# Design and Optimization of an Active OTA-C Filter Based on STOHE Algorithm

Dalibor Barri[1,2]
[1,2] Department of Microelectronics
[1] Czech Technical University, [2] STMicroelectronics
[1,2] Prague, Czech Republic
[1]barridal@fel.cvut.cz, [2]dalibor.barri@st.com

Jiří Jakovenko
Department of Microelectronics
Czech Technical University
Prague, Czech Republic
jakovenk@fel.cvut.cz

*Abstract* – **With increasing demands on design and optimization of analog circuits in real applications, a limited number of algorithms for practical use have been presented. The drawbacks of already existing standard algorithms are in a possibility to stagnate in a not optimal solution and also big time consumption. These drawbacks have been overcome by our new proposed algorithm STOHE. The new algorithm is a combination of a STOchastic and HEuristic algorithms. As the stochastic respectively heuristic algorithm was chosen differential evolution algorithm respectively simplex algorithm. The algorithm has been verified by the design and optimization of an active OTA-C filter where the standard approach fails.**

***Keywords – differential evolution algorithm; OTA-C filter; simplex algorithm; STOHE algorithm***

## I. INTRODUCTION

In the present time, the computing technology and calculation approaches offer advanced optimization methods. These methods are currently acknowledged as powerful tools for finding out an optimal solution for complicated tasks. Typically, the character of these tasks has many independent variables, and therefore, it is very complicated to find the optimal solution. It is a reason why it is essential for us to have an advanced algorithm like proposes STOHE algorithm that may help us.

Modern optimization methods can be sorted into three main categories: heuristic, deterministic and stochastic methods. The first category contains heuristics algorithms that are designed to solve problems where direct search methods [1] are too slow, or for finding an approximate solution where traditional methods fail. A well-known example of a heuristic algorithm is the conventional Traveling Salesmen Problem (TSP). The problem is as follows: a list of cities and the distances between each city is given. What is the shortest possible route that visitor visits each city exactly once? A heuristic algorithm used to solve this problem quickly is the nearest neighbor (NN) algorithm (also known as the Greedy Algorithm) [2] or Nelder-Mead algorithm (simplex algorithm) [3]. The next methods are deterministic global optimization methods that take advantage of the analytical properties of the problem to generate a sequence of points that converge to an optimal global solution. Each step and the following step is clearly defined. For this kind of algorithm, we get the same result for the same input data. It is the main difference from other optimization algorithms. Into this category, we categorize algorithms such as linear programming, and non-linear programming [4]. The last methods are stochastic or random algorithms that generate and use random variables. For stochastic problems, the random variables appear in the formulation of the optimization problem itself, which involve random objective functions or random constraints. Stochastic optimization methods also include methods with random iterates. These methods very well find the global maximum or minimum, but the trade-off of that is unpredictable computing time. Into stochastic optimization methods, we can include the following kind of algorithm such as simulated annealing [5], evolution algorithms, and quantum annealing algorithm.

In this paper, we present a new algorithm that combines stochastic and heuristic algorithm to get the best-optimized performance for the given task. The new proposed algorithm has been named as STOHE algorithm because the algorithm is a combination of STOchastic differential evolution algorithm (DE) [6] and HEuristic simplex algorithm (SA) [3]. This combination, as we will see later, have become more robust against stagnations and moreover, it is faster in terms of computing time.

## II. STOHE ALGORITHM

The base-core of STOHE algorithm is DE algorithm that is expanded by a powerful heuristic simplex optimization method. The combination of these methods ensures advanced optimization performances such as short computing time, numeric precision, and immunity stagnation in local values.

### *A. The Evolution Part of STOHE Algorithm*

The evolution part works with a population of individuals *NP* that are randomly dispersed within the design space in the initialization part. Each one individual represents a list of variables *D* that are passed into one or more fitness (cost) functions. These fitness functions define the quality of the individual solutions and therefore also indirectly controls the search area of possible solutions.

STOHE algorithm uses in each generation *G* sets of *D*-dimension vectors $\vec{x}_{i,G}$, where all parts of the vector are independent itself (e.g., zeros and poles transfer function). In each generation *G,* there is the same number of individuals *NP* involved in the evolution process.

ISBN 978-80-261-0812-2, 

The first step toward initiating the implementation of the algorithm is the initialization of the 0th randomly generated population for each variable. These values are generated in already predefined ranges. After generating, the next step is a crossing process inside of the DE algorithm that is applied to the randomly initialized 0th population. It creates a new "noisy" vector $\vec{v}_{i,G+1}$ by the following expression:

$$\vec{v}_{i,G+1} = \vec{x}_{r_1,G} + F\,(\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) \qquad (1)$$

where $r_1$, $r_2$, $r_3$ are randomly generated values from the range $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$, $F$ is a multiple weigh coefficient in interval $F \in \langle 0,1 \rangle$, $\vec{x}_{i,G}$ is an $i$ current individual vector in the generation $G$.

The crossing process (1) ensures the diversity of all individuals, and it is run in each generation of STOHE algorithm. The crossing process is a process where for each variable of the individual $\vec{v}_{i,G+1}$ vector exists a probability $CR$ to inherit the value of the variable from the previous generation of individual $\vec{x}_{i,G}$ vector. If the crossing parameter $CR$ is equal to 1 the crossing process will be performed, and if $CR$ is equal to 0 the crossing process will not be performed. As the result of the crossing process, we have got a new vector $\vec{u}_{i,G+1}$.

The new vector $\vec{u}_{i,G+1}$ will be compared with the current vector $\vec{x}_{i,G}$ in order to disclose which one has a better solution. If the tested vector $\vec{u}_{i,G+1}$ has a better result of the fitness function than current vector $\vec{x}_{i,G}$, then the current vector $\vec{x}_{i,G}$ is replaced by the tested vector $\vec{u}_{i,G+1}$. In the opposite case, the current vector $\vec{x}_{i,G}$ keeps all parameters and is passed without any changes into the next generation $G$+1.

### B. Integration of the Simplex Method into STOHE Algorithm

The previous section has explained the evolution part of STOHE algorithm that offers *NP* new possible solutions. In this part, we will explain the integration of simplex algorithm into the algorithm.

STOHE algorithm works in two phases. The first evolution phase ensures diversity of members of the population in the generation. The outcome of the first differential evolution phase is, an intermediate population. In the second phase of STOHE algorithm, the percentage part of generation from the intermediate population (*PPGIP*) is passed to the simplex algorithm. This algorithm looks for minimum value in the closest possible solutions.

The benefit of this approach is shown in Fig. 1, where we can see the following idea. The first randomly generated individuals of population *G* (pink circles in Fig. 1a) are passed into the intermediate population (Fig. 1b). The *PPGIP* of individuals (plus symbols highlighted by green color in Fig. 1b) will be modified by SA. The reason for this step is that the SA algorithm can effectively find out the local minimum. Therefore, the settings of the *PPGIP* should be done very sensibly. Consider the following example. If we pass all individuals into the SA process and no one

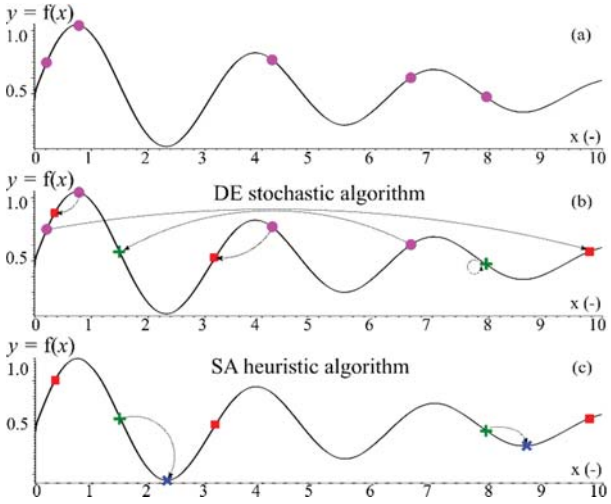


Figure 1: A simple example of searching for global minimal value by STOHE algorithm. The first randomly generated individuals of population *G* are represented as circles with pink color (a); intermediate population (b) shows optimization by the stochastic algorithm for all individuals from previous generation *G* and the result of it are green and red individuals, plus and square symbols respectively. The ratio of green and red individuals is controled by *PPGIP* parameter. The green individuals are passed to the heuristic algorithm, and red individuals are directly passed to the next generation *G*+1; the next generation *G*+1 (c) shows blue cross individuals as the result of stochastic and heuristic algorithm, red individuals are on the same position as it was in the intermediate population.

solution will be near to the global minimum, then the algorithm will never find out the global minimum. In contra with the situation (blue crosses in Fig. 1c), where we have succeeded to get the global minimum.

### C. The Control Parameters of STOHE Algorithm

STOHE algorithm is based on the iteration of the DE algorithm and direct searching method in an area by SA. Its control parameters are *F, CR, NP,* and *PPGIP* and each of them is more describes in this sub-section.

*F* is a mutation factor used to generate a "noisy" vector. We recommend setting this parameter in a range $F = 0.5 \ldots 0.8$. The smaller values lead to premature convergence and, on the contrary, the higher values lead to stagnation of the population.

*CR* is a crossing factor that ensures diversity of individuals in the population and thus partially prevents stagnation. We recommend setting this parameter in a range $CR = 0.5 \ldots 0.8$. The smaller values lead to a slowdown in finding optimal solutions and higher values increase the risk of stagnation and rapidly reduce the number of possible solutions.

*NP* is a number of individuals (solutions) in the population of each generation. The value of this parameter is published in [6] and [7] as multiple values of independently optimized variables *D*. In [7], the value of this parameter is recommended to set to 20*D*. Otherwise, it is in [6], where the value of this parameter is recommended to set to 10*D*. We recommend setting *NP* parameter to 10*D*. Larger population reduces the risk of stagnation, but on the other hand, the algorithm becomes computational more complicated and finding out the solution is so far away. We do not recommend a lower number of individuals in the population

ISBN 978-80-261-0812-2, © University of West Bohemia, 2019

because it reduces the ability to find out the optimal solution and also it can easier and early achieves convergence to one solution. However, there are also applications where it is possible. It is for example in case, where we have not so many independent variables of individuals (less than five). Choosing of population size is individual matters and depends on the nature of the case.

*PPGIP* is the percentage part of the intermediate population of the previous generation. We recommend choosing a value in a range $PPGIP = 0.2 \ldots 0.6$. Greater or lesser values led to getting a slower calculation of optimization.

### *D. The Fitness Function of STOHE Algorithm*

The fitness function is not built in as a part of STOHE algorithm. The algorithm only invokes a request for calculation of it. It is an independent function that calculates a fitness value for each individual in the population in each generation step.

The fitness function is defined by users of STOHE algorithm who decide the complexity of the tasks, and based on it they define the fitness function. In our paper, we are using two kinds of fitness function. The first one is used to find transmission characteristic that meets a frequency requirement of a low-pass active OTA-C filter. Frequency requests are defined as a set of attenuations $\{\delta_1, \delta_2, \ldots, \delta_N\}$ in specific frequency values. In case that particular frequency values are within the passband of transmission characteristic (in our case, it is for frequency $f$ from 0 to 300 kHz in Fig. 2), the fitness function can be defined as (2)

$$K_1(\delta) = \begin{cases} 0 & \text{if } e(\delta_i) < 0 \text{ and } e(\delta_i) > E(\delta_i) \\ \sum_{i=1}^{N} e(\delta_i) & \text{if } e(\delta_i) > 0 \text{ and } e(\delta_i) > E(\delta_i) \\ \sum_{i=1}^{N} |E(\delta_i) - e(\delta_i)| & \text{if otherwise} \end{cases} \quad (2)$$

where $E(\delta_i)$ is requested value of the attenuation in the $\delta_i$ point, $e(\delta_i)$ is a real value of the attenuation in the $\delta_i$ point, $\delta_i$ is a point of interest – in our case that point represents the frequency of interest, $N$ is the total number of the points of interest.

In case that specific frequency values are not within the passband of transmission characteristic (in our case it is for frequency $f$ higher than 300 kHz in Fig. 2), it can be defined as (3)

$$K_1(\delta) = \begin{cases} 0 & \text{if } e(\delta_i) < E(\delta_i) \\ \sum_{i=1}^{N} |E(\delta_i) - e(\delta_i)| & \text{if } e(\delta_i) > E(\delta_i). \end{cases} \quad (3)$$

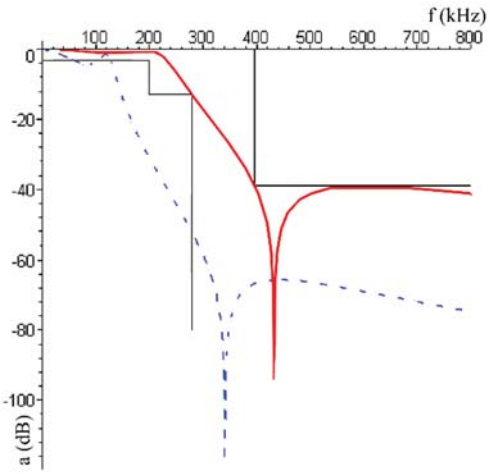


Figure 2: Transmission characteristic before optimization (blue dashed line), after optimization (red solid line)

Hence, in the case where a tested value of the attenuation in the passband is lower than zero and higher than the real value, the $K_1(\delta)$ is set to zero. Similarly, it is in a case where a tested value of the attenuation in the stopband is lower than the real value $K_1(\delta)$. In this case, $K_1(\delta)$ is also set to zero.

The second fitness function $K_2(i,\delta)$ is different from the first one $K_1(\delta)$ because a purpose of this fitness function is to find an optimal solution with specific requests on the already existing structure. $K_2(i,\delta)$ uses as input transmission function coefficients that were founded in cooperation with fitness function $K_1(\delta)$. In our case, the fitness function $K_2(i,\delta)$ respects dynamic voltage states of the designed OTA-C filter by the following expression:

$$K_2(i,\delta) = W_1 \sum_{i=1}^{K} \left( \frac{E_1(i) - e_1(i)}{E_1(i)} \right)^2 + W_2 \sum_{l=1}^{N} |E_2(\delta_l) - e_2(\delta_l)| \quad (4)$$

where $E_1(i)$ is a requested coefficient of transmission function, $e_1(i)$ is a real transmission coefficient designed by STOHE algorithm, $K$ is the total number of coefficient of the transmission function, $E_2(\delta_l)$ is requested value of dynamic voltage states of the designed OTA-C filter at the $\delta_l$ point, $e_2(\delta_l)$ is a real value of dynamic voltage states of the designed OTA-C filter in the $\delta_l$ point, $\delta_l$ is a point of interest – in our case, this point represents the frequency where the dynamic voltage has the highest value, $N$ is the total number of the points of interest, $W_1$ and $W_2$ are weight constants. Dynamic voltage state is calculated as a voltage ratio of the voltage in the output of each OTA-C circuit to the output voltage of the last OTA-C circuit.

The second fitness function $K_2(i,\delta)$ is advanced because due to two weighted constants $W_1$ and $W_2$, we can control optimization of the designed circuit. Let's consider that $W_1$ is 100 times higher than $W_2$. In this case, we have entered a 100 times higher request for

the transmission function contra to dynamic voltage states of the designed active OTA-C filter. The optimal setting of the $W_1$ and $W_2$ weights is not an easy matter. However, if we set these parameters correctly, we can achieve very high-quality results. The general rule for balancing it does not exist. The user should rely on his intuition or his experience gained in practice. Therefore, we need to be careful in the setting of these constants.

## III. Experiment

### A. Requested Parameters on the OTA-C Filter

To verify complexity and practicability of STOHE algorithm the active low-pass OTA-C filter has been chosen as a reference analog circuit. The OTA-C filter should follow frequency criteria defined by tolerant schematic Fig. 2, and the maximal dynamic voltage states of the designed OTA-C filter should be in a range from 0.7 to 1.15. As the next point, the design should respect the real properties of the discrete transconductance operational amplifier device LM 13700 [8].

### B. Algorithm Setup

STOHE algorithm is able to cooperate with all settings of differential evolution algorithm such as *DE/rand/1/bin, DE/rand/2/bin, DE/current-to-rand/1/bin,* and other described in [6], and [7]. In this paper, we have chosen *DE/rand/1/bin* because it is basic settings of DE algorithm.

Since the DE algorithm is the fundamental algorithm of STOHE algorithm, the best settings of the DE algorithm of STOHE algorithm should be researched. Therefore, DE algorithm was investigated with the following a set of parameters $F$ = {0.05, 0.8, 0.95}, $CR$ = {0.05, 0.8, 0.95}, and $NP$ = {10, 50, 100}. The DE setting parameters $F$ and $CR$ were combined with each other, and for each set, it was run 30 times. The algorithm was stopped when the fitness function (2) and (3) reached the fitness value lower than $10 \cdot 10^{-5}$ or when a number of generation $G$ reached a value equal to 200. The maximal number of generations was set based on our experience. The average result value of the fitness function $K_1(\delta)$ and its status decision is reported in Table I, and the progress of it is shown in Fig. 3.

TABLE I. IMPACT OF DIFFERENT DE SETTING PARAMETERS ($F$, $CR$) ON FITNESS FUNCTION $K_1(\Delta)$

| $F$ (-) | $CR$ (-) | $\overline{K_1(\delta)}$ (dB) | status |
|---|---|---|---|
| 0.05 | 0.05 | $1.21 \cdot 10^{-2}$ | FAIL |
| 0.05 | 0.95 | $5.49 \cdot 10^{0}$ | FAIL |
| 0.05 | 0.80 | $1.67 \cdot 10^{0}$ | FAIL |
| 0.95 | 0.05 | $4.47 \cdot 10^{-3}$ | FAIL |
| **0.95** | **0.95** | $\mathbf{2.73 \cdot 10^{-5}}$ | **PASS** |
| **0.95** | **0.80** | $\mathbf{9.96 \cdot 10^{-5}}$ | **PASS** |
| 0.80 | 0.05 | $3.09 \cdot 10^{-3}$ | FAIL |
| **0.80** | **0.95** | $\mathbf{3.34 \cdot 10^{-5}}$ | **PASS** |
| **0.80** | **0.80** | $\mathbf{1.33 \cdot 10^{-5}}$ | **PASS** |

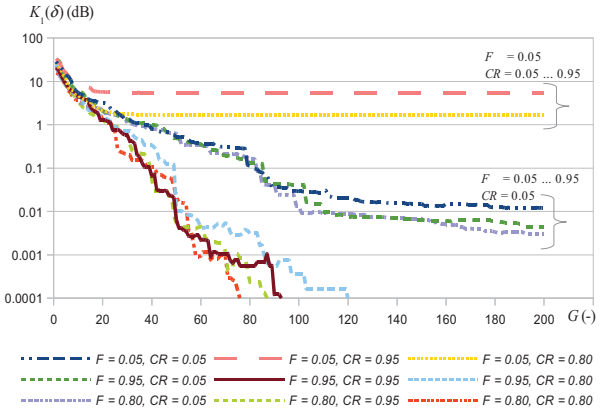


Figure 3: Impact of different DE setting parameters ($F$, $CR$) on fitness function $K_1(\delta)$

As we can see in Fig. 3, if the mutation factor $F$ is too low, the DE algorithm fails independently on settings of crossing factor $CR$. The similar failure we can also see for crossing factor $CR$. If the $CR$ is too low then, DE fails. In the case, where mutation factor $F$ and crossing factor $CR$ are not so low, we will obtain better results. After several different settings of the DE algorithm (Table I), we have got the best result for the combination where $F$ is equal to 0.8 and $CR$ is equal to 0.8.

The best combinations of the $F$ and $CR$ setting parameters have been passed for further investigation. The next step of the investigation is to analyze the impact of $NP$ and $PPGIP$ setting parameter on the final result in terms of time consumption and accuracy.

Results of this analysis with different settings of STOHE algorithm are reported in Table II. Each result in Table II has been calculated as the average value of computing time per one generation $G$ for the specific settings of STOHE algorithm. In order to obtain relevant data, the analyses have been performed 200 times. The Table II shows that the average compute time is increasing with the higher value of a number of population $NP$ and also it is increasing with the higher value of percentage part of the intermediate population of the previous generation $PPGIP$.

TABLE II. COMPARISON OF COMPUTING TIME FOR DIFFERENT PPGIP AND NP SETTINGS OF STOHE ALGORITHM

| | average computing time $t_{avg}$ (s) @ | | | | | |
|---|---|---|---|---|---|---|
| *PPGIP* / *NP* | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 |
| 10 | 0.55 | 0.63 | 1.18 | 1.47 | 1.70 | 1.96 |
| 20 | 1.35 | 2.23 | 2.75 | 3.37 | 3.93 | 4.46 |
| 30 | 2.19 | 3.52 | 4.47 | 5.34 | 6.22 | 7.19 |
| 40 | 3.08 | 5.05 | 6.19 | 7.58 | 8.81 | 10.04 |
| 50 | 4.02 | 6.51 | 8.13 | 9.79 | 11.33 | 12.73 |
| 60 | 5.34 | 7.96 | 9.93 | 12.21 | 14.09 | 16.30 |
| 70 | 6.57 | 10.18 | 12.57 | 14.89 | 17.51 | 19.93 |
| 80 | 8.23 | 11.98 | 14.86 | 17.81 | 20.60 | 23.44 |
| 90 | 10.04 | 13.92 | 17.22 | 20.53 | 23.67 | 27.05 |
| 100 | 11.53 | 15.91 | 19.63 | 23.36 | 27.18 | 30.90 |

ISBN 978-80-261-0812-2, © University of West Bohemia, 2019

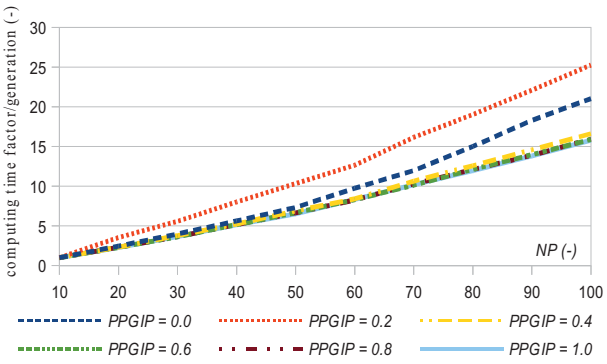


Figure 4: A computing time factor per generation vs. number of population NP

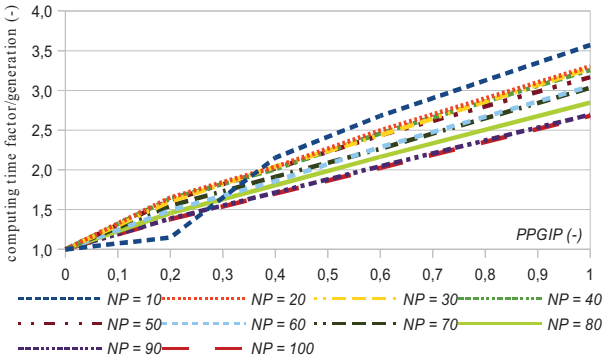


Figure 5: A computing time factor per generation vs. percentage part of the intermediate population of the previous generation *PPGIP*

Based on data in Table II, we can analyze the relative impact of a number of population *NP* on computing time. The reference settings of STOHE algorithm has been chosen a combination with *NP* equal to 10 and *PPGIP* equal to 0. For this setting, STOHE algorithm works as the DE algorithm with a small population equals to 10 individuals. The result of this analysis is depicted in Fig. 4. As we can see, the computing time on generation is rapidly increasing with respect to the reference settings of STOHE algorithm. If the population is five times higher than reference one, then computing time on generation is ten times of reference one. If the population is ten times higher than the reference, then computing time on generation is twenty-five times of reference one. So, the computing time increases rapidly faster than any linear curve.

The opposite observation (in compare with the previous case) we have got for the case where we have modified *PPGIP* parameter in a full range i.e. from zero to one as it is depicted in Fig. 5. As we can see, if the *PPGIP* is set to 0.5 it has value 2.0 for *NP* equals to 80. If the *PPGIP* is set to 1.0 (it is two times of the previous case), the computing time factor on generation is less than two times for the same *NP* parameter.

Based on the analysis, it can be said that the number of population *NP* has a much more significant effect on the computing time per generation than the different value of the percentage part of the intermediate population *PPGIP*. This knowledge should be respected during settings of STOHE algorithm.

At the end of this part, the optimal settings of STOHE algorithm can be determined to get the best performance of this algorithm. After a lot of analysis runs, it can be said that the optimal setting of the algorithm has following character: the mutation factor *F* is equal to 0.8 and the crossing process factor *CR* is also equal to 0.8.

### C. Optimization of the OTA-C Filter

In Fig. 2, there are shown two curves that represent the status of the designed OTA-C filter by STOHE algorithm before and after optimization, respectively. As we can see, the impact of real properties absolutely change the curve and the optimization of the OTA-C filter is needed. There is not possible to rely on the design based on only the ideal components.

In Table III and Table IV, the final results of the OTA-C filter are shown before and after optimization. The state *before optimization* means a realization of OTA-C filter that includes parasitic components such as ESR, ESL and parasitic capacitance depicted in Fig. 6 but without run STOHE algorithm. In opposite, it is for the case *after optimization*. The final results passed our criteria, and moreover, the achieved results are better than it was required.

For the design and optimization of the active OTA-C filter, STOHE algorithm worked with different seeds of the control parameters. The algorithm has found out the best solution for the following settings its control parameters: $NP = 50$, $F = 0.8$, $CR = 0.8$, and $PPGIP = 0.2$. The optimization process took 108 minutes. The number of optimized parameters was 12 (7x $g_m$, 5x $C_l$) with respecting parasitic components. The parasitic components are highlighted by light-grey color in Fig. 6. For evaluation has been used a laptop with the Intel® Pentium® M, 1.73 GHz processor. The algorithm ran in Maple software from The Maplesoft™ company [9] with the additional packages for circuit simulation - Syrup [9] and SYNTFIL package [10].

TABLE III. VOLTAGE DYNAMIC STATES

| Voltage ratio (-) | Voltage dyn. states before optimization (-) | Voltage dyn. states after optimization (-) |
|---|---|---|
| $V_{\max,gm1}/V_{\max,gm4}$ | 2.02 | 0.70 |
| $V_{\max,gm2}/V_{\max,gm4}$ | 1.16 | 1.09 |
| $V_{\max,gm3}/V_{\max,gm4}$ | 3.37 | 1.07 |

TABLE IV. AMPLITUDE PARAMETER RESULT

| Frequency (kHz) | Requested attenuation (dB) | Real attenuation (dB) |
|---|---|---|
| < 200 | 3.00 | 1.19 |
| 200 – 280 | 13.00 | 13.00 |
| > 280 | 40.00 | 40.00 |

ISBN 978-80-261-0812-2, © University of West Bohemia, 2019

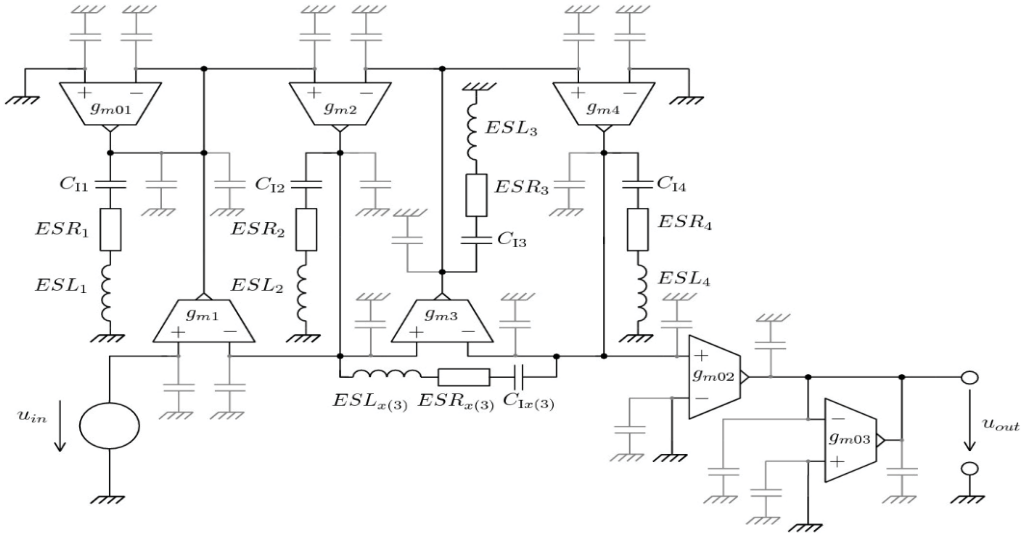


Figure 6: The final design of the active OTA-C filter with respecting real components. Real additional elements of the ideal devices are grey colored (ESL, ESR parasitic components).

## IV. Conclusions

In this paper, a new STOHE algorithm has been presented. The invented algorithm combines a stochastic and heuristic method that offers up a very powerful algorithm that allows designing any analog circuits such as operation amplifier, transconductance operation amplifier or other more complexly structures. As the reference analog block for verification of STOHE algorithm has been chosen a low-pass active OTA-C filter. The achieved electrical parameters of the designed circuit are the same or better than it has been required (Table IV) and this data was not able achieved by the standard way (blue dashed line vs. red solid line depicted in Fig. 2). This approach of design and optimization has been verified, and therefore we recommend to use it also for other electrical circuit design or optimization tasks. Based on the results (Table III and Table IV) we can say that design and optimization by STOHE algorithm is a possible way leading to a successful one solution. It is recommended to use it to achieve correct results in a short time.

The STOHE algorithm has been also investigated in terms of a possibility to accelerate convergence by modification of percentage part of the intermediate population of the previous generation *PPGIP*. We have observed that the computing time is increasing with the increasing *PPGIP* parameter. In our experiment, we have defined as the best settings for *PPGIP* equal to 0.2. We recommend setting this parameter in a range from 0.2 to 0.6 because if the *PPGIP* is lower than 0.2 or higher than 0.6, then there is a success rate lower due to a possibility of stagnation. Also, this recommendation ensures that the global minimum will be achieved. In order to be sure that the global minimum has been found, we can run the algorithm in parallel mode and evaluates partial results.

As the next point of the investigation has been an impact of a number of population *NP* on computing time. The result is that the computing time per generation is much more dependent on the number of population *NP* than on the different value of the percentage part of the intermediate population *PPGIP*. This knowledge should be respected during settings of STOHE algorithm.

The final results prove that the proposed STOHE algorithm is working well and uses of it can also be extended for other optimization tasks.

## Acknowledgments

This work is part of the CTU SGS grant no. SGS17/188/OHK3/3T/13.

## References

[1] R. Hooke and T. A. Jeeves, ""Direct Search" Solution of Numerical and Statistical Problems", J. ACM, vol. 8, no. 2, pp. 212–229, 1961.

[2] Z. Sevkli and F. Evilgen, "Variable neighborhood search for the orienteering problem", Computer and Information Sciences – ISCIS, pp. 134–143, 2006.

[3] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization", Computer Journal, vol. 7, no. 4, pp. 308–313, 1965.

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Second Edition. Cambridge University Press, 1992.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing", Science, vol. 220, no. 4598, pp. 671–680, 1983.

[6] R. Storn, "On the usage of differential evolution for function optimization", Proceedings of North American Fuzzy Information Processing, pp. 519–523, 1996.

[7] K. Price, "An Introduction to Differential Evolution", Advanced Topics In Computer Science Series, Maidenhead (UK): McGraw-Hill Ltd., pp. 79–108, 1999.

[8] National Semiconductor [Online]. Available: http://www.national.com. [Accessed: 05-Mar-2018].

[9] Waterloo Maple, Inc. [Online]. Available: http://www.maplesoft.com. [Accessed: 05-Mar-2018].

[10] SYNTFIL package [Online]. Available: http://syntfil.feld.cvut.cz/. [Accessed: 05-Mar-2018]