NON-LINEAR SELF-INTERFERENCE CANCELLATION ON BASE OF MIXED NEWTON METHOD

Degtyarev A.[1]**, Bakhurin S.**[2]

[1]MIPT
141701, Russia, Dolgoprudny, Institutsky per., 9
degtyarev.aa@phystech.edu

[2]MPEI
111250, Russia, Moscow, Krasnokazarmennaya 14, build. 1,
BakhurinSA@mpei.ru

Abstract. The article researches solution to the task of self-interference cancellation (SIC) which arises during design of in-band full-duplex (FD) systems. Conventionally self-interference cancellation is conducted in digital domain by means of behavioral models due to the simplicity of implementation and appropriate performance. In addition, the models are usually trained by gradient-based optimizers. However, convergence speed of mentioned optimization approach leaves much to be desired. In this regard we propose to exploit a novel second-order training approach, called Mixed Newton Method. This algorithm uses information about loss function second-order derivatives and, as a result, provides high convergence speed in comparison to the traditional first-order methods.

Keywords: *Second order methods, convergence speed, complex-valued hessian, in-band full-duplex (IBFD) systems, self-interference cancellation (SIC), behavioral modelling.*

Introduction. Recent years researches in Internet-of-Things inspired noticeable results in communication research and development. One of the key research directions in telecommunication field is the task of multiple users service with the high data rates [**6g_trends**]. Among the variety of approaches to solving this issue, in-band full-duplex technology is a promising method, which doubles spectral usage efficiency. The idea of current approach is based on the frequency bandwidth sharing between transmitter and receiver [**analog_sic_1**, **fd_transceiver**, **analog_sic_2**]. Recently, current technology was considered to be unrealizable due to the high power self-interference coupling at the receiver‘s input. Thus, self-interference cancellation is the crucial problem in the full-duplex systems design.

In FD systems interference occurs due to the simultaneous work of transmitter and receiver. Thus transmitted signal must be subtracted from the signal at the receiver part. However, these approach is unsuitable for SIC task due to the fact of transmitter signal distortion by SI transmitter-receiver channel, analog components impairments such as in-phase and quadrature phase imbalance of mixer [**linear_sic**], phase noise [**phase_noise**], ADC/DAC non-linear distortion [**impairments**], and the most powerful duplexer and power amplifier non-linearities [**behav_model_ghann**]. Therefore, the main task of interference suppression could be stated as an identification of interference. In other words one is required to build suitable interference model.

Self-interference is usually suppressed in several steps. Firstly, analog RF-cancellation is applied to the receiver input signal, then it processed in digital domain [**behav_model**]. In current paper we consider active digital cancellation.

In modern SIC research there are two main investigation directions. The first one is related to the adaptation of behavioral models, such as Wiener-Hammerstein model [**wiener_hammerst**, **spline_sic**]. The second direction is introduced by Complex and Real-Valued Neural Network training [**ffnn**, **advanced_ml**, **hlnn**].

All mentioned models are trained by gradient-based methods. In current article we propose to exploit Mixed Newton method (MNM) [**mixed_newton**] for Hammerstein model training. We provide performance and convergence speed comparison with conventional first order methods.

System model. In current section we provide the simplified scheme of FD-transceiver with active digital cancellation module. After that, behavioral interference model, which is used in current simulations, is introduced.

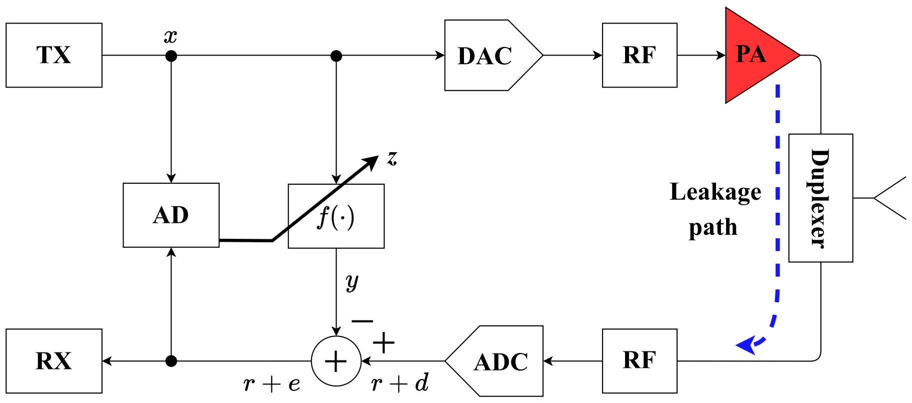

Figure 1: Self-interference identification scheme.

Self-interference cancellation issue. Conventional interference identification block scheme is shown in Fig. 1. Transmitted signal samples x_n are transformed to analog domain, then propagated through the nonlinear analog components, such as analog power amplifier (PA). Resulting nonlinear interference is distorted during propagation through the leakage path between transmitter and receiver. Thus, useful receiver signal is mixed with interference and transformed back to the digital samples $r_n + d_n$, where r_n – useful signal samples, d_n – interference samples.

In order to get rid of parasitic interference d_n, adaptive signal processing block AD in Fig. 1 adjusts parameters of the nonlinear block $f = f(x_n, \boldsymbol{z})$, where $\boldsymbol{z}$ – nonlinear block parameters. Since useful signal r_n has no correlation with interference, then adaptive processing block is supposed to minimize SI d_n. As a result, receiver samples are introduced by $r_n + e_n$, where $e_n = d_n - y_n = d_n - f(x_n, \boldsymbol{z})$.

SI minimization quality criteria is represented by Mean Square Error (MSE) [**behav_model**], since it is in fact algorithm error energy:

$$\mathrm{MSE} = \sum_{n=0}^{N-1} e_n^* e_n = \boldsymbol{e}^H \boldsymbol{e}, \tag{1}$$

where $\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{y}$, $\boldsymbol{e} \in \mathbb{C}^{N\times 1}$,$\boldsymbol{d} \in \mathbb{C}^{N\times 1}$,$\boldsymbol{y} \in \mathbb{C}^{N\times 1}$, N – signal length, $(\cdot)^H$ – hermitian conjugation operator.

Interference model. Generally, behavioral modelling [**behav_model**] implies consideration of model which reflects physical processes of an appropriate phenomena. For instance, according to the Fig. 1

in SIC-task signal firstly distorted by nonlinearity in the transmitter path. After that, signal is affected by transmitter-receiver leakage path. Consequently, Hammerstein model [**behav_model**], which is shown in Fig. 2 properly depicts behavior of parasitic interference. Note, that Hammerstein model is a simplification of Wiener-Hammerstein model [**behav_model**] for the case of memoryless non-linear effects. NL-block in Fig. 2 could be represented by various types of series, such as polynomials,

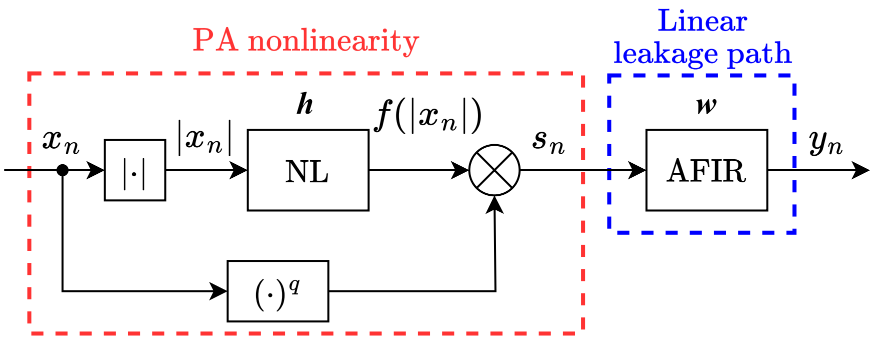

Figure 2: Hammerstein model.

trigonometric polynomials, splines etc. Nonetheless, in current work we have chosen 1 - dimensional spline-based polynomial [**lut_dpd**] as a nonlinearity due to its low computational cost and easy hardware implementation. NL-block has parameters $\boldsymbol{h} \in \mathbb{C}^{P\times 1}$, P – polynomial order. AFIR-block shows adaptive FIR [**haykin**] with parameters $\boldsymbol{w} \in \mathbb{C}^{L\times 1}$, L – FIR order. Power q in practice is chosen according to the intermodulation order. Since transmitter and receiver work in the same frequency band, then in current research $q = 1$. Thus, model output sample could be derived as:

$$y_n = \sum_{m=-D}^{D} w_m \sum_{k=0}^{P-1} h_k x_{n-m} NL(|x_{n-m}|), \tag{2}$$

where $L = 2D + 1$, $NL(\cdot)$ – spline-based basis function [**lut_dpd**], which is described in section .

Mixed Newton general approach. Mixed Newton algorithm is a second-order method, designed for the training of complex-valued structures by minimization of MSE (1). In addition, error vector $e_n = e_n(x_n, \boldsymbol{z})$ (1) must be holomorphic with respect to the whole model parameters $\boldsymbol{z}$. In this case MNM has a property of repulsion from saddle points [**mixed_newton**].

Let operator $D_{\boldsymbol{z}}(\cdot)$ be a differentiation operator with model parameters vector $\boldsymbol{z} \in \mathbb{C}^{K\times 1}$, K– number of whole model parameters:

$$D_{\boldsymbol{z}}(\cdot) = \begin{pmatrix} \frac{\partial}{\partial z_0} & \frac{\partial}{\partial z_1} & \cdots & \frac{\partial}{\partial z_{K-1}} \end{pmatrix} \tag{3}$$

Then, consider $D_{\boldsymbol{z}}J \in \mathbb{C}^{1\times K}$ as a scalar real-valued function $J \in \mathbb{R}$ derivative with respect to the whole model parameters. Thus, first-order differential of the function $J = J(\boldsymbol{z}, \boldsymbol{z}^*)$ could be derived as [**complex_deriv**]:

$$dJ = D_{\boldsymbol{z}}J d\boldsymbol{z} + D_{\boldsymbol{z}^*}J d\boldsymbol{z}^*, \tag{4}$$

where $D_{\boldsymbol{z}}J \in \mathbb{C}^{1\times K}$, $D_{\boldsymbol{z}^*}J \in \mathbb{C}^{1\times K}$, $d\boldsymbol{z} \in \mathbb{C}^{K\times 1}$, $d\boldsymbol{z}^* \in \mathbb{C}^{K\times 1}$.

Second-order derivative of scalar real-valued function J is described as follows:

$$D_z(D_zJ)^T = H_{z,z}J \in \mathbb{C}^{K\times K} \text{– Hessian matrix,} \tag{5}$$

where $(\cdot)^T$ – transpose operator.

Using notations mentioned above, second-order differential of the function J can be described in following way [**complex_deriv**]:

$$d^2J = \begin{pmatrix} dz^T & d(z^*)^T \end{pmatrix} \begin{pmatrix} H_{z,z}J & H_{z^*,z}J \\ H_{z,z^*}J & H_{z^*,z^*}J \end{pmatrix} \begin{pmatrix} dz \\ dz^* \end{pmatrix} \tag{6}$$

Since MNM algorithm uses only mixed hessian $H_{z^*,z}J$ for the model parameters training [**mixed_newton**], then general view of its difference equation is as follows:

$$z_{k+1} = z_k - \mu(H_{z^*,z}J)^{-1}(D_{z^*}J)^T, \tag{7}$$

where J – MSE function, defined in (1).

Let us consider error vector $\boldsymbol{e} = \boldsymbol{e}(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{C}^{N\times 1}$ as a holomorphic function with respect to $\boldsymbol{z}$. Then we can calculate MSE gradient and mixed hessian (7). MSE gradient is derived as follows:

$$\begin{aligned} (D_{z^*}J)^T &= (D_{z^*}(\boldsymbol{e}^H\boldsymbol{e}))^T = (D_{z^*}(\boldsymbol{e}^T\boldsymbol{e}^*))^T = \\ &= (\boldsymbol{e}^T D_{z^*}\boldsymbol{e}^*)^T = (\boldsymbol{e}^T(D_z\boldsymbol{e})^*)^T = (D_z\boldsymbol{e})^H\boldsymbol{e}. \end{aligned} \tag{8}$$

Since $\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{z})$, where $\boldsymbol{d}$ doesn‘t depend on model parameters $\boldsymbol{z}$, then MSE gradient is as following:

$$(D_{z^*}J)^T = -(D_z\boldsymbol{y})^H\boldsymbol{e}, \tag{9}$$

where $D_z\boldsymbol{y} \in \mathbb{C}^{N\times K}$ – model output Jacobian with respect to the whole model parameters.

In the same time, according to the notation (5) and gradient expression (9), mixed hessian could be calculated in current way:

$$H_{z^*,z}J = D_z(D_{z^*}J)^T = -D_z((D_z\boldsymbol{y})^H\boldsymbol{e}). \tag{10}$$

Since $\boldsymbol{y}$ is a holomorphic with respect to the model parameters, then mixed hessian is introduced by the following expression:

$$H_{z^*,z}J = (D_z\boldsymbol{y})^H D_z\boldsymbol{y}. \tag{11}$$

Thus, considering expressions (9), (11) and (7), Mixed Newton Method difference equation could represented as below:

$$z_{k+1} = z_k - \mu((D_z\boldsymbol{y})^H D_z\boldsymbol{y})^{-1}(D_z\boldsymbol{y})^H\boldsymbol{e}. \tag{12}$$

Thus, algorithm step requires calculation of an error $\boldsymbol{e}$, model output Jacobian with respect to the model parameters $D_z\boldsymbol{y}$ and hessian inversion.

Mixed Newton for Hammerstein model. According to the difference equation (12), Mixed Newton requires calculation of model output Jacobian with respect to the whole model parameters $\boldsymbol{z}$. Hammerstein model is 2-layer model, which includes memoryless nonlinearity with parameters $\boldsymbol{h} \in \mathbb{C}^{P\times 1}$ and FIR with parameters $\boldsymbol{w} \in \mathbb{C}^{L\times 1}$. Consider Hammerstein model parameters vector and conjugated parameters vector as follows:

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{w} \end{pmatrix}, \boldsymbol{z}^* = \begin{pmatrix} \boldsymbol{h}^* \\ \boldsymbol{w}^* \end{pmatrix} \in \mathbb{C}^{(P+L)\times 1}. \tag{13}$$

Then Hammerstein model output Jacobian with respect to the parameters is derived as below:

$$D_{\boldsymbol{z}}\boldsymbol{y} = \begin{pmatrix} D_{\boldsymbol{h}}\boldsymbol{y} & D_{\boldsymbol{w}}\boldsymbol{y} \end{pmatrix} \in \mathbb{C}^{N\times(P+L)}. \tag{14}$$

Thus we need to calculate model output vector derivative for the nonlinearity layer and FIR.

According to adaptive filtering theory [**haykin**] FIR output could be derived through the state matrix $\boldsymbol{U}$:

$$\boldsymbol{y} = \boldsymbol{U}\boldsymbol{w}, \boldsymbol{U} \in \mathbb{C}^{N\times L}. \tag{15}$$

State matrix has following structure [**haykin**]:

$$\boldsymbol{U} = \begin{pmatrix} s_D & \cdots & s_1 & s_0 & 0 & \cdots & 0 \\ s_{D+1} & \cdots & s_2 & s_1 & s_0 & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & s_{N-1} & s_{N-2} & s_{N-3} & \cdots & s_{N-D-2} \\ 0 & \cdots & 0 & s_{N-1} & s_{N-2} & \cdots & s_{N-D-1} \end{pmatrix}.$$

Here s_n – FIR input sample in accordance with Fig. 2. In correspondance with expression (2) FIR input sample could be shown as:

$$s_n = \sum_{k=0}^{P-1} h_k x_n NL(|x_n|). \tag{16}$$

Since FIR state matrix doesn‘t depend on FIR parameters (16), then, according to (15), model output derivative with respect to the FIR parameters is:

$$D_{\boldsymbol{w}}\boldsymbol{y} = D_{\boldsymbol{w}}(\boldsymbol{U}\boldsymbol{w}) = \boldsymbol{U}. \tag{17}$$

Now, let us turn to the nonlinearity mathematical representation. Note, that spline polynomials operate as piecewise linear interpolation. In addition, this type of polynomial is efficient in terms of resources required for hardware implementation, that is why it has been chosen for behavioral modelling.

Let us represent Hammerstein model output in matrix-vector multiplication manner through the spline-nonlinearity parameters vector. Consider FIR input vector as:

$$\boldsymbol{s} = \boldsymbol{V}\boldsymbol{h}, \boldsymbol{V} \in \mathbb{C}^{N\times P}. \tag{18}$$

Nonlinearity state matrix $\boldsymbol{V}$ for spline-based NL is shown below [**lut_dpd**]:

$$\boldsymbol{V} = \begin{pmatrix} 0 & 0 & \cdots & x_0(1-\Delta_0) & x_0\Delta_0 & \cdots & 0 \\ 0 & \cdots & x_1(1-\Delta_1) & x_1\Delta_1 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & \cdots & x_k(1-\Delta_k) & x_k\Delta_k & \cdots & 0 \\ & & & \cdots & & & \end{pmatrix}.$$

Each row of state matrix $\boldsymbol{V}$ include only 2 non-zero elements: $x_k(1-\Delta_k)$ and $x_k\Delta_k$, where x_k - Hammerstein model input sample, $\Delta_k = ||x_k| - \lfloor |x_k| \rfloor|$, where $\lfloor\cdot\rfloor$ – floor operator.

Current state matrix could be written through its columns:

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}_0 & \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_{P-1} \end{pmatrix}, \tag{19}$$

then FIR input vector (18) is as below:

$$\boldsymbol{s} = h_0\boldsymbol{v}_0 + h_1\boldsymbol{v}_1 + \cdots + h_{P-1}\boldsymbol{v}_{P-1}. \tag{20}$$

Thus, Hammerstein model output vector could be presented as:

$$\boldsymbol{y} = \mathrm{conv}_w(\boldsymbol{s}), \tag{21}$$

where $\mathrm{conv}_w(\cdot)$ – 1-dimensional linear convolution operator. Using convolution linearity property, equations (20) and (21) we obtain Hammerstein model output through matrix-vactor multiplication:

$$\begin{aligned}\boldsymbol{y} = \mathrm{conv}_w(h_0\boldsymbol{v}_0 + \cdots + h_{P-1}\boldsymbol{v}_{P-1}) = h_0\mathrm{conv}_w(\boldsymbol{v}_0) + \cdots + h_{P-1}\mathrm{conv}_w(\boldsymbol{v}_{P-1}) = \\ = \begin{pmatrix}\mathrm{conv}_w(\boldsymbol{v}_0) & \cdots & \mathrm{conv}_w(\boldsymbol{v}_{P-1})\end{pmatrix}\boldsymbol{h} \equiv \boldsymbol{V}_f\boldsymbol{h}.\end{aligned} \tag{22}$$

Therefore, Hammerstein model output vector is derived through the product of nonlinearity state matrix, filtered across columns, and nonlinearity parameters vector. Note, that matrix $\boldsymbol{V}_f$ doesn‘t depend on spline polynomial parameters $\boldsymbol{h}$.

As a result, using equation (22), model output derivative with respect to nonlinearity parameters could be written as:

$$D_{\boldsymbol{h}}\boldsymbol{y} = D_{\boldsymbol{h}}(\boldsymbol{V}_f\boldsymbol{h}) = \boldsymbol{V}_f. \tag{23}$$

Finally, using expressions for model output Jacobians (14), (17), (23), Mixed Newton difference equation (12) could be rewritten as:

$$z_{k+1} = z_k + \mu \begin{pmatrix}\boldsymbol{V}_f^H\boldsymbol{V}_f & \boldsymbol{V}_f^H\boldsymbol{U} \\ \boldsymbol{U}^H\boldsymbol{V}_f & \boldsymbol{U}^H\boldsymbol{U}\end{pmatrix}^{-1} \begin{pmatrix}\boldsymbol{V}_f^H \\ \boldsymbol{U}^H\end{pmatrix}\boldsymbol{e}. \tag{24}$$

Note, that algorithm (24) can be modified by additional step size μ tuning. In our computations parameter μ is adjusted in accordance with the solution of the one-dimensional optimization problem every training epoch.

Also, notice that Gradient Descent approach for Hammerstein model is represented by the following expression:

$$z_{k+1} = z_k + \mu(D_{z^*}J)^T = z_k + \mu\begin{pmatrix}\boldsymbol{V}_f^H \\ \boldsymbol{U}^H\end{pmatrix}\boldsymbol{e}. \tag{25}$$

Computational complexity of the MNM step (24) is determined by matrices $\boldsymbol{U}$ (17), $\boldsymbol{V}_f$ (23) calculation, gradient $D_{z^*}J$ (25), inverse Hessian-gradient product, calculation and, the most important, mixed Hessian $H_{z^*,z}J$ (11) calculation and inversion.

Signal sample propagation through the spline-based nonlinearity (16) is estimated as $\mathcal{O}(1)$ float-point operations (FLOPs), then whole signal $\boldsymbol{x} \in \mathbb{C}^{N\times 1}$ propagation through nonlinearity and, as a result, matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ calculation require $\mathcal{O}(N)$ calculations. Matrix $\boldsymbol{V}$ (19) filtering could be evaluated as $\mathcal{O}(PLN)$.

Gradient and Hessian calculation requires $\mathcal{O}((P+L)N)$ and $\mathcal{O}((P+L)^2N)$ operations respectively. Inverse Hessian-gradient product complexity is $\mathcal{O}((P+L)^2)$. Hessian inversion is the most complex operation among mentioned. Since mixed Hessian for holomorphic funcitons is hermitian matrix, then matrix inversion could be implemented by means of eigenvalue decomposition (EVD), its complexity is estimated as $\mathcal{O}((P+L)^3)$ [**matrix_comput**].

According to aforementioned, whole algorithm step complexity is evaluated as:

$$\begin{aligned}\chi_{MNM} =&\mathcal{O}(N)+\mathcal{O}(PLN)+\mathcal{O}((P+L)N)+\mathcal{O}((P+L)^2N)+\mathcal{O}((P+L)^2)+\\ &+\mathcal{O}((P+L)^3)=\mathcal{O}((P+L)^3+(P+\ L)^2N).\end{aligned} \tag{26}$$

Thus, Hessian calculation and inversion are the most difficult in terms of FLOPs number.

It can be seen, that gradient descent complexity is sufficiently lower, since it requires only matrices $\boldsymbol{U}$, $\boldsymbol{V}_f$ and gradient calculation:

$$\chi_{grad} = \mathcal{O}(N)+\mathcal{O}(PLN)+\mathcal{O}((P+L)N) = \mathcal{O}(PLN+(P+L)N). \tag{27}$$

Nevertheless, gradient descent steps in real application might be made much faster comparing to MNM steps. As a result, gradient-based methods require sufficiently more steps. Consequently, number of FLOPs for both approaches may probably differ insignificantly.

Numerical Results. The transmitter data is provided by QAM-modulated OFDM signal with 60 MHz pass-band bandwidth. The sample rate is 484 MHz. The whole data consists of $\sim$80000 samples, with 50% is used for training and the rest samples used for testing.

Testbench is shown in fig. 3. Digital transmitter data is sent to the signal generator, where it is transformed to the analog domain and moved to the 1.7 GHz carrier frequency. Then, it propagated through

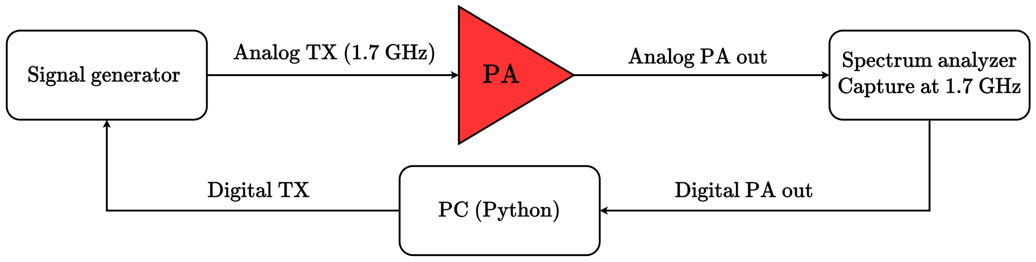

Figure 3: Testbench scheme for signal nonlinear distortion generation

the PA with average output power 20 dBm and captured by means of spectrum analyzer at 1.7 GHz. Captured signal is further sent back to the PC. Leakage path is simulated by digital FIR order 51. Captured PA out is propagated through the generated filter. FIR output signal is considered as the self-interference.

Note, that in real application interference is mixed with useful receiver signal as in Fig. 1. In this case adaptive correction module suppresses only self-interference, due to useful signal might have no correlation with transmitter signal [**haykin**]. Therefore, in our simulations we don‘t take into account useful RX-signal and consider only SI as the receiver path.

For SIC experiments we have chosen Hammerstein model (Fig. 2) with $L = 45$ taps adaptive FIR, 1-dimensional linear-spline polynomial order $P = 8$. In exponentiation branch $q = 1$.

All provided results were made on NVIDIA Tesla V100 GPU with 16 GB RAM.

Cancellation Performance and Convergence Speed. Mixed Newton and BGD are applied to whole training data. In other words, matrices V_f and U in equations (24), (25) are calculated for $\sim$40000 samples.

On the other hand, in case of SGD whole training set is divided into the non-overlapping blocks, which are packed into the batches. Each batch consist of 50 blocks of 200 samples. Thus each epoch model trains on 4 batches. Gradients are calculated for each block in batch and averaged. Batch size and block size were chosen experimentally in terms if the highest performance.

Also, notice that in BGD and MNM model parameters are updated once per epoch, whereas in SGD parameters – once per batch.

In following simulations NMSE is tracked every epoch on train and test data, which is shown in fig. 4, 5.

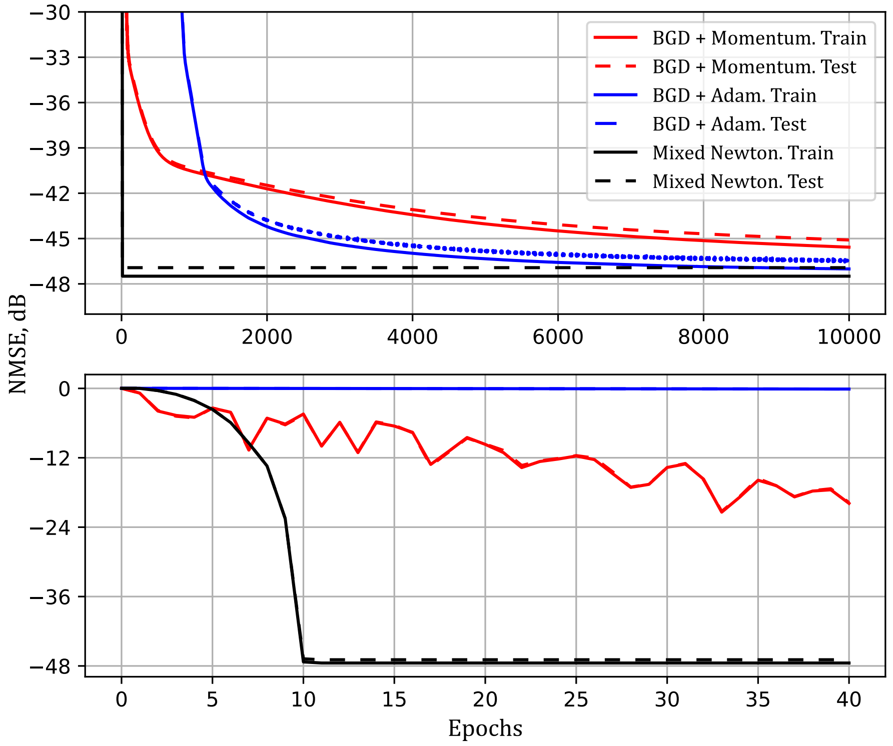

Figure 4: BGD with Momentum, BGD with Adam and MNM learning curves on train and test data sets

Convergence speed comparison of Block Gradient Descent with Momentum and Adam optimizers, and Mixed Newton are shown in Fig 4. MNM requires approximately 30 epochs to achieve final performance, whereas BGD with both optimizers require $\sim$ 10000 epochs to acquire the same NMSE.

Fig. 5 represents learning curves obtained for SGD with Momentum and Adam optimizers and Mixed Newton. It can be seen that SGD also requires approximately 10000 epochs to achieve performance obtained by MNM.

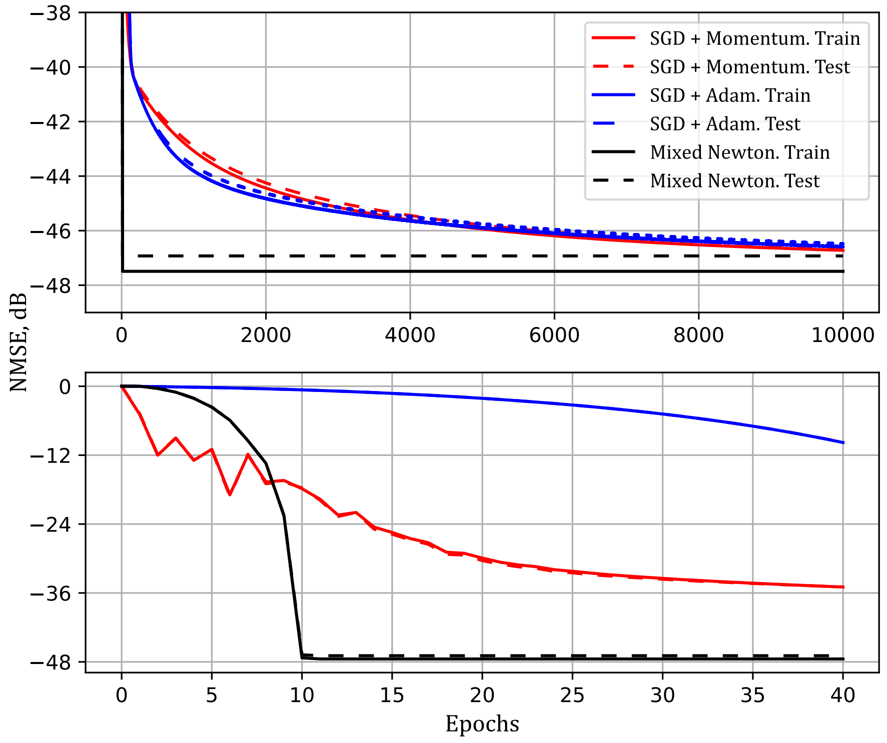

Figure 5: SGD with Momentum, SGD with Adam and MNM learning curves on train and test data sets

Note, that optimizers parameters and learning rate for gradient descent where chosen in the way to achieve the best final performance and the highest convergence speed.

Power spectral densities (PSD) of the initial and suppressed by different methods interference are shown in Fig. 6. PSD figures are built on test data set.

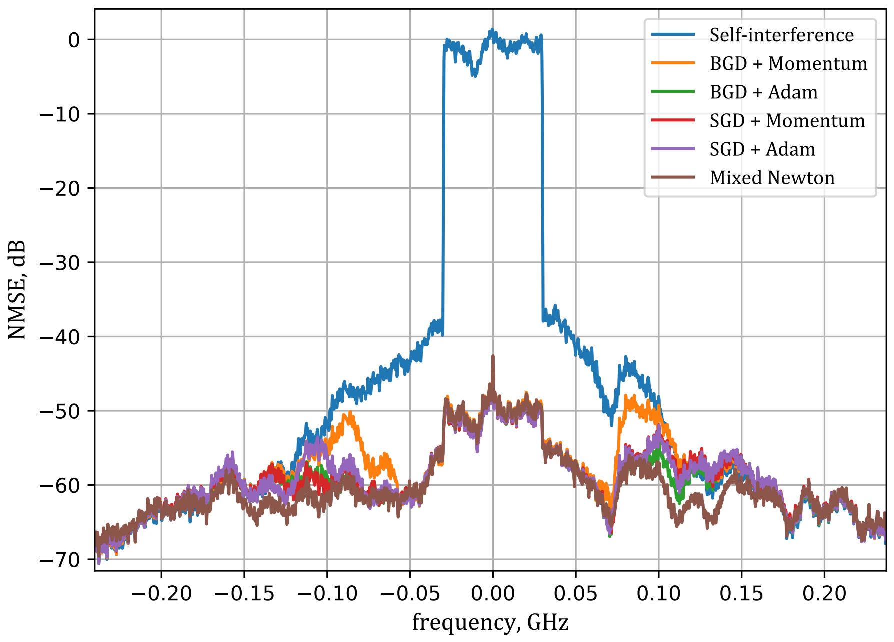

Figure 6: PSD of initial and suppressed interference

Table 1 shows remarkable convergence speed increase of Mixed Newton method comparing to the common gradient methods used in the interference suppression tasks. MNM requires only 30 epochs to achieve comparable performance. Although, provided second order method requires sufficiently less optimization steps, the steps in current simulation ~ 5 times longer. Nevertheless, total time required for training is significantly decreased.

Table 1: Performance and convergence speed comparison

Algorithm	**BGD Moment.**	**BGD Adam**	**SGD Moment.**	**SGD Adam**	**MNM**
Epoch number	10000	10000	10000	10000	30
Time per epoch, 10^{-2} s	3.8	4.0	3.7	4.1	21
Total elapsed tims, s	380	403	386	412	6.2
NMSE, dB	-45.1	-46.7	-46.6	-46.5	-46.9

According to calculations provided in theoretical part, MNM step complexity (26) is high comparing to the gradient-based algorithms (27). Nonetheless number of steps, required for convergence is significantly lower for Mixed Newton (table 1). Thus, MNM is remarkably suitable for offline model performance estimation. Also, Mixed Newton could be exploited for online models training, since in real application it requires mixed Hessian accumulation and, as a result, makes significantly lower training steps comparing to gradient-based methods.

Conclusion. In current paper, we provided a novel method of SI cancellation. The method is strictly derived for Hammerstein model case using matrix differentiation approach. Although, Mixed Newton requires ~ 5 times longer calculation step, total training duration is sufficiently decreased to 30 epochs, instead of 10000 required for conventional first-order optimization methods, to achieve the same interference cancellation level 46.9 dB.

Future work may consider an application of proposed second-order algorithm for training of non-linear interference models, which are introduced by Complex-Valued Neural Networks (CVNNs). Since Mixed Newton implies inverse matrix calculation, it has high memory consumption. Therefore current algorithm, in its original view, could be implemented only for low-complexity NNs and doesn‘t scale to the high-complexity models. Thus, crucial research direction is introduced by modification of MNM in order to minimize Hessian inversion overhead, which, for instance, could be represented by various inverse matrix estimation techniques.