OFDM peak suppression using time-rotated impulse approximators

Eleanor Clifford, Beep Boop Troop

Introduction

OFDM symbols, owing to their white noise characteristics, have a very high peak to average power ratio in the time domain. This makes them difficult to transmit at high power, inviting problems of noise. This whitepaper describes a method of partially suppressing the peaks using frequency bins which are unused for information.

This was intentionally performed in the absence of research (for reasons of fun and challenge) and is therefore almost definitely not the optimal method. But it works! And achieves 40% peak reduction! Probably even more with further tuning.

A quick brief on obvious methods which don't work

(There are several more things I tried, these are just the most obvious ones)

1. Starting with an impulse train that counters the peaks, and then frequency filtering it. The resulting sincs end up much too wide and nothing but frustration is achieved.
2. Anything involving optimisation on the whole block (unless you're okay with your transmission taking 1000 times longer)

Working Method

1. Precompute the best approximation of an impulse that you can, using only the frequency bins outside of the data range. Other functions would work, but the key here is that an impulse allows each peak suppression to be independent, since it dies quickly in time. The one I am using is in Figure 1, but it is far from optimal.
2. Split your non-suppressed OFDM block into small regions surrounding the peaks.
3. Consider a few time-rotated impulse approximatiors with peaks in the aforementioned regions (Use significantly fewer impulses than the number of samples you have taken, or you will overfit).
4. Fit a linear combination of the time-rotated peaks to the inverse of the region you are trying to suppress, and add to the original signal. Note that you can solve each region independently, since the impulse should die down near the other peaks.
5. ???
6. Profit[1]

Results

The choice of peak thresholding, time window size and shape, number of rotated impulse-approximations, and fitting algorithm contributes significantly to the effectiveness of this method. At best so far I have achieved 40% peak reduction (Figure 2), but of course the tradeoff between computation speed and optimality applies.

Some caveats:

[1] The last two steps are just a meme.

The method is prone to overfitting if the parameters are not chosen very carefully. A fair bit of tweaking is required. It's not especially fast either, though this is due to my implementation in python.

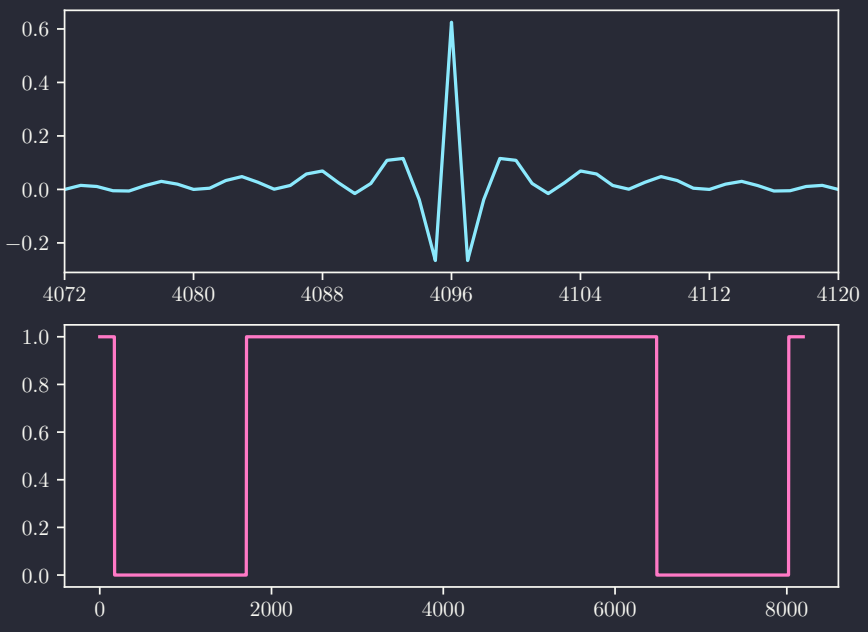

Figure 1: Selected impulse approximator

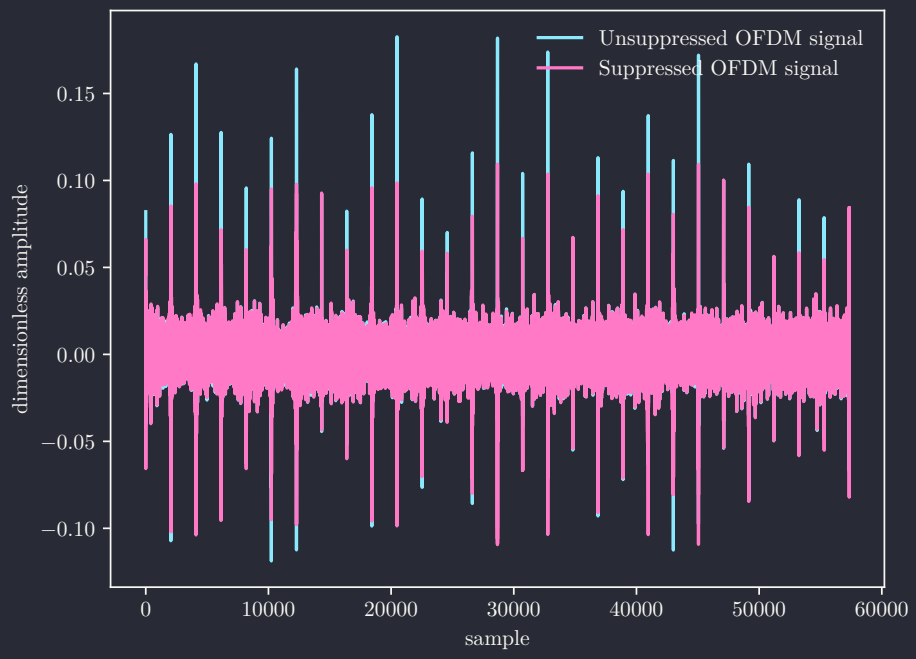

Figure 2: Successful 40% maximum peak reduction