# Neural-Network Based Self-Initializing Algorithm for Multi-Parameter Optimization of High-Speed ADCs

Shrestha Bansal, *Student Member*, *IEEE*, Erfan Ghaderi, *Student Member*, *IEEE*, Chase Puglisi, *Student Member*, *IEEE*, and Subhanshu Gupta, *Senior Member*, *IEEE*

***Abstract*— This work proposes a new automatic model parameter selection approach for determining the optimal configuration of high-speed analog-to-digital converters (ADCs) using a combination of particle swarm optimization (PSO) and stochastic gradient descent (SGD) algorithm. The proposed hybrid method first initializes the PSO algorithm to search for optimal neural-network configuration via the particles moving in finite search space with coarse quantization. Using the PSO estimates, the SGD algorithm then finds the global optimum solution. The global search ability of the PSO algorithm and the local search ability of the SGD are thus exploited to determine an optimal solution that is close to the global optimum with reduced latency. Several experiments were constructed to optimize the non-linearities in Nyquist flash and pipeline ADC datasets to show that the neural networks trained by the PSO-SGD algorithm outperform the random search-based performance optimization. Comparative resource analysis of the proposed algorithm is also conducted against the state-of-the-art that highlights improved latencies and performance with similar area and implementation complexity.**

***Index Terms*— Analog-to-digital converter (ADC), stochastic gradient descent (SGD), particle swarm optimization (PSO), neural-network (NN), artificial intelligence, bias optimization.**

## I. Introduction

Process, voltage and temperature (PVT) variations affect the overall performance of many data converters leading to severe degradation in their performance. Widely used numerical optimization techniques such as gradient descent (GD) [1]–[6] are targeted for a particular ADC architecture and require some modification (minor or major) in the implementation, to be applied to another type of ADC architecture in real-time. An intricate system-level understanding of the ADC and its respective parameters is thus required to perform any kind of optimization. This necessitates the need of a general-purpose ADC optimization technique. Further, there is a need for multi-parameter optimization to overcome non-linearities due to severe correlated effects which warrants the use of a neural network architecture [3], [7]. Unlike existing schemes, NNs can optimize multiple system parameters simultaneously, thus improving the system performance considerably. Enabled by high-speed field-programmable gate arrays (FPGAs), they also have a significantly faster closed loop response [8]. However, empirical methods for model parameter selection and weight matrix determination for NN have been demonstrated mostly using GD algorithm [9], [10] which is easily trapped in local optima during the optimization of system-wide variables (such as signal-to-noise ratio, linearity/dynamic range, gain, offset, timing mismatch, phase noise, and jitter). This not only limits the use of NN optimization but is further expected to get worse in advanced semiconductor nodes with higher mismatches.

Prior works in generic ADC optimization have used look-up-table (LUT) and integral non-linearity (INL) based calibration techniques that are not adequate for multi-parameter optimization of high-speed ADCs. In [11], a LUT-based foreground calibration algorithm is proposed to correct memoryless non-linear impairments in the amplifiers and comparators and mismatches in the capacitors in time-interleaved (TI) ADCs. However, the use of one LUT table per TI sub-ADC channel does not sufficiently capture inter-channel dependencies and is further limited by the LUT size. In [12], an INL based black-box calibration mechanism for ADCs with strong input-output discontinuities between the adjacent output codes is proposed. This scheme uses internal ADC signals to estimate static non-linear errors for multi-valued ADCs having strong discontinuities between adjacent codes and overcomes the limitation of histogram based INL calibration which incorrectly captures the non-linear ADC transfer characteristic, thus, leading to miscalibration. This method, however, requires complex matrix multiplications and a large training dataset to achieve accurate calibration resulting in higher latencies.

Architecture-specific calibration techniques have also been proposed for Nyquist ADCs. In [13], background calibration is proposed to correct the non-linearities in a pipeline ADC comprising inter-stage amplifiers, DACs, buffers and switches. It uses a single calibration bit per pipelined stage to detect and correct any INL breaks and harmonic distortion appearing in the sub-range (flash code) from the respective stages. This method is highly effective in mitigating intermodulation components but its implementation is specific to pipeline ADC.

Particle swarm optimization (PSO) is a typical stochastic optimization algorithm that has shown impressive performance for a broad range of parameter optimization in data converters. Proposed by Eberhart [14], PSO mimics the behavior of a flock of birds or fish searching for food. Based on this principle, PSO

S. Bansal, C. Puglisi, E. Ghaderi, and S. Gupta are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, 99164, USA. (shrestha.bansal@wsu.edu). This work was supported in part by the Office of Naval Research under grant #N00014-18-1-2254.

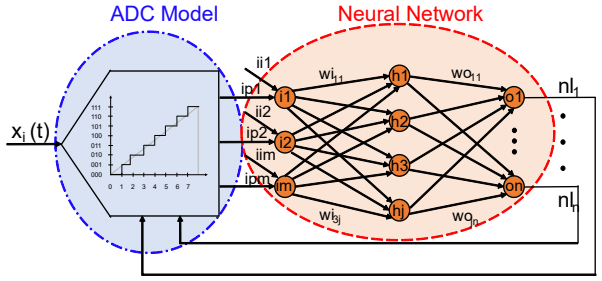


Fig. 1. Proposed black-box multi-parameter optimization.

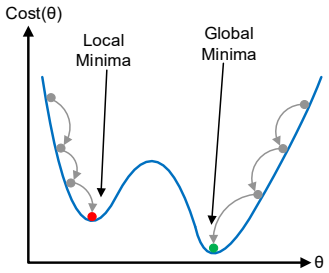


Fig. 2. Optimization using SGD algorithm.

works by randomly spreading out multiple particles in the range of probable solutions and each particle works to obtain the best possible solution. After every iteration, all the particles communicate their respective positions and best solutions they have arrived at to the other particles in the swarm, directing the other particles in the direction of best solution. This algorithm thus works through co-operation and competition amongst the individual particles. Because multiple particles are spread out in the search space, each trying to find an optimum solution, the probability of all the particles getting stuck in local minima simultaneously is negligible and it is almost certain that at least one particle in the swarm would find the global minima. The PSO complexity is independent of the dataset size which makes it readily applicable for optimizing high-speed ADCs with large generated dataset as compared to size dependent SGD. In [15] PSO is implemented off-chip to calibrate the inter-stage gains for a pipelined ADC. The calibration method, however, does not correct any non-linearities or mismatch and thus does not utilize the full potential of PSO. In [16], these challenges are overcome using an additional ADC for calibrating sub-ADC stages hence consuming additional power and area.

We propose a hybrid approach that is computationally efficient and has lower latency compared to the prior calibration techniques. In addition, the proposed technique does not suffer from local minima problems of SGD. The proposed work leverages PSO to narrow down the optimization space, while using GD to converge to the global minima at a much faster rate. This provides us with the flexibility to tune a much larger number of parameters in a smaller time, to get optimal ADC performance. We implement this in a neural-network (NN) with three layers – input layer, hidden layer, and an output layer as shown in Fig. 1. Simulations are demonstrated on a conventional 3-bit flash ADC and a 5-bit pipeline ADC, modelled using the Python programming language with intentional gain and offset errors. In [17], PSO has been used to optimize the time-amplifier linearity and shows a 6.4dB improvement in the pipeline TDC performance, thus demonstrating that PSO has the potential to calibrate a wide set of ADC non-linearities. The input layer of the NN uses time-series data to compute instantaneous errors between the two inputs. This difference is then applied to the hidden layer by multiplying with 8-bit weighted vectors. The hidden layer in this process now applies the weights from the SGD, PSO, and the proposed PSO-SGD algorithm and provides its output to the output layer. Output layer is connected to the bias controls of the non-ideal ADC which will be tuned appropriately to reach the desired transfer function. This completes the feedback loop and is used to achieve the most optimum values in real-time.

Section II briefly describes the PSO and SGD optimization techniques and their use in NN and then presents the proposed PSO in combination with the SGD algorithm for multi-parameter bias optimization with NN in real-time. Section III then presents the simulation results with the ADC datasets followed by the conclusions in section IV.

## II. Proposed NN Based Multi-Parameter Optimization

The training phase in the GD requires considering the depth and the number of hidden neurons which can be very influential in determining the optimization performance [9]. This is further affected by the choice of model parameter configuration setup that initializes learning rate, decay, momentum, dropout rate, and the number of hidden neurons. This consequently affects the weights of the gradient vectors as well as the variation amplitude of the parameters around the optimal bias point, leading to sub-optimal optimization. Despite recent works [10], [18] constructing a variety of empirical methods to improve the performance of NN-based optimization approaches and the use of classical multi-layer perceptron-based classifiers, GD cannot find the global minimum and its search procedure can easily be trapped in local optima as illustrated in Fig. 2. It is thus imperative to select the appropriate learning rate which is further challenged due to the large number of optimization variables in the analog front-end.

Because of the above limitations, PSO-based optimization [19], [20], [21] have replaced GD to solve the multi-parameter optimization problem. However, despite many benefits of PSO, such as low-latency, and improved performance in a multivariate optimization task without being trapped in local minima, the PSO lags behind GD in achieving a similar accuracy with the same hardware complexity [22].

To overcome these limitations, first, the PSO algorithm will be initialized to search for a network configuration that is closer to the global minimum with coarse quantization. The pseudo-code for this is as shown in Fig. 3. This will be followed by GD that trains the NN using the PSO estimates and fine tunes the learning rate to find the global optimum solution. By exploiting the global search ability of the PSO algorithm and the local

a. Evaluate difference between desired ($y_i$(t)) and actual ADC output ($h$(t)) to get mean squared error ($e$)
b. while $e$ > 0.05 * target error
   ➔ Use PSO to optimize the ADC parameters
c. Initialize the starting point of SGD with the global best position of the PSO
d. Use SGD to optimize the ADC parameters

Fig. 3. Pseudo code for optimization.

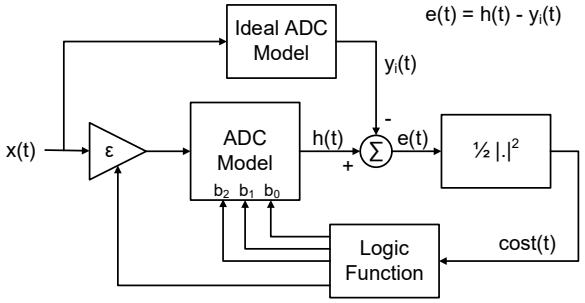


Fig. 4. Implementation of SGD to calibrate flash-ADC.

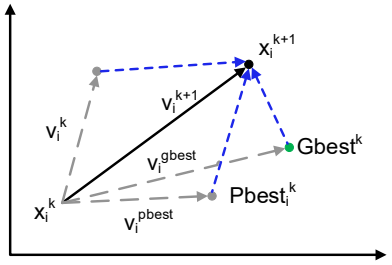


Fig. 5. Optimization using Particle Swarm Optimization (PSO) algorithm

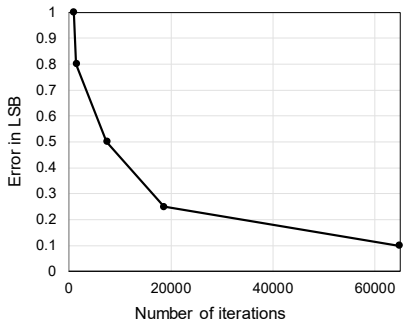


Fig. 6. Error vs number of iterations run in PSO.

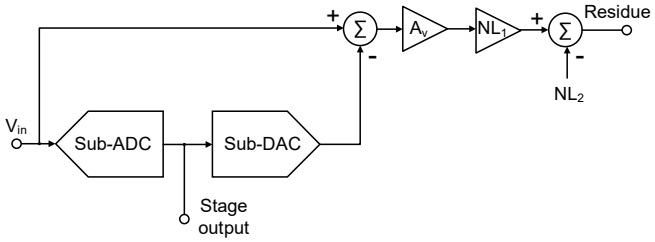


Fig. 7. A 5-bit pipelined ADC model.

search ability of the GD, an optimal solution that is close to the global optimum with reduced latency can be found. We next describe the cost function analysis using GD and PSO used for the proposed optimization algorithm.

### *A. Cost function analysis with Stochastic GD optimization*

A cost function defines the performance of a system based on its expected value and the actual output obtained through the system. GD follows the gradient of this cost function to arrive at the most optimum solution, which is the minimum value the cost function can achieve. The system parameters are updated after computing the mean squared error value of a batch of samples, leading to averaging of the errors, and thus more time for achieving minimum value [23]. Stochastic GD (SGD) computes error value based on one sample point per iteration and updates the system parameters after every single iteration. As a result, the errors are not averaged out and the function achieves minima faster [24]. In the case of high frequency ADCs, the errors in the output bits might average out due to a single bit flipping frequently, so it is of prime importance to preserve the error information of each bit. This makes SGD the obvious choice for such an optimization task. However, as discussed earlier, the SGD algorithm suffers from the same limitation as the GD and can converge to a local minima. The cost function has been defined as the mean square difference between the two concurrent bit values of the two data streams. With a unity amplitude input signal, SGD is then applied in the feedback loop to vary the threshold levels of the non-ideal ADC to achieve the least possible error. The update mechanism of the SGD optimization implemented in Fig. 4 is as follows:

$$s^{t+1}=s^t+2.\mu.e^t.\mathrm{sgn}(x^t) \quad (1)$$

where $s$ represents the cost function, μ controls the accuracy, and sgn($x$) represents the signum function.

### *B. Cost function analysis with PSO optimization*

A similar cost function analysis for PSO is described in (2) and (3). Equation (2) defines the particle velocity ($v_i$). The first term, $w.v^t$ represents the inertia factor accumulated by the particle over the previous iterations. The second term $c1.\mathrm{rand}(0,1).\left\{x^t_{p_{best}} - x^t\right\}$ represents the competition factor driving the particle towards its individual best. The third term $c2.\mathrm{rand}(0,1).\left\{x^t_{g_{best}} - x^t\right\}$ represents the push towards the global best position discovered by any of the particles in the swarm. (3) further defines the displacement ($x_i$) of the particle based on the updated velocity value of the particle, as shown in Fig. 5.

$$v_i^{t+1} = w \cdot v^t + c_1 \cdot \mathrm{rand}(0,1) \cdot \left\{x^t_{p_{best}} - x^t\right\} + c_2 \cdot \mathrm{rand}(0,1) \cdot \left\{x^t_{g_{best}} - x^t\right\} \quad (2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3)$$

where $w$, $c_1$ and $c_2$ are scaling factors for the inertia, competition and co-operation factors respectively. The function rand(x, y) generates a random number between x and y.

### *C. Proposed hybrid PSO and SGD optimization*

To overcome the limitations in PSO and SGD, we propose a hybrid algorithm, that achieves not only lower latencies with significantly lower probability of getting stuck in a local minima, but also achieving higher accuracy than PSO alone.

PSO optimization is done first. Once the mean squared error is reduced to less than 5%, the partially optimized system parameters act as the input to the SGD algorithm which further optimizes the system to achieve an accuracy of less than 0.1%. The empirically determined limits of transitioning from PSO to SGD optimizations are to realize less computational time while also avoiding getting stuck in the local minima. As shown in Fig. 6, we observe that the convergence time increases exponentially as we reduce the mean squared error (i.e., increase the accuracy of the PSO algorithm). The 5% error provides an optimal transition point for the proposed experiments. Another important factor is the step-size for the bias variation controlling the ADC outputs. Unlike software-based optimization, hardware realization imposes several constraints on the algorithm, such as the size of each variable and the fixed-point accuracy. For custom circuit design using

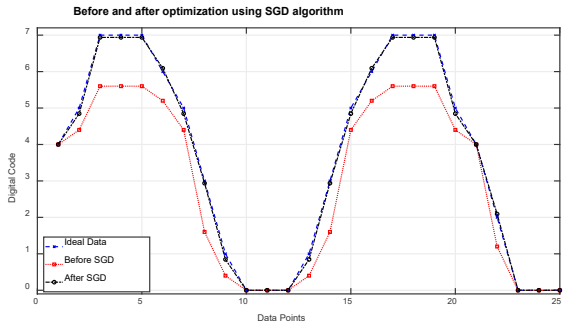


Fig. 8. Performance comparison before and after calibration using SGD.

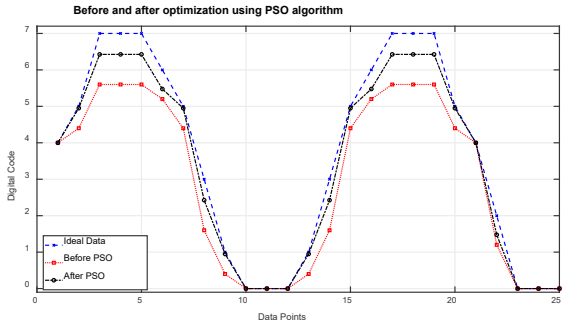


Fig. 9. Performance comparison before and after calibration using PSO.

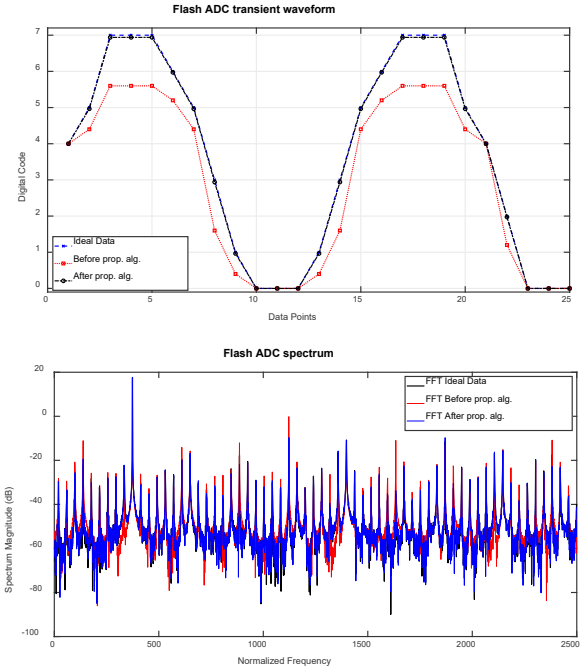


Fig. 10. Performance comparison before and after calibration using proposed algorithm (3-bit Flash ADC).

mixed-signal approaches, it translates to a large area overhead

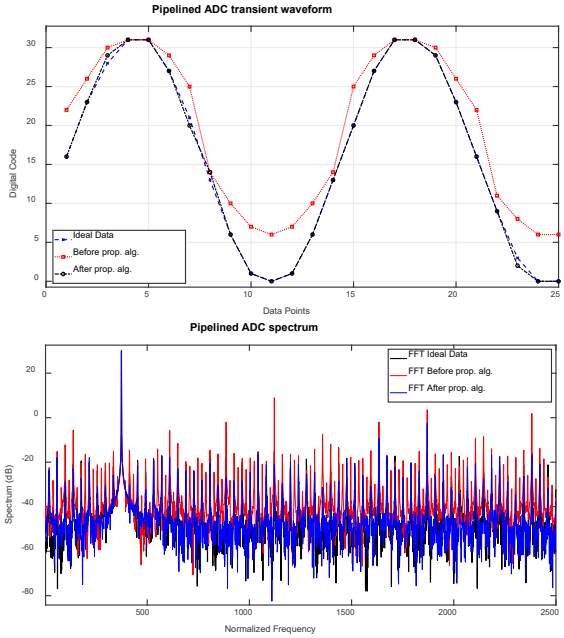


Fig. 11. Performance comparison before and after calibration using proposed algorithm (5-bit pipelined ADC).

which is undesirable. The accuracy has thus been limited to 8-bit fixed-point sufficient for the targeted ADC models.

The non-linearities were then removed using the same method as described above. For the flash ADC model, gain non-linearities were introduced in the LSB and (LSB+1) bits, while for the pipelined ADC, gain error and a random DC offset was introduced in the MSB residue stage. The model of a 5-bit pipelined ADC along with the MSB residue stage with errors is shown in Fig. 7. The output of the first stage after the introduction of these non-linearities is captured as follows:

$$V_{\text{residue}} = \text{A}_\text{v} \cdot (V_{\text{in}} - V_{\text{q}}) \cdot \text{NL}_1 - NL_2 \qquad (4)$$

where $\text{NL}_1$ and $\text{NL}_2$ are gain and offset errors.

## III. Simulation Results

We design a 3-layer NN to compare the SGD and PSO algorithms with the proposed PSO-SGD algorithm. The 3-bit flash ADC with the NN-based feedback loop was first simulated by applying SGD optimization as shown in Fig. 4. The real-time optimization approach demonstrated a 5.34dB SNR improvement with the transient data showing improved sinusoidal response (Fig. 8). The closed-loop model ADC was then re-simulated with the NN configured to use PSO optimization and a 5.24dB of SNR improvement was observed as shown in Fig. 9. Finally, the proposed PSO-SGD hybrid optimization was applied to this model (Fig. 10). Initial optimization with the PSO first yields a SNR of 18.76dB which improves to 19.34dB after SGD is ran using the global best

TABLE I. Comparison Of Computational Complexity Of Proposed Algorithm With PSO And SGD (Flash ADC).

| Operations | Particle Swarm Optimization (PSO) | | | Gradient Descent (GD) | | | Proposed Algorithm | | |
|---|---|---|---|---|---|---|---|---|---|
| | Theoretical Complexity | Calculated | Actual | Theoretical Complexity | Calculated | Actual | Theoretical Complexity | Calculated | Actual |
| Additions | n1*n2*2$l$ | 30,000,000 | 480,000 | 3$l$ | 15,000 | 15,000 | n1*n2+3$l'$ | 15,007,500 | 127,500 |
| Multiplications | n1*n2 | 3,000 | 24 | $l$ | 5,000 | 5,000 | n1*n2+$l'$ | 5,500 | 2,524 |
| ADC Calls | n1*n2 | 3,000 | 24 | $l^2$ | 25,000,000 | 25,000,000 | n1*n2+$l'^2$ | 6,253,000 | 6,250,008 |
| SNR | | | 18.76dB | | | 19.1dB | | | 19.34dB |

n1: Number of PSO iterations (8); n2: Number of swarm particles (3); $l$: Dataset length (5000); $l'$: Number of iterations for GD using proposed algorithm (2500)

position from PSO regressions taken as the starting point. The relative SNR improvement is 0.24dB and 0.34dB when compared to SGD and PSO respectively.

The above experiments were repeated replacing the flash-ADC with a traditional 5-bit pipeline ADC (Fig. 6). The SNR improves significantly from 16.49dB to 28.95dB after PSO as against the ideal SNR value of 31.76dB, while further improving to 31.4dB using the proposed algorithm, showing a 14.91dB improvement in SNR value as shown in Fig. 11.

It is also observed from these models that the SNR improvement for PSO standalone was not observed to be as good as the one with SGD standalone with similar hardware requirements. However, there is a significant reduction in simulation time to achieve the demonstrated performance, which was expected. Based on the NN, the computational complexity of the PSO algorithm is dominated by the number of iterations performed to do the optimization and the number of swarm particles, while the complexity of GD only depends on the length of the dataset. For high-speed GHz range ADCs, the size of the captured data rapidly increases to the order of $10^{10}$, while requiring 4× computations for the GD to achieve a similar performance as PSO. Further, as the number of times the feedback loop runs scales quadratically with the data set size, it takes even larger (order of $10^{20}$) number of iterations to calibrate the ADC for the targeted performance. These factors combined have a direct impact on the computational hardware requirements and the optimization latency (directly proportional to the ADC calls) as shown in Table I. As seen from the table, the proposed algorithm has significantly lower latency than SGD while having lower number of computations than PSO and providing better optimization results than either SGD or PSO.

## IV. Conclusions

This work proposes a hybrid optimization algorithm using stochastic gradient descent (SGD) and particle swarm optimization (PSO) to overcome simultaneously the large computational overhead of PSO, and the model parameter convergence limitation and high latency in the SGD. Two different architectures of a 3-bit flash and 5-bit pipeline ADC have been modelled with gain and random offsets errors in multiple stages. These models were then calibrated using the proposed multi-parameter optimization to recover the lost performance. The simulated flash performance improves by 5.58dB (0.63b ENOB), while for the pipeline ADC, overall SNR improvement of 14.91dB (2.18b ENOB) was observed showing the effectiveness of the proposed approach. The outcome from this work will be especially useful for extreme high-speed data converters that require multi-parameter optimization to overcome both static and dynamic errors.

## References


[1] Takashi Oshima, *et al*, "Fast nonlinear deterministic calibration of pipelined A/D converters," *IEEE Mid. Symp. on Cir. and Syst*, (MWSCAS), 2008, pp. 914–917.

[2] Un-Ku Moon, *et al*, "Background digital calibration techniques for pipelined ADCs," *IEEE Trans. on Cir. and Syst. II: Analog and Digital Signal Processing*, vol. 44, no. 2, pp. 102–109, Feb. 1997.

[3] P. Kiss *et al.*, "Adaptive digital correction of analog errors in MASH ADCs. II. Correction using test-signal injection," *IEEE Trans. on Cir. and Syst. II: Analog and Digital Signal Processing*, vol. 47, no. 7, pp. 629–638, Jul. 2000.

[4] S. Kundu, *et al*, "Frequency-Channelized Mismatch-Shaped Quadrature Data Converters for Carrier Aggregation in MU-MIMO LTE-A," *IEEE Trans. on Cir. and Syst. - I: Reg. Pap.*, vol. 64, no. 1, pp. 3–13, Jan. 2017.

[5] S. Gupta, *et al*, "Multi-rate polyphase DSP and LMS calibration schemes for oversampled data conversion systems," *IEEE Intl. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2011, pp. 1585–1588.

[6] S. Kundu, *et al.*, "DAC mismatch shaping for quadrature sigma-delta data converters," *IEEE Intl. Mid. Symp. on Cir. and Syst. (MWSCAS)*, 2015, pp. 1–4.

[7] Y. Tang, *et al*, "Cascaded complex ADCS with adaptive digital calibration for i/q mismatch," *IEEE Trans. on Cir. and Syst. I: Reg. Pap.*, vol. 55, no. 3, pp. 817–827, Apr. 2008.

[8] P. D. Reynolds, *et al*, "FPGA implementation of particle swarm optimization for inversion of large neural networks," *Proc. IEEE Swa. Intell. Symp. (SIS)*, 2005, pp. 389–392.

[9] B. Widrow, *et al*, "A comparison of adaptive algorithms based on the methods of steepest descent and random search," *IEEE Trans. on Ant. and Prop.*, vol. 24, no. 5, pp. 615–637, Sep. 1976.

[10] G. E. Hinton, *et al*, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, May 2006.

[11] A. Salib, et.al., "A generic foreground calibration algorithm for ADCs with nonlinear impairments", in *IEEE Trans. on Cir. and Syst. I: Reg. Pap.*, vol. 66, no. 5, pp. 1874-1885, May 2019.

[12] A. Gines, et.al., "Black-box calibration for ADCs with hard nonlinear errors using a novel INL-based additive code: a pipeline ADC case study", in *IEEE Trans. on Cir. and Syst. I: Reg. Pap.*, vol. 64, no. 7, pp. 1718-1729, July 2017.

[13] A. Ali, et.al., "16.1 A 12b 18GS/s RF sampling ADC with an integrated wideband track-and-hold amplifier and background calibration", *IEEE Intl. Solid-State Cir. Conf. (ISSCC)*, San Francisco, CA, 2020, pp. 250-252.

[14] J. Kennedy, *et al*, "Particle swarm optimization," *IEEE Intl. Conf. on Neural Networks*, 1995, vol. 4, pp. 1942–1948 vol.4.

[15] C. Briseno-Vidrios et al., "A 44-fJ/Conversion Step 200-MS/s Pipeline ADC employing current-mode MDACs," in *IEEE Jour. of Solid-State Circ.*, vol. 53, no. 11, pp. 3280-3292, Nov. 2018.

[16] D. Zhou *et al*, "A digital-circuit-based evolutionary-computation algorithm for time-interleaved ADC background calibration," in *IEEE Intl. System-on-Chip Conf. (SOCC)*, Seattle, WA, 2016, pp. 13-17.

[17] E. Ghaderi *et al*, "10.8 A 4-Element 500MHz-modulated-BW 40mW 6b 1GS/s analog-time-to-digital-converter-enabled spatial signal processor in 65nm CMOS," *IEEE Intl. Solid-State Circ. Conf. (ISSCC)*, San Francisco, CA, 2020, pp. 186-188.

[18] D. Ritchie, *et al*, "Neurally-guided Procedural Models: Amortized Inference for Procedural Graphics Programs Using Neural Networks," *Proc. Intl. Conf. on Neu. Inf. Proc. Syst.*, USA, 2016, pp. 622–630.

[19] H. M. V, *et al*, "An Integrated MaxFit Genetic Algorithm-SPICE Framework for 2-Stage Op-Amp Design Automation," *IEEE Comp. Soc. Ann. Symp. on VLSI (ISVLSI)*, 2018, pp. 170–174.

[20] D. Needell, *et al*, "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm," *Math. Program.*, vol. 155, no. 1, pp. 549–573, Jan. 2016.

[21] D. J. Allstot, J. Park, K. Choi, "Parasitic-aware optimization of CMOS RF circuits", *Springer Science & Business Media*, 2003.

[22] V. G. Gudise, *et al*, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," *Proc. IEEE Swa. Intell. Symp. (SIS)*, 2003, pp. 110–117.

[23] R. H. Byrd, *et al*, "Sample size selection in optimization methods for machine learning," *Math. Program.*, vol. 134, pp. 127–155, Aug. 2012.

[24] L. Bottou, "Large-scale machine learning with stochastic gradient descent," Proc. of COMPSTAT, 2010, pp. 177–186.