ADSP Final Project

Power Amplifier Characterization with DPD for Reduced Signal Distortion

※Program Flow

Topic1: PA

Ⅰ. PA's impairment

In the fifth-generation (5G) mobile communication system, efficient use of spectrum resources is always an urgent issue due to the massive connections with high volume data transmission. High efficient modulation, e.g., 256 QAM-OFDM, has been employed for the 5G scenarios. As the trade-off of the spectrum efficiency, these modulation types require ultra-high linearity to maintain transmitting quality. However, the wideband envelope-varying signals with high peak power to average power ratio (PAPR) excite the nonlinear impairments of PA. Thus, a design for PA linearization from a nice-to-have to a must-have. The measured PA input/output data's information are listed below:

PA_input: [81920×1 complex double]

PA_output: [81920×1 complex double]

rbnum: 106

rbstart: 0

OFDM_type: 'cp'

Mod_scheme: 'qpsk'

Pow_mat: 28.9690

CBW: 20000000

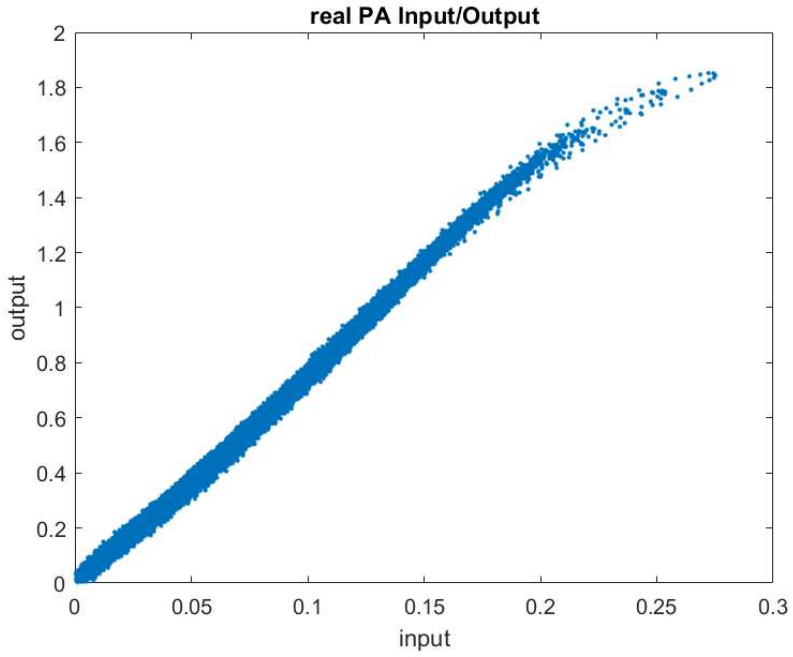
SCS: 15

band: 4101

1. Visualize PA Data to see the impairments

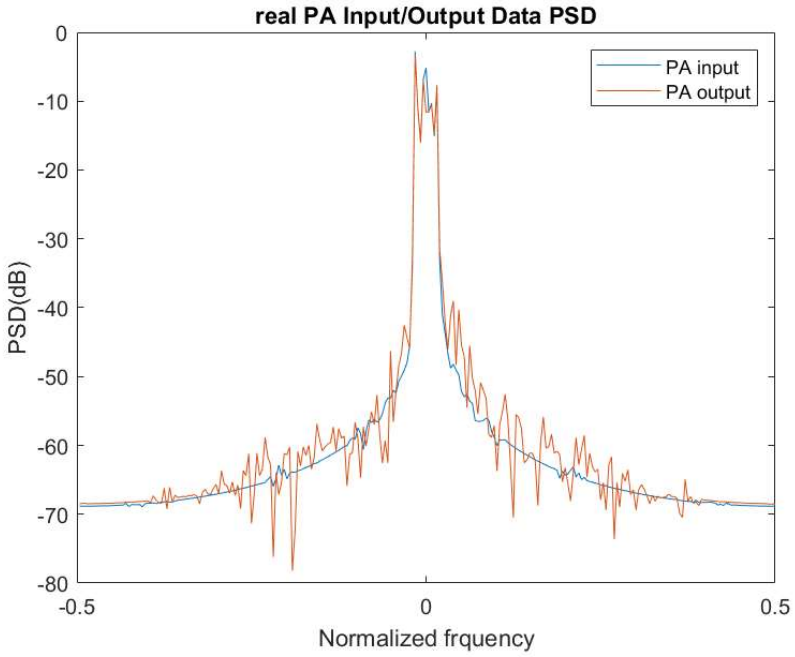
*The following figure plots the magnitude characteristic of the PA: The bending curve of the real PA for large values of input signal results from **PA nonlinearity**, and scattering around a straight line representing the gain results from **memory effects** (data points dispersion).*

These two impairments cause errors of data transmission to increase.



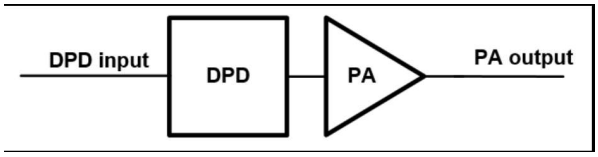
*The following figure plots the PSD of the PA input/output: PA output PSD is suffered from **out-band spectral regrowth**. This effect would interference the signal transmission of adjacent bands and then reduce transmission data rate.*

*Originally, PA output PSD is larger than PA input PSD. We perform the **complex gain normalization** to let PA output and PA input have the same gain level. by this trick the output PSD seems to fit the input PSD, and we can figure out the out-band spectral regrowth.*



In Conclusion, when PA input voltage is high, the nonlinearity and memory effect of PA output is even worse, causing the power efficiency reduces.

*Therefore, the goal of **digital predistortion**(DPD) is to reduce the PA nonlinearity, memory effect and out-band spectral regrowth so that PA can maintain the power efficiency. DPD is put before the PA to make the transmission signal predistorted so the signal fed to PA can combat the impairment discussed above.*



Ⅱ.Simulate PA Model by Memoryless Polynomial

We try to simulate the nonlinearity and memory effects by polynomial models. We first try memoryless polynomial with different polynomial order sets:

$$y(t) = \sum_{k \in \mathcal{K}} b_k z(t) |z(t)|^{k-1}$$

where y(t) is PA output, z(t) is PA input, k is polynomial order, b_k *is PA coefficient.*

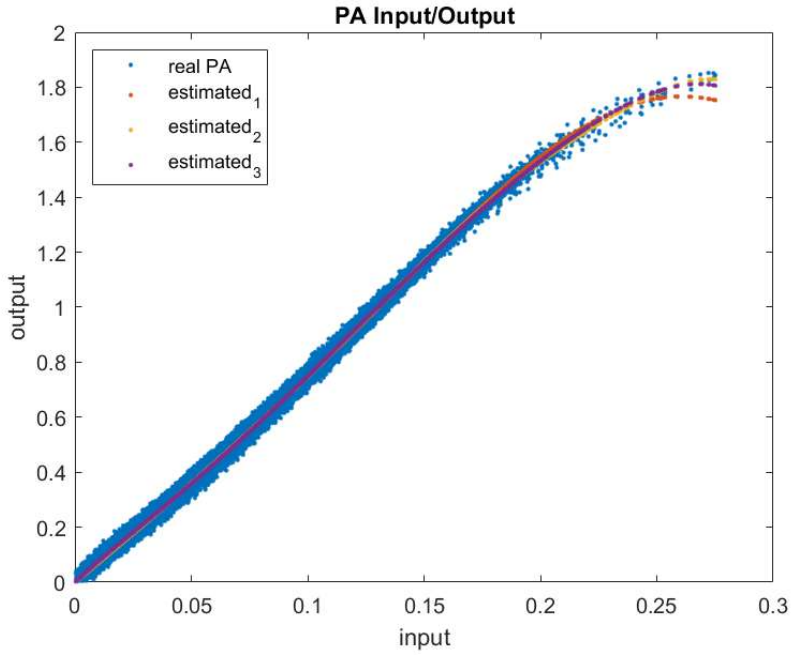
*The function `fit_memory_poly_model` use **Least Square**(LS) to estimate b_k. Then we use **Normalized Mean Square Error**(NMSE) to give a quantitative measure of model performance.*

$$\text{NMSE (dB)} = 10\log_{10}\left[\frac{\sum_{n=0}^{N-1}|y(n)-\hat{y}(n)|^2}{\sum_{n=0}^{N-1}|y(n)|^2}\right]$$

polynimial order set	{1,3,5}	{1,3,5,7,9,11}	{1,2,3,4,5,6}
NMSE(dB)	-28.2982	-28.3917	-28.4284

Apparently, higher polynomial order gives less model error. Further, a paper states that including even-order terms(with the same number of terms) can reduced model errors. But the result shows no difference.

From the following figure, we see that memoryless polynomial can model the PA's nonlinearity. However, the memory effect(data points dispersion) can't be modeled by memoryless polynomial, due to it has no memory.



Ⅲ.Simulate PA Model by Memory Polynomial

Secondly, We try memory polynomial with different polynomial order sets, and with different memory lengths.

$$y(n) = \sum_{k\in\mathcal{K}}\sum_{p=0}^{P-1} b_{kp}\, z(n-p)\,|z(n-p)|^{k-1}$$

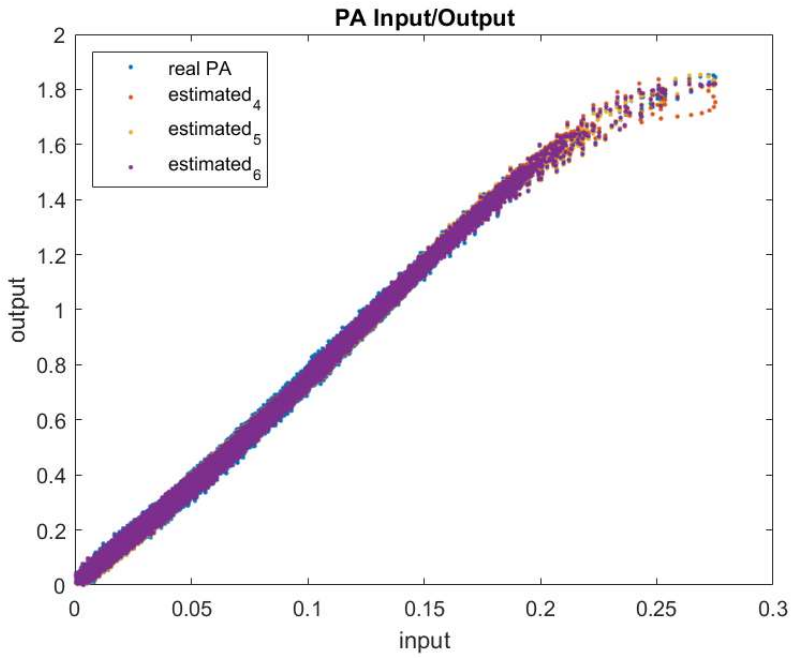
where P is memory length.

*Again, we use LS to estimate b_{kp}. Then use NMSE to compare model error. Since this model has memory, y(n) starts from y(P-1) because z(n-p) is undefined for z(n), n < 0. This issue is called **time alignment**.*

1. keep the same memory length, try different polynomial order sets

With memory length fixed to 3, We get:

polynimial order set	{1,3,5}	{1,3,5,7,9,11}	{1,2,3,4,5,6}
NMSE(dB)	-35.5882	-36.1806	-36.4371

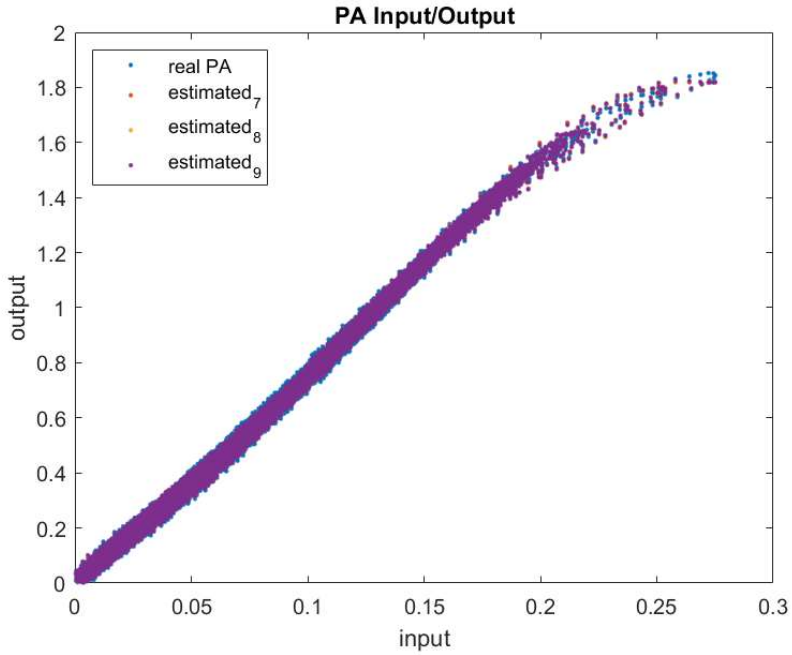


2. keep the same polynomial order set ,try different memory lengths

With polynomial order set fixed to {1,2,3,4,5,6}, We get:

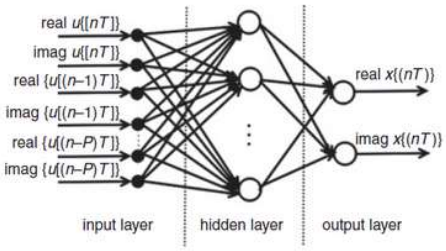
memory length	3	6	9
NMSE(dB)	-36.4371	-36.4499	-36.4524

It seems that with higher polynomial order set and higher memory length, the model error is almost the same(about -36 dB). Increasing polynomial order and memory length doesn't help a lot.



Ⅳ.Simulate PA Model by Neural Network using MATLAB

*Finally we simulate the PA model by **Neural Network**(NN). NN is very capable of nonlinear data fitting problem. NN input is PA input data, NN target is PA output data. Because all data the data are complex, we need to separate data to real part and imagine part. Considering PA's memory effects, we arrange NN inputs with two delay lines. Therefore the number of neuron of the input layer =*



*The back propagation training function is **Levenberg-Marquardt(LM)** backpropagation. It combines gradient descent method with Newton method and is specifically designed to minimize sum-of-square error functions. Therefore it is very suitable for our nonlinear data fitting problem.*

We just use one hidden layer and set the number of neurons of the hidden layer to 30. The performance of just one hidden layer is good enough for LM backpropagation algorithm. Adding more hidden layer, or more number of neurons, has almost the same performance.

The details are listed below.

Number of layers	***One hidden layer***
Length of delay line	*10*
Number of neurons	*20 - 30 - 2*
Activation function	*Hyperbolic tangent sigmoid*
Back propagation training function	*Levenberg-Marquardt*
Loss function	*MSE*
Epoch	*1000*

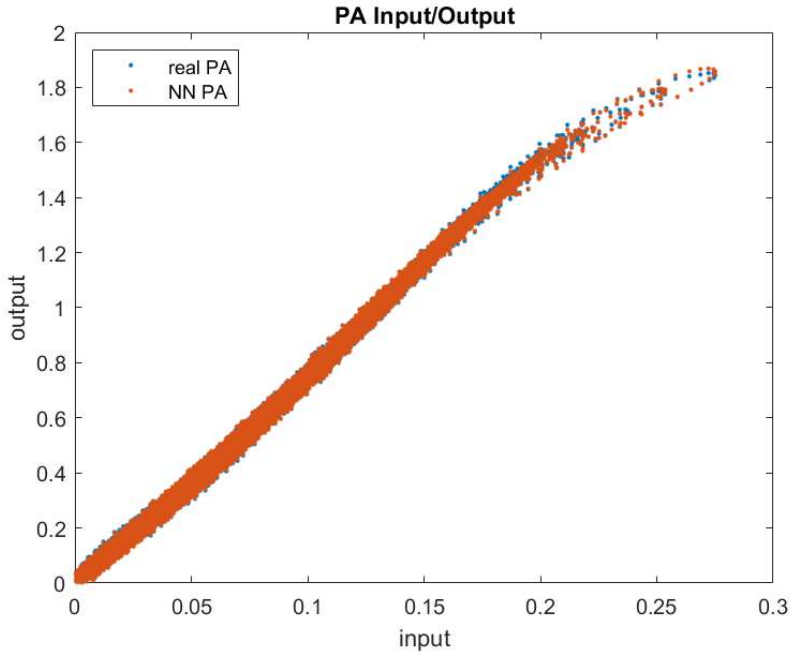
1. NN PA Performance

NMSE(dB)	-42.2405

NN PA model's NMSE is about 6 dB lower than LS PA model. Although it has excellent performance, the training process is very time-consuming (about 30mins once). In the other hand, LS training consumes just few seconds.

Another Disadvantage by using NN is we need huge amount of data. If the training data is insufficient, NN's performance will be worse than LS.

Because NN has the best performance, we will use NN PA model for all the following simulation.



Topic2: DPD

Ⅰ.Simulate DPD Model by Memory Polynomial

The DPD model can also be modeled as memory polynomial:

$$z(n) = \sum_{l \in \mathcal{L}} \sum_{q=0}^{Q-1} a_{lq}\, x(n-q)\, |x(n-q)|^{l-1}$$

where z(n) is DPD output, x(n) is DPD input, l is polynomial order, a_{lq} is DPD coefficient, Q is memory length.

Here we set to train LS DPD.

1. train the DPD coefficient

When DPD is in training phase, DPD input is the PA output data, and DPD output needs to fit the PA input data. In other words, we first train a postdistorter without any predistorter, then the postdistorter is used as a predistorter and the postdistorter is removed. This method is called ***Indirect Learning****.*

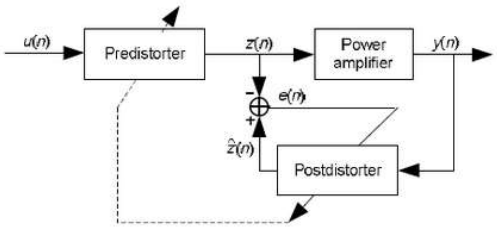
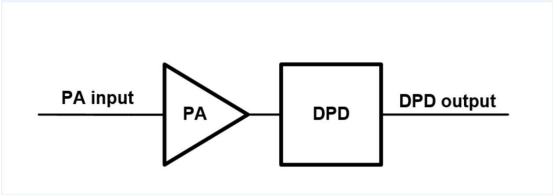


Fig. 2. Indirect learning architecture.

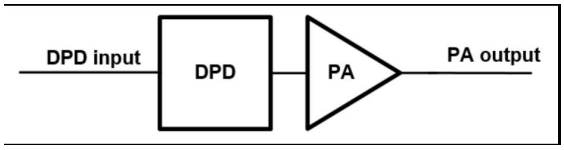
Before training the DPD , the PA output data needs do complex gain normalization to let DPD have a unit gain. Otherwise when DPD and PA are in testing phase, DPD would reduce signal power and let PA have no power gain.



Again we use LS to train a_{lq}.

2. test the DPD by combine DPD with PA

After DPD training is completed, Combine DPD with PA to test the DPD performance.



3. DPD performance

*The **raw NMSE** compares original PA output with original PA input. This gives a quantitative measure of linearity. The **train NMSE**(training error) compares PA input with DPD output when in training. The **test NMSE**(testing error) compares PA output with DPD input when in testing.*

The result shows that test NMSE is about 26 dB lower than raw NMSE. This means that PA has better linearity with DPD added.

raw NMSE	train NMSE	test NMSE
-10.7155	-36.5569	-35.975

(unit: dB)

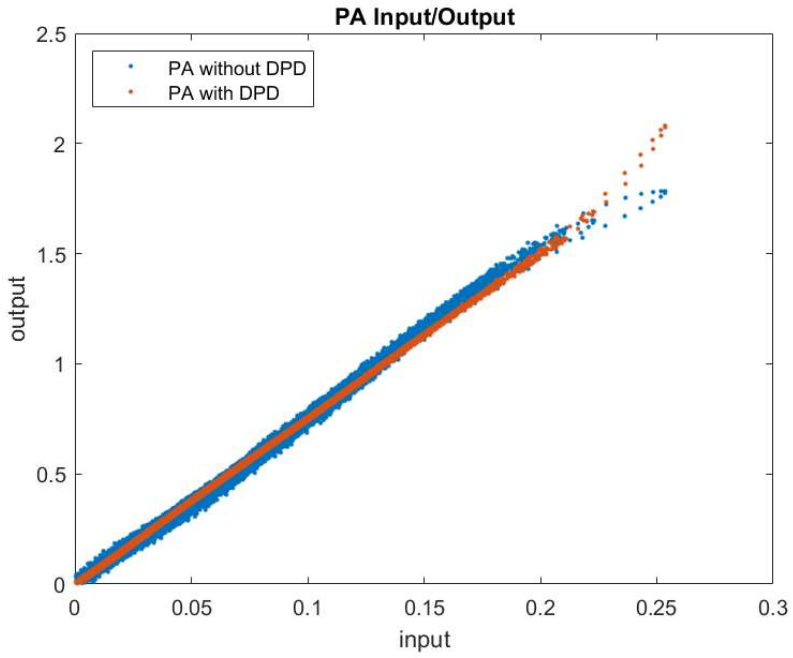
The first figure shows that with DPD, PA nonlinearity is reduced. The data distribution is more like a straight line.

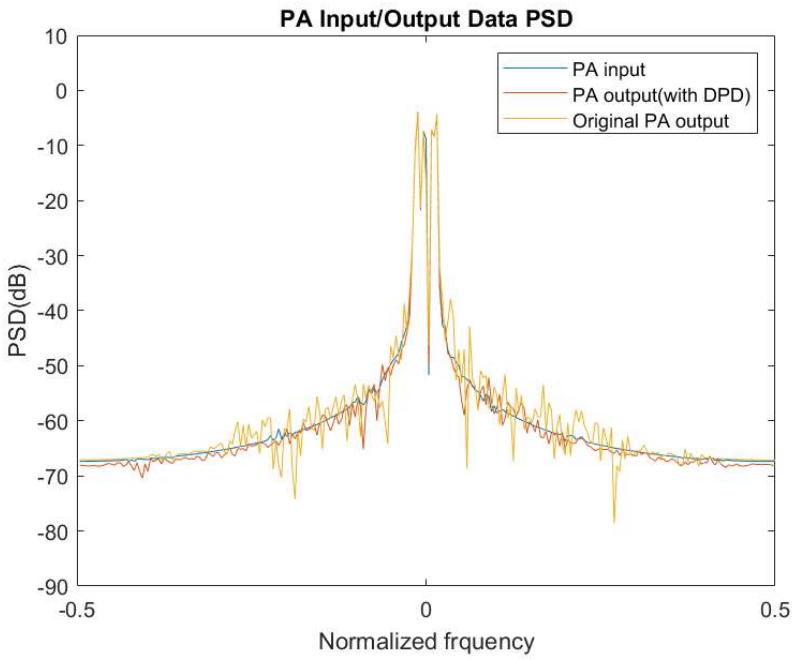
The second figure shows PA output PSD with and without DPD. To give a quantitative measure of out-band spectral regrowth, we use ***adjacent channel power ratio****(ACPR):* $10\log_{10}\left(\frac{\text{Power}_{\text{outband}}}{\text{Power}_{\text{inband}}}\right)(\text{dB})$

raw ACPR	test ACPR
-31.9279	-35.8614

(unit: dB)

The result shows that test ACPR is about 4 dB lower than raw ACPR. This means that PA has suppression on out-band spectral regrowth with DPD added.





Ⅱ.Neural Network - based DPD using MATLAB

We can also use NN to simulate DPD. As we simulate PA by NN, we do the same thing for DPD.

By indirect learning method, the NN input now is PA output data, NN target is PA input data. The back propagation training function is still Levenberg-Marquardt backpropagation. We also set input delay line to combat memory effects.

As in NN PA, we just use one hidden layer and set the number of neurons of the hidden layer to 40. The performance of just one hidden layer is good enough for LM backpropagation algorithm. Adding more hidden layer, or more number of neurons, has almost the same performance.

The details are listed below.

Number of layers	***One hidden layer***
Length of delay line	*3*
Number of neurons	*6 - 40- 2*
Activation function	*Hyperbolic tangent sigmoid*

Back propagation training function	*Levenberg-Marquardt*
Loss function	*MSE*
Epoch	*1000*

1. DPD Performance

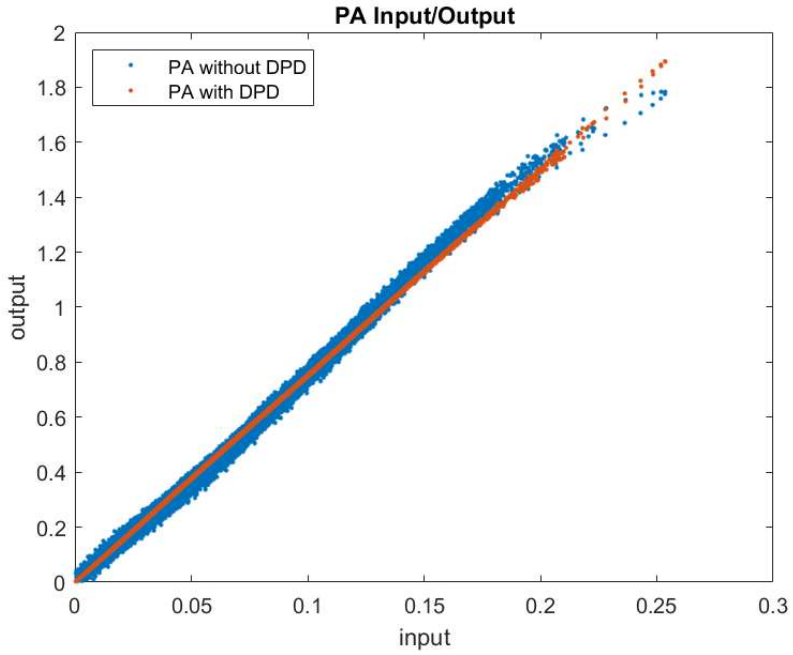
raw NMSE	train NMSE	test NMSE
-10.7155	-42.978	-41.9679

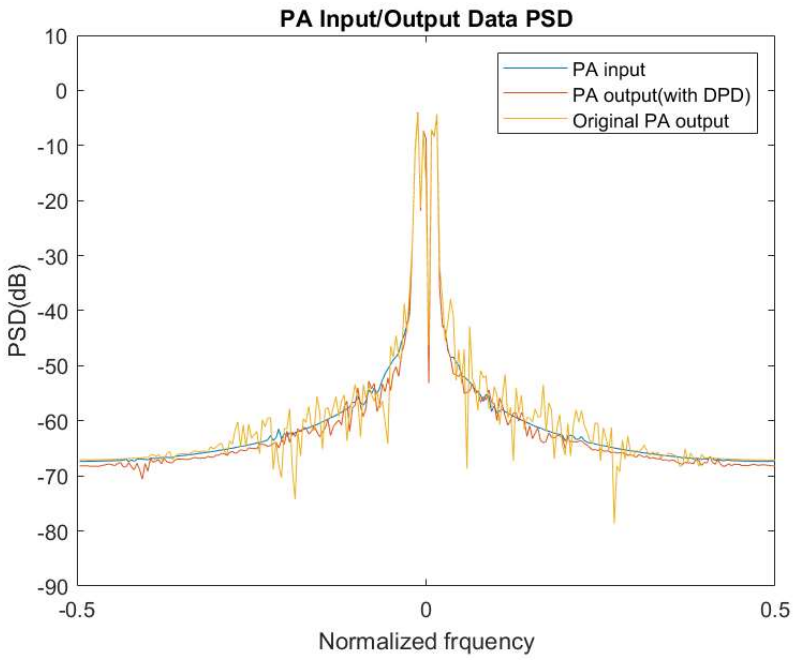
(unit: dB)

raw ACPR	test ACPR
-31.9279	-36.543

(unit: dB)

Just like we train NN PA model, NN DPD model has better performance(both NMSE and ACPR) than LS DPD model. But the training process is also very time-consuming(about 15 mins).





Ⅲ.Neural Network - based DPD by using Pytorch

Although one hidden layer NN DPD already has better performance than LS DPD, we still want to try two or more layer NN, or use other training function(optimizer) to seek other possibility to get more better performance. Here we change to Pytorch to build a multilayer NN because MATLAB can't use GPU to accelerate training process.

The details are listed below.

Number of layers	***3 hidden layer***
Length of delay line	*3*
Number of neurons	*6 - 100 – 100 -100 -2*
Activation function	*ReLU*
Back propagation training function(optimizer)	*Adam*
Loss function	*MSE*

Epoch	*10000*
Batch size	*4096*
Learning rate	*1e-4*

1. DPD Performance

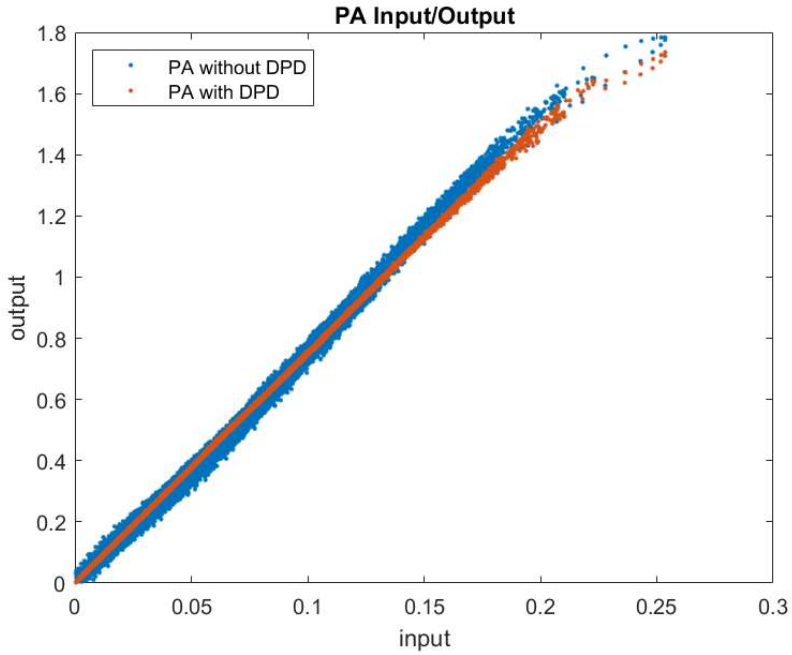
raw NMSE	train NMSE	test NMSE
-10.7155	-40.3307	-39.2752

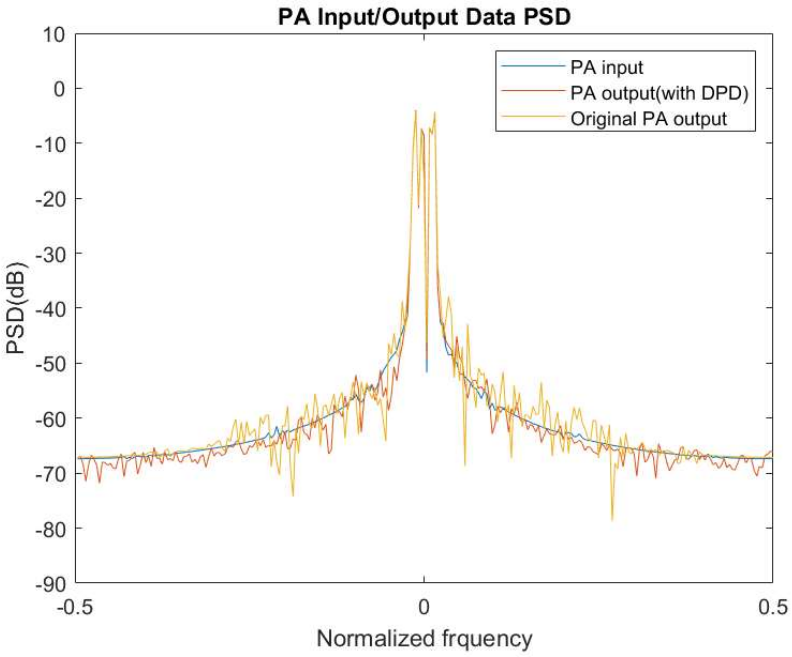
raw ACPR	test ACPR
-31.9279	-35.8709

In consequence, no matter how hard we tried, by adding more layers or adding more neurons, the multilayer NN's performance can't exceed the one hidden layer NN.

After our investigation, we find LM algorithm is specially suitable for data fitting problem. However, optimizers provided from Pytorch such as Adam are for general usage, its performance is not as good as LM algorithm.(Pytorch doesn't provide LM)

If we compare Adam with LM, Adam training is super faster than LM, but it would need a lot more layers or neurons to achieve better performance as like LM.





Topic3: Conclusion

Ⅰ. Method Comparison

We should compare traditional polynomial DPD with NN DPD by their hardware complexity:

	NN DPD	***Polynomial DPD***
Multiplier		
adder		

n: number of neurons

k: number of layers

p: polynomial order

m: memory length

Generally NN needs more resource for calculation. But if we just use one hidden layer NN and few neurons, we can get better performance with almost the same complexity. In other words, NN could achieve lower complexity with specific architecture.

Finally, the following table shows pros and cons comparing polynomial-based DPD with NN-based DPD:

	NN DPD	Poly DPD
pros	1.Excellent performance 2.Can achieve lower complexity through specific spec.	1.Simple to implement 2.Moderate performance 3. Can train with little data 4. Time- conserving for training
cons	1. Time- consuming for training 2. Must collect lots of data	1.Performance is limited 2.Model has less flexibility if data distribution is diverged.

Ⅱ. Conclusion

In this project we build PA and DPD model by traditional polynomial models and by NN models. Our experimental results show NN models for both PA and DPD have better performance(NMSE,ACPR) than polynomial models. Specifically LM algorithm has extremely excellent performance for our nonlinear data fitting problem.

However, considering hardware complexity, NN may need more resources for its complex calculation. Besides, NN training process is very time-consuming and needs sufficient training data.

Furthermore, there are other NN architecture such as RNN or LSTM, but we don't have enough time to explore them. Hence we stop here.

Ⅲ. Reference

1. *Deep Neural Network-Based Digital Predistorter*
2. *Digital predistortion method combining memory polynomial and feed-forward neural network*
3. *A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers*
4. *Effects of Even-Order Nonlinear Terms on Power Amplifier Modeling and Predistortion Linearization*
5. *Comparison of Direct Learning and Indirect Learning Predistortion Architectures*
6. *Digital Predistortion With Advance Delay Neural Network and Comparison With Volterra Derived Models*
7. *A New Volterra Predistorter Based on the Indirect Learning Architecture*