Google Summer of Code 2019 Proposal: Digital Pre-Distortion (DPD) for GNU Radio

Tucker Reinhardt

tbr1@rice.edu

Rice University

1 Objective

Power amplifiers (PA) are a necessary element for signal broadcasting, however device physics tells us that power amplifiers can never be perfectly linear. Power amplifiers and other non-linear components can introduce spectral leakage which is regulated by the FCC to avoid aliasing between frequency channels. Typically PAs have a linear region towards the lower end of their power rating, however restricting input signals to this linear region would expensive due to the significant loss in power efficiency and extra expense of higher power rating PA. The open source software suite GNU Radio is currently lacking a core block to deal with this issue, which is what this project aims to fix via implementation of a generalized Digital Pre-Distortion (DPD) block.

2 Background

2.1 Digital Pre-Distortion

Power amplifiers are imperfect in the sense that scaling the input will not always linearly scale the output. As the input is increased the scaling factor will inevitably decay, and increasing this linearity through higher quality amplifiers is very expensive. Thus one must either limit the input signal to contain the signal to the linear region of the power amplifier, or live with the effects. **Digital Pre-Distortion** is the technique if dealing with these effects by distorting the input such that after going through the PA the output will be as if the original input was multiplied by a linear gain factor. However, what function the input should be distorted by is not known, thus any DPD technique must learn this function. The scheme I plan to implement starts with the concept of a memory polynomial.

2.2 Memory Polynomial

The predistorter which distorts the input signal accordingly is typically a nonlinear function that acts as an inverse of the PAs transfer function. However, this function is not known, thus some combination of memory modeling with polynomials must be used to learn it. One of the most common configurations is the Parallel-Hammerstein model, also known as a memory polynomial.Below is a figure showing a single branch of such a memory polynomial with a memory depth of N, accounting for memory effects in the polynomial function which we are trying to model.

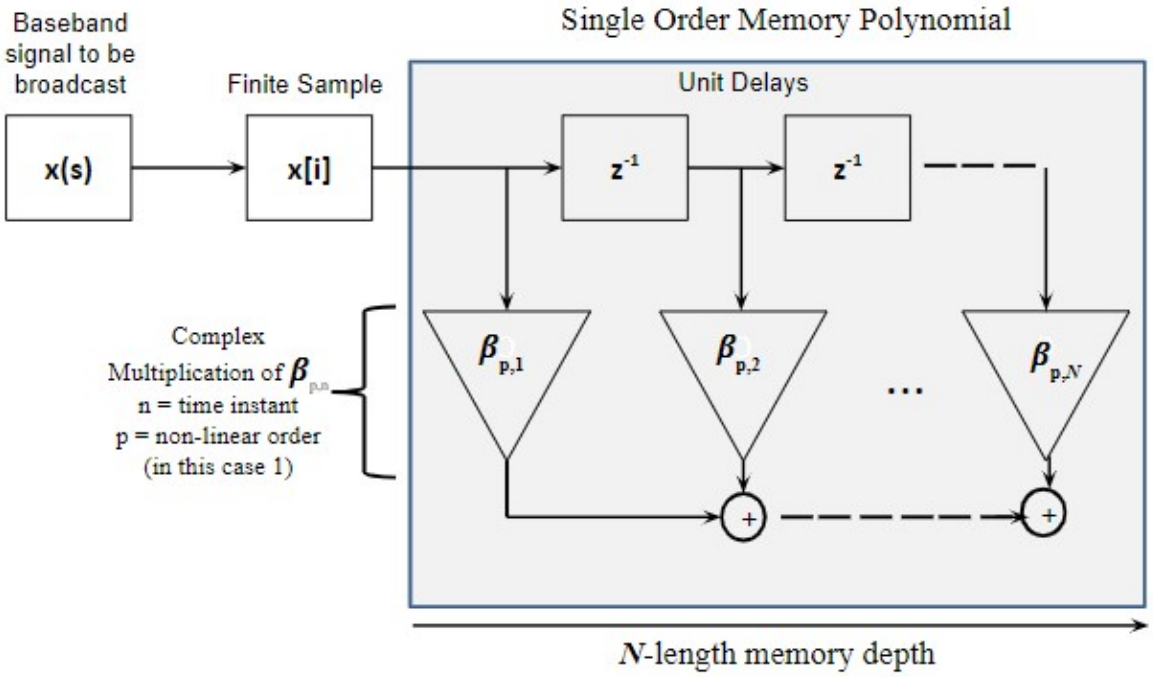

Figure 1: Single Polynomial branch of a Memory Polynomial

Then the "parallel" portion of the model can be thought of as having multiple polynomial "branches" followed by finite-impulse-response (FIR) filters to account for the multiple non-linear orders of polynomial being modelled.

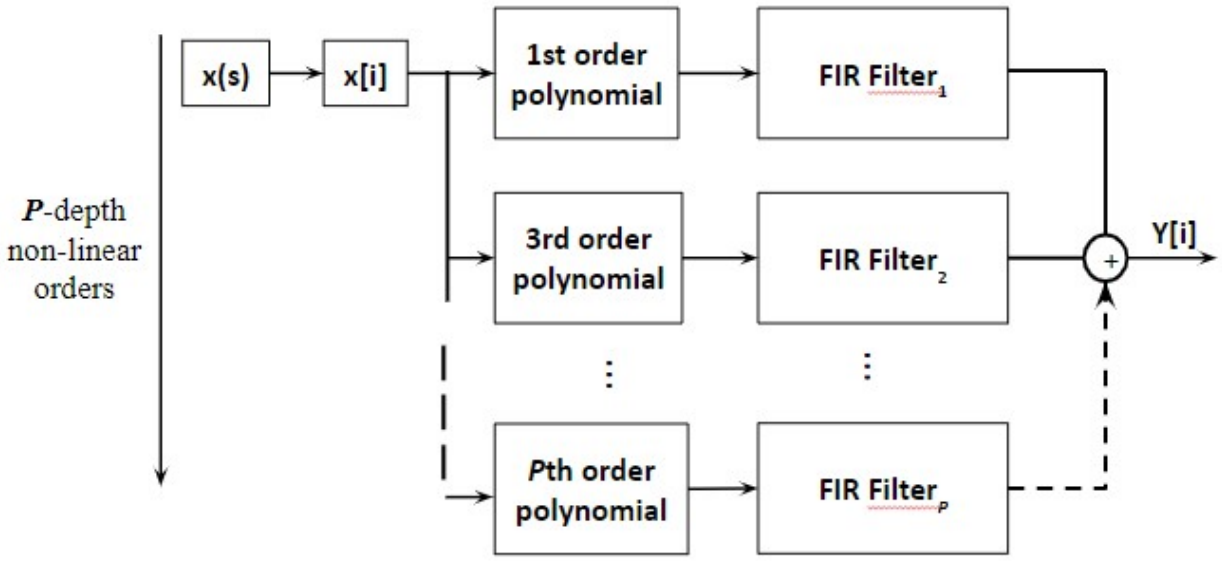

Figure 2: Parallel-Hammerstein Structure of a Memory Polynomial

This is also shown mathematically in equation 1. Here, $x(n)$ is a baseband signal to be broadcast, and $y(n)$ is the output of the memory polynomial based predistorter. P is the maximum polynomial degree considered, and M is the maximum memory depth considered. The symbols i and j index the polynomial degree and depth respectively. In most systems, only odd degrees are considered since the odd degrees contribute to in-band distortions, so i can be restricted to only index even numbers. The coefficient β is a complex scalar.

$$y(n) = \sum_{i=0}^{P-1} \sum_{j=0}^{M-1} \beta_{i,j} x(n-j)|x(n-j)|^i \tag{1}$$

2.3 Indirect Learning Architecture

Now with the memory polynomial technique of modelling a function to predistort by, there is still fundamental problem with designing a predistorter that the ideal input to the PA is unknown. Hence, we can not create a DPD model from some sort of fit of input/output data. One popular way to overcome this is an Indirect Learning Architecture (ILA) shown in Figure 3.

In an ILA, the output of the PA, $y(n)$ is fed into a postdistorter that is equivalent to the predistorter. If the PA output is the ideal, linear output that we desire, then after we remove the gain, G, the input to the training distorter would be equal to the input of the predistorer, $x(n)$. The output of these blocks, $z(n)$ and $\hat{z}(n)$, should therefore be equal and the error would be zero.

This relationship can be leaverged for the learning of a predistorter. For the error to be zero, $\hat{z}(n) = z(n)$. For a use of the PA where we have the output, $y(n)$, this gives us an input/output relationship around the training distorter. We can use this to perform a fit to a DPD structure such as a memory polynomial through a technique such as least squares which will be discussed in the following section.

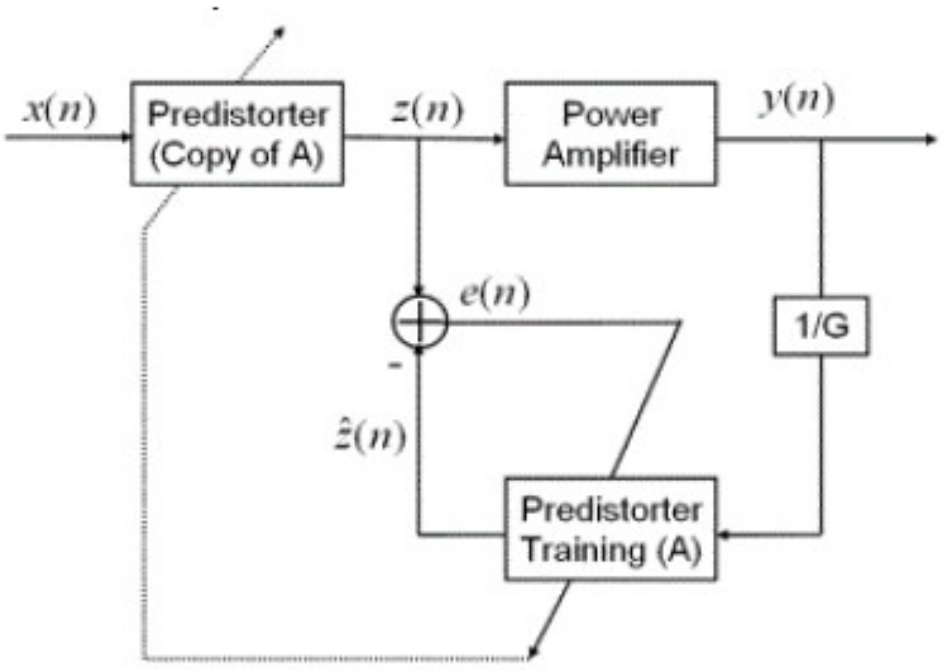

Figure 3: Indirect Learning System Schematic

2.4 Least Squares learning of Coefficients

Although the indirect learning architecture from the previous section provides an intuitive way to perform learning, the actual learning algorithm may be independent of the architecture. One common way to do this is by performing a least-squares fit.

One major advantage of the memory polynomial DPD structure is that the output is linear with respect to the coefficients, β. To illustrate this, let us consider the signal $x(n)$ to be a vector of length N. An equivalent representation of the memory polynomial would then be as follow where for brevity, we consider only up to polynomial degree $P = 3$ and memory depth $M = 2$.

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} x(0) & 0 & x(0)|x(0)|^2 & 0 \\ x(1) & x(0) & x(1)|x(1)|^2 & x(0)|x(0)|^2 \\ \vdots & \vdots & \vdots & \vdots \\ x(N-1) & x(N-2) & x(N-1)|x(N-1)|^2 & x(N-2)|x(N-2)|^2 \end{bmatrix} \begin{bmatrix} \beta_{1,1} \\ \beta_{1,2} \\ \beta_{3,1} \\ \beta_{3,2} \end{bmatrix} \tag{2}$$

This system of equations is overdetermined as there are N equations for four unknowns, thus there is no exact solution. Instead, we choose the β that minimizes the sum of the square residuals.This is known as the least squares solution. In essence, we want to choose the β such that it is as close as possible to the desired output of the system, y. This is shown below.

$$\beta = \arg\min_{\beta} ||y - X\beta||_2 \tag{3}$$

This has the following solution.

$$\beta = (X^H X)^{-1} X^H y \tag{4}$$

We can therefore use this formula over a block of samples to compute DPD coefficients to use in a memory polynomial.

3 Deliverables

Digital Pre-Distortion is not yet a standard part of GNU Radio. This project aims to introduce a new, well documented, module allowing for basic DPD usage in any project.

- **Implement Memory Polynomial Model:** The first element to be implemented is the memory polynomial which will be used to model the function which distorts the input signal. Initially I plan to start with making a single memory polynomial branch with degree one, before moving on to building out the complete Parallel Hammerstein model.
- **Implement Indirect learning DPD system:** With the Memory Polynomial up and running I will then be able to start building the indirect learning architecture which leverages the memory polynomial to be able to model the function
- **Implement Indirect learning DPD system:** The final step will be to implement a least squares fit algorithm which fits the memory polynomial coefficients to minimize the difference between the expected and actual outputs over a large number of samples.
- **Documentation:** I will keep all code readable and well documented so that the final DPD code block for GNU Radio is easy to use and understand.

4 Timeline

I am a sophomore at Rice University studying Electrical Engineering and Computer Science. There is a group working on DPD here at Rice, who sparked my interest in this project - and will serve as great resources for me should this project be approved. Furthermore they will meet with me weekly to discuss the project and help me with any issues.

- Mid May: Further familiarity with GNU Radio structure, module development, and DPD.
- End of May: Initial single degree implementation of a PA model
- Beginning of June: Memoryless Parallel Hammerstein memory polynomial model
- End of June: Indirect Learning Architecture
- Beginning of July: Buffer for any issues with previous items, else move on
- End of July: Least Squares fit learning
- Beginning of August: Final bug fixes
- Mid August: Final Evaluation: submission of final code

5 About Me

I really want the world to know: Cyberspectrum is the best spectrum. Other than that, I have always been interested in computers since I was a kid who quickly outstripped my parent's technological abilities when I learned how to ctrl-alt-delete when a program froze on the screen. One of my goals for my University education is to gain an electron to executable understanding of how computers operate. This project was brought to my attention by a group working on DPD here at Rice, which quickly gained my interest as a great opportunity to merge the skills of signal processing and programming that I have been learning in my two programs of study EE and CS respectively. I have worked extensively with Python and to a lesser extent C through our coursework here at Rice, and have also taken two introductory courses in Signal Processing as well as Linear Algebra. These courses together form the basis of theory behind this project, and getting to also use and improve my programming skills is another bonus. Thus I think working on this project this summer would be a great opportunity for me, and I am confident that I can deliver a great feature for GNU Radio.

6 Conclusion

A DPD module would be a great addition to an already extensive set of signal processing tools in the GNU Radio suite. It would allow users to correct the imperfections introduced to their systems by power amplifiers and other non-linear components that they may not have been aware of. The spectral leakage introduced can be very problematic and is regulated by the FCC, thus accounting for it is essential to any project.