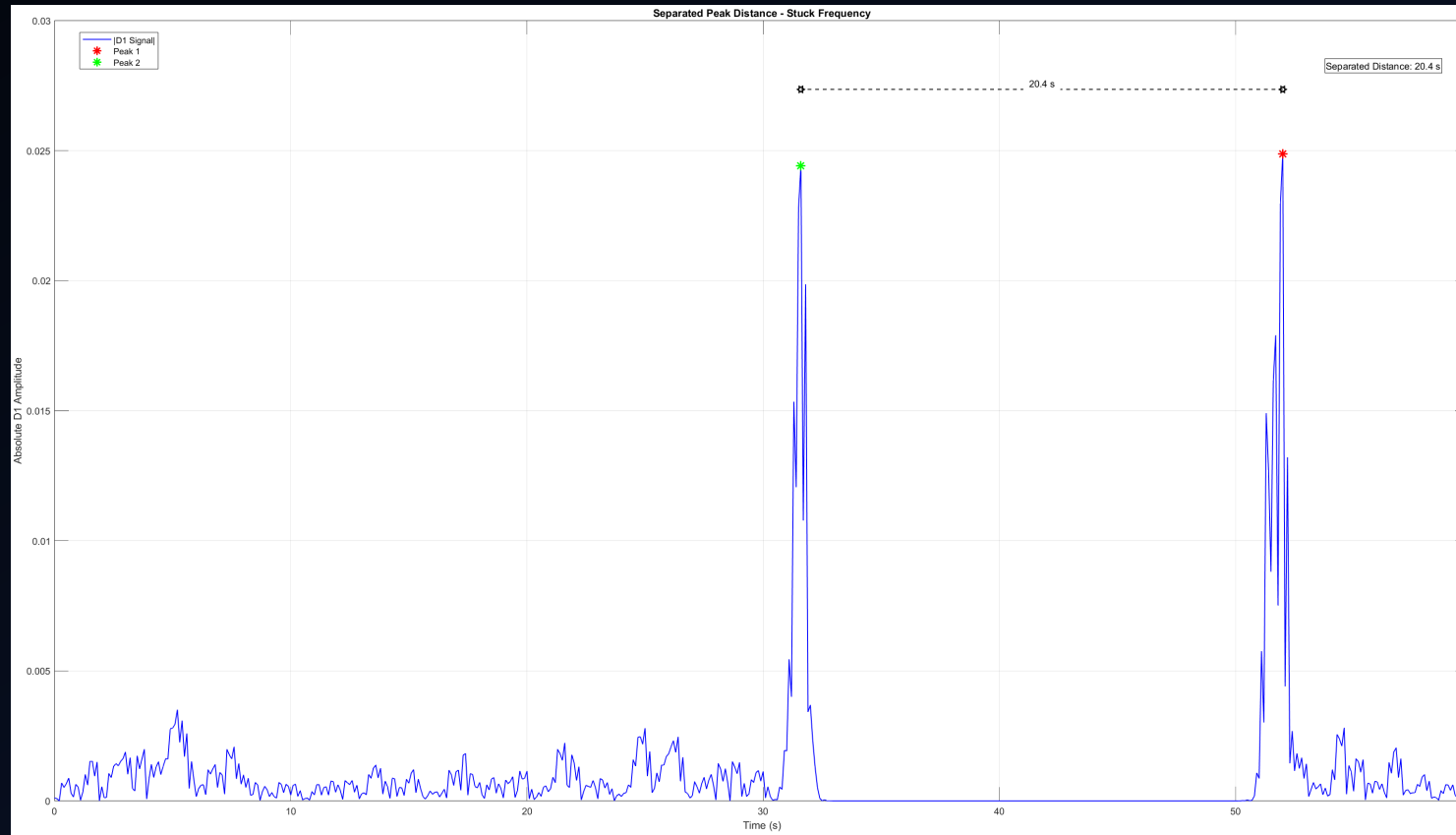
Calculation: Average of the top 2 peaks / average of the rest of the signal.



Feature 3: Separated Peak Distance

What it is: Time between the largest peak and the next largest peak (min dist. 3s).

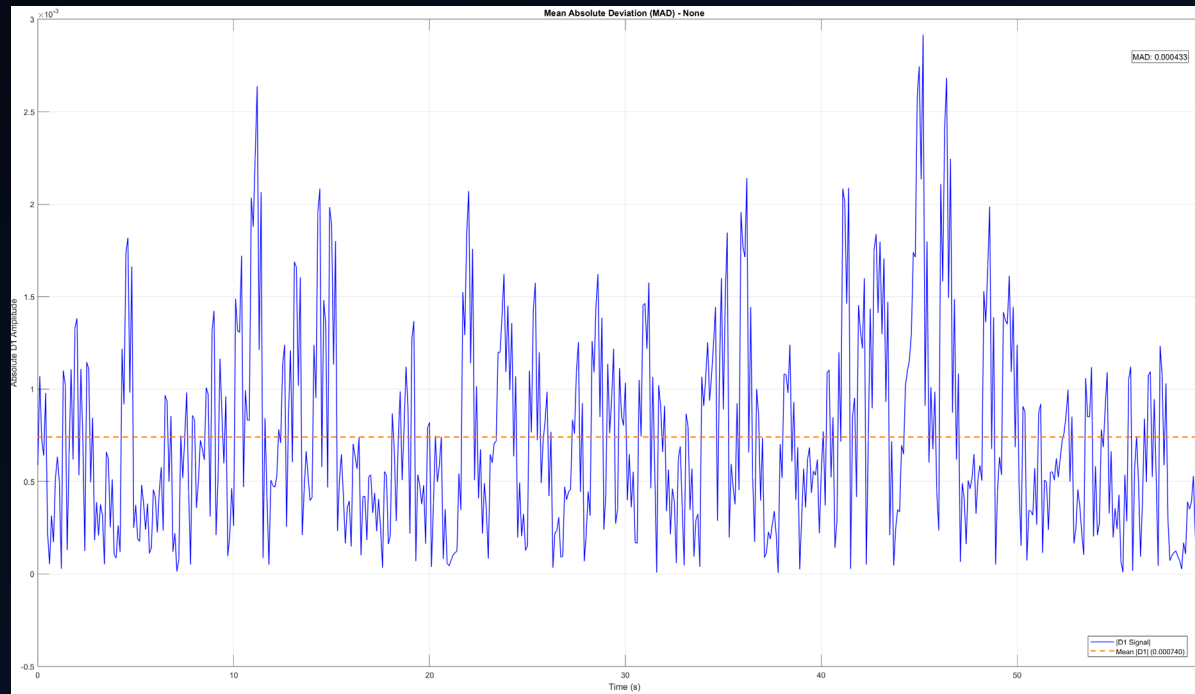
Calculation: Time difference between location of two most significant peaks.



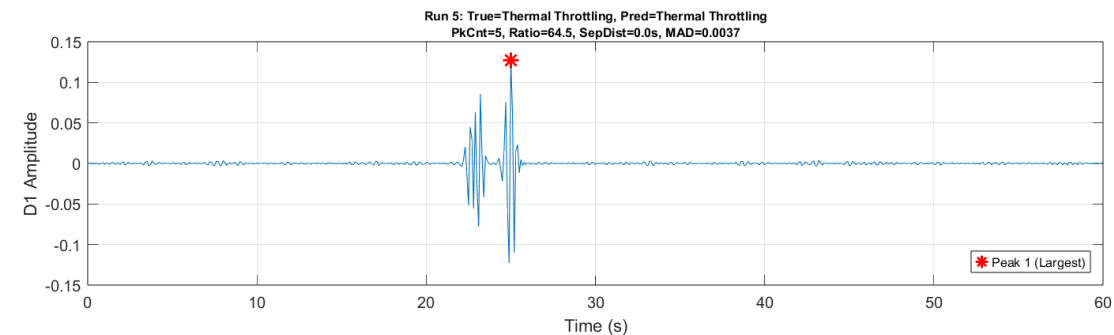
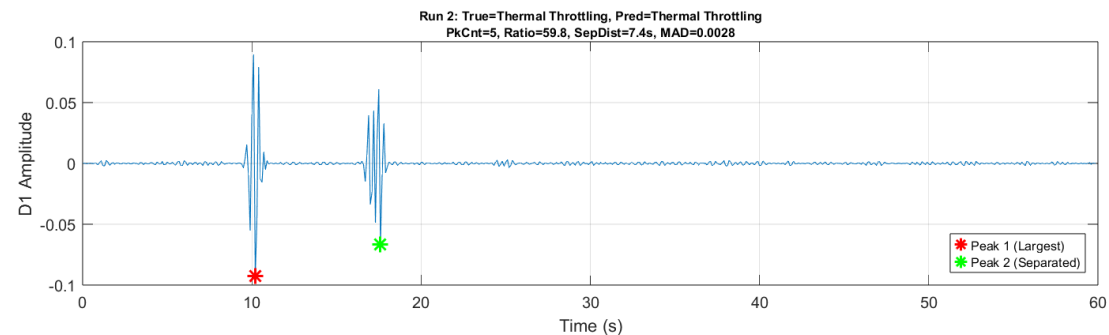
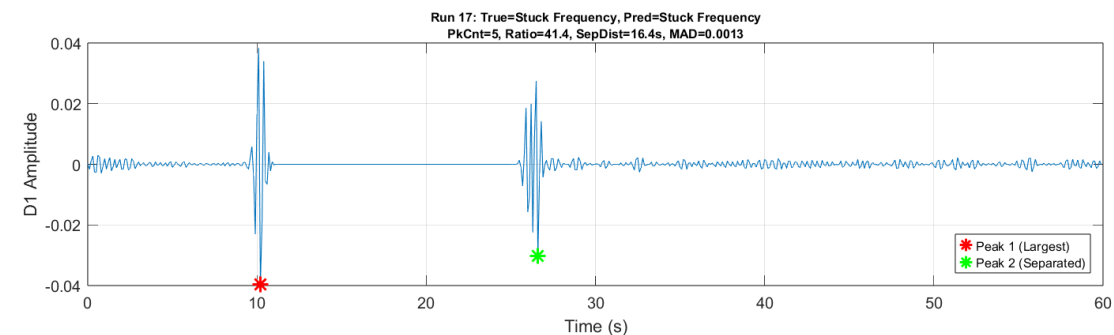
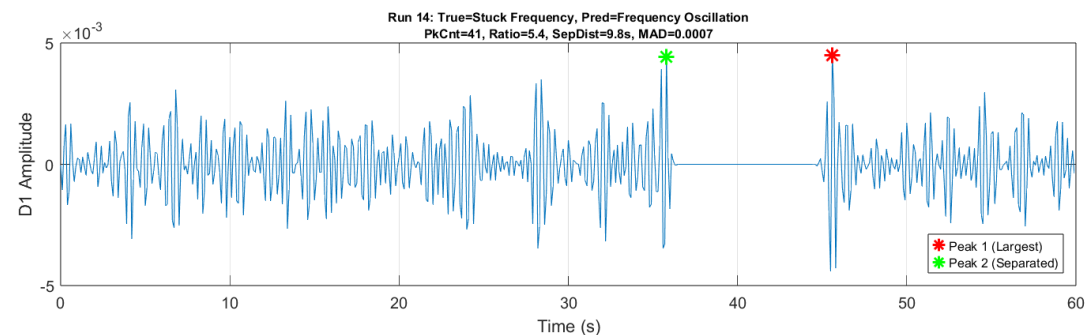
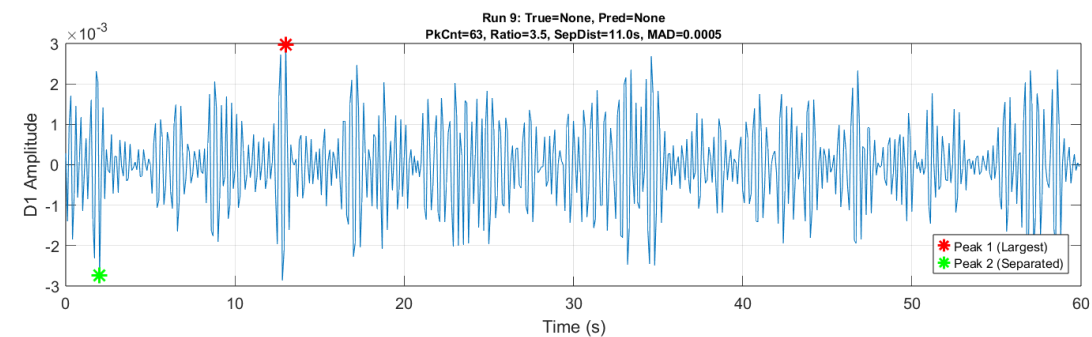
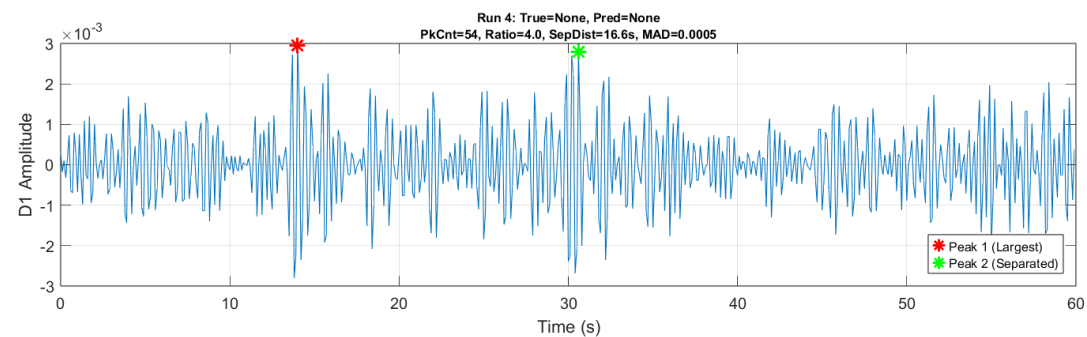
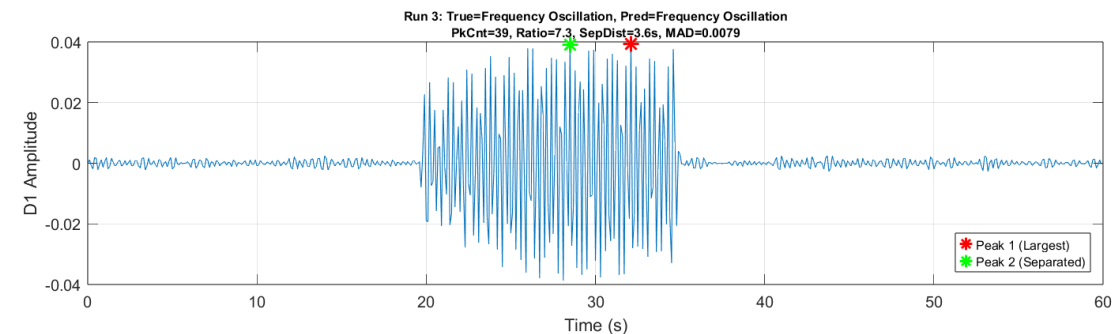
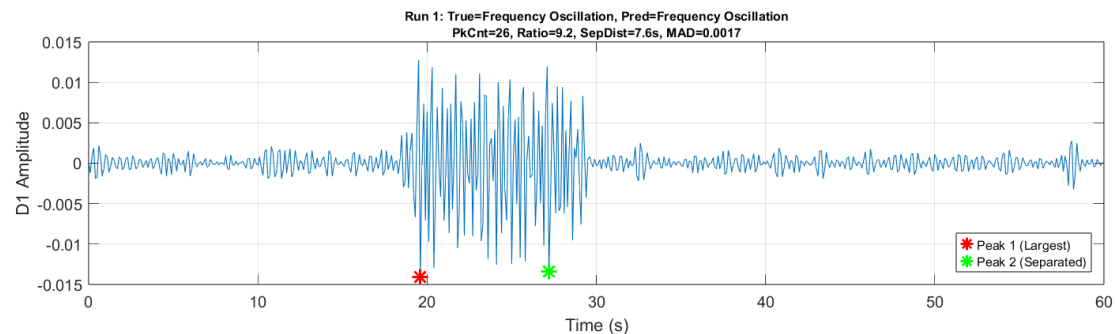
Feature 4: Mean Absolute Deviation (MAD)

What it is: The average variability of the magnitude of the D1 signal around its average magnitude.

Calculation: Take the average of the magnitude of the signal, then take difference between each point in signal and the average. Average of this value is the MAD.



Detail Level 1 Reconstruction with Calculated Features

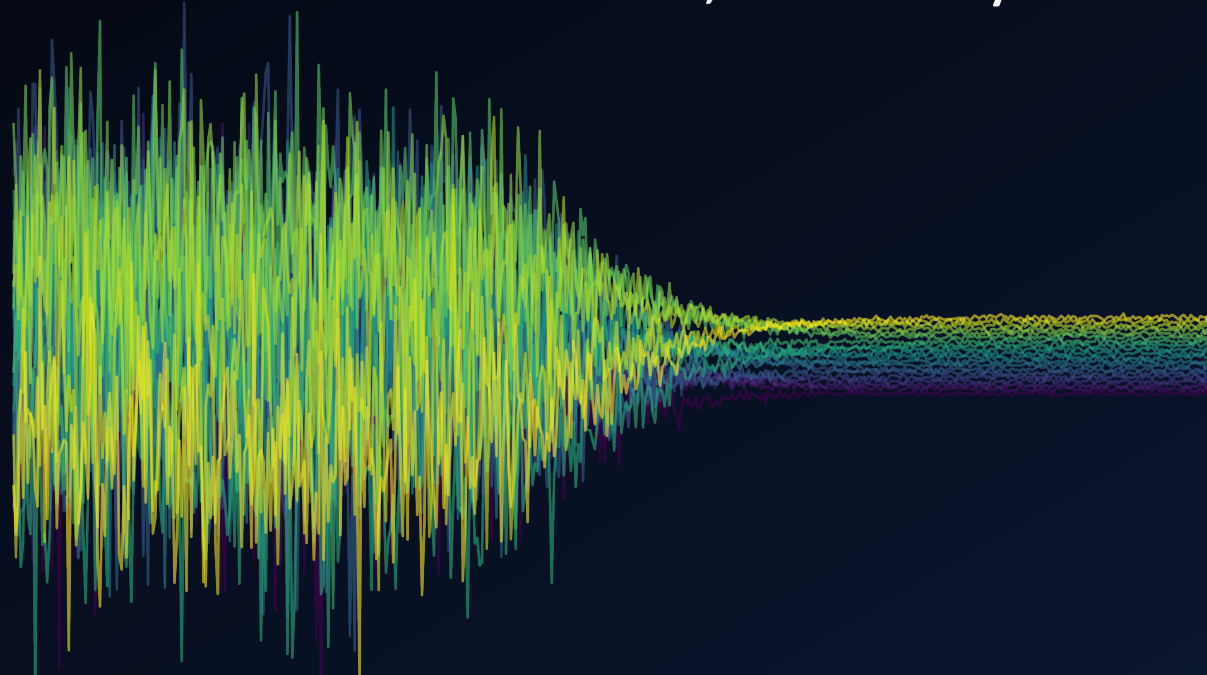




Fail classification

Building an algorithm

- With features defined, analyzing the plots to spot similarities between fail types allows us to build an algorithm
- Lots of trial and error needed to come up with thresholds
- After looking at features for each fail, manually derived algorithm



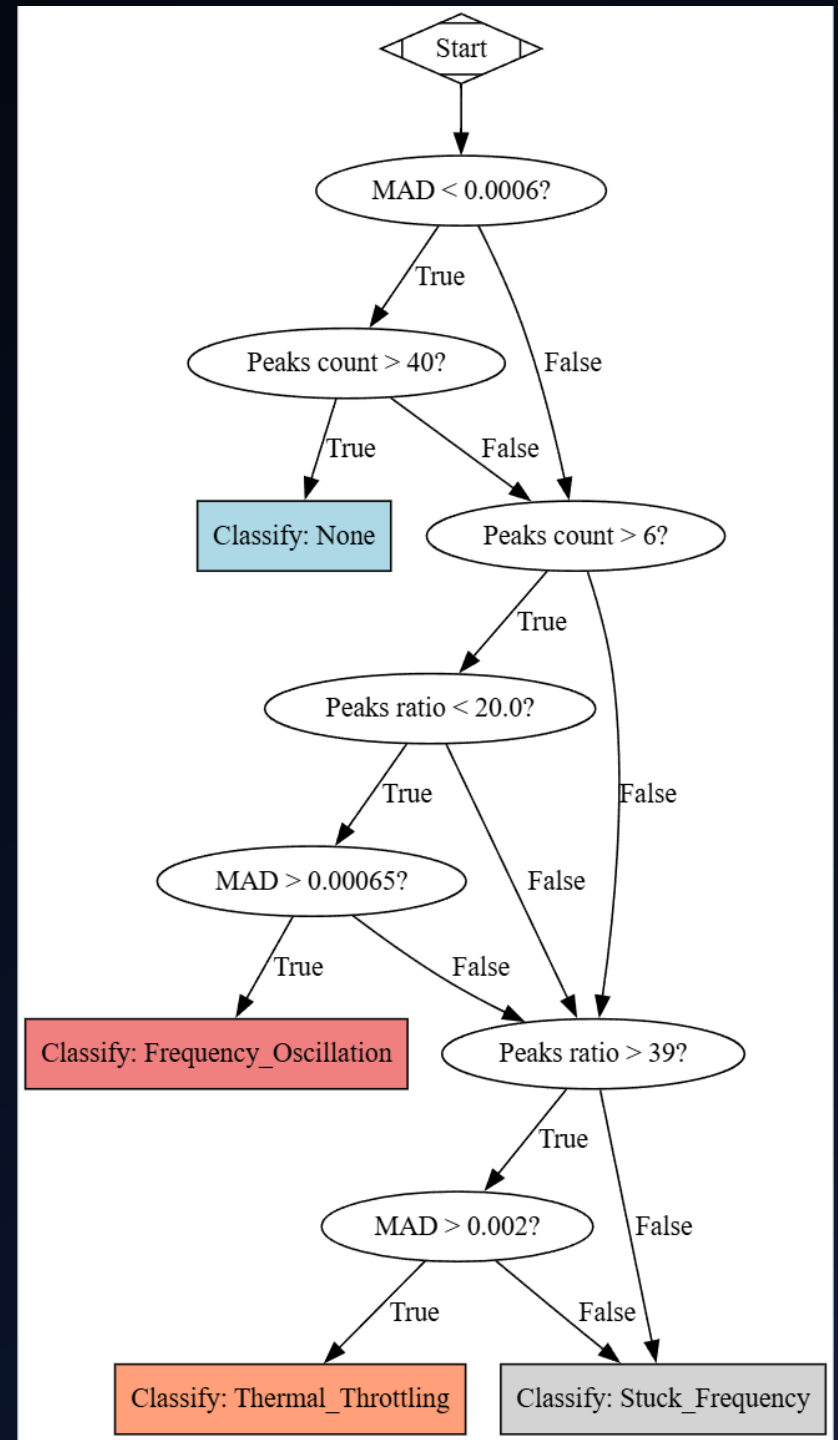
Visualizing the algorithm

If MAD is less than 0.0006 and there are more than 40 peaks, no fail.

If above are false and there are more than 6 peaks, the two peaks ratio is less than 20, and the MAD is more than 0.00065, there is a Frequency Oscillation fail.

If the above are false and the peaks ratio is more than 39, and the MAD is more than 0.002, fail is Thermal Throttling.

If the above are false, fail is Stuck Frequency.



Why did this work?

- No fail had consistently low MAD value, with lots of relative peaks
- Frequency oscillation had a higher MAD due to larger deviations, but more relative peaks than others
- Thermal throttling and Stuck Frequency runs had consistently higher two peaks ratios, as the changes were instantaneous and significant relative to rest of run

CPU Failure Classification (Overall Acc: 96.60%)

True Class	Frequency_Oscillation	None	Stuck_Frequency	Thermal_Throttling
	242	1	7	
		248	2	
	14	10	226	
Predicted Class				250
	Frequency_Oscillation	None	Stuck_Frequency	Thermal_Throttling

Classification was done on 1000 runs of generated data, split evenly between 4 fail modes. Diagonal means predicted=actual.

Thoughts and next steps

Takeaways:

- Very successful overall, classification is over 95% on majority of runs
- MAD feature was surprisingly robust, reliably separating no fail runs

Next steps:

- Using different CPU data (temperature, voltage, cache hits/misses)
- Expanding the features to predict time location of fails
- Trying to prevent false passes (11 total)



End

QUESTIONS?