# OPTIMIZATION OF WIRELESS COMMUNICATIONS APPLICATIONS USING DIFFERENTIAL EVOLUTION

Rainer Storn (ROHDE & SCHWARZ GmbH & Co. KG
Radiocommunications Systems Division
81671 Muenchen, Germany, Email: rainer.storn@rohde-schwarz.com)

## ABSTRACT

Differential Evolution (DE) is a very simple yet powerful algorithm for finding the global minimum of multivariate functions. Since DE belongs to the so-called direct search methods it can also handle functions which are hihgly nonlinear, non-differentiable, mixed-integer or even discrete. Also constraints can be incorporated with relative ease. This broad applicability makes DE a suitable candidate for design taks in wireless communications that are difficult to tackle otherwise.

Real-world examples are RF-circuit design, filter design, antenna optimization, construction of error correcting codes for CDMA and others. This contribution shows possibilites how to transform a design task into a minimization task, which then can be solved with a global optimization method like DE. It introduces the most effective variants of DE and, as an example, details a digital filter design problem which was successfully used for a channel simulator at Rohde & Schwarz.

## 1. INTRODUCTION

Many design problems for wireless communications applications can be recast into a parameter optimization task which means that the structure of the problem is fixed and just parameters need to be adjusted to find the best solution. In most cases the best solution can be represented by the minimum of a cost function f(**x**) where **x** is the vector of parameters $x_j$, j=1, 2, …, D. In order to minimize such a cost function traditionally the theory of multivariate optimization [1] is employed which heavily relies on gradient computations. However, real-world design problems often result in highly nonlinear, constrained cost functions with parameters from both the continuous and discrete number space. In this case gradient-based techniques are difficult to apply and one has to resort to optimization methods with a more general scope. Since communication designers are not necessarily optimization experts the optimization method should both be effective and straightforward to use. Differential Evolution (DE) [2] is such a method and has proven its effectiveness in a wide variety of communication applications, like antenna design [3], …, [6], RF circuit design [7], …, [9], pulse shape optimization [10], design of error correcting codes [11], [12], filter design [13], …, [16], power allocation optimization [17], and others.

## 2. THE RATIONALE FOR DIFFERENTIAL EVOLUTION

As already indicated before cost functions f(**x**) belonging to real-world design problems typically exhibit the following difficulties:

1) f(**x**) has regions of non-differentiability, especially if there are parameters from the continuous as well as the discrete domain.
2) f(**x**) is multimodal, i.e. there is more than one minimum.
3) f(**x**) exhibits several constraints on the parameters and/or the cost function itself.

There are other issues like parameter dependence or number of objectives a more detailed treatment of which can be found in [2], but the rationale for DE can already be deduced from the problems 1), 2), and 3) above.

In order to handle the first problem, i.e. non-differentiality of a cost function, using a minimization method not relying on gradients is the natural answer, so DE belongs to the so-called *direct search methods*.

The second problem, multimodality, gives rise to two subproblems, the *starting point problem* and the *relocation problem* which are best illustrated by an example. Figure 2.1 shows the so-called peaks function [2] which has several local minima and one global minimum. It is obvious that using a gradient minimizer and choosing a disadvantageous starting point will get the minimizer trapped in a local minimum rather than the global one. Obviously more than one starting point must be chosen. This multiplicity can be implemented sequentially in time such that several runs of the same minimization routine can be done with varying starting points, or the multiplicity is put into effect by using a population of *Np* points which is worked upon simultaneously. DE takes the latter approach and hence belongs to the *population-based methods*. Population-based methods have the advantage that interaction and information transfer between the points can be established more extensively.

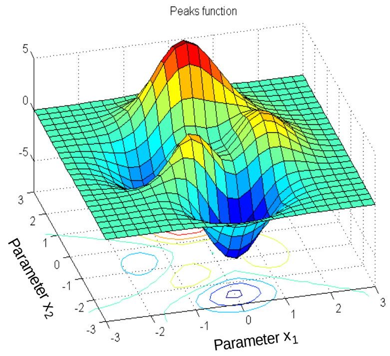


**Figure 2.1**: The peaks function [2] has several minima.

Since the topology of the cost function surface is generally unknown, the search for the minimum requires to alter the starting points from one minimization iteration to the next into a new set of probing points or probing vectors (the term point or vector will be used interchangeably). The question is how to relocate the probing points in terms of direction and step size. Many population based minimizers employ predefined probability distribution functions such as Gaussian or Cauchy to determine the relocation. Such a choice, however, still leaves the question on how to select the standard deviations and hence just shifts the relocation problem to a problem of determining standard deviations. DE takes a different approach by determining the probability distribution from the population itself by computing *difference vectors* from pairs of population vectors.

Without going into the details of DE yet the advantage of using difference vectors for relocation shall be illustrated in Figs. 2.3 to 2.5 which show the population vectors on the left hand side of the drawings and the ensuing difference vector distribution on the right hand side. Note that for the sake of clarity only the endpoins of the vectors and difference vectors are shown in all cases. It can readily be seen that the difference vector distribution used for relocation is strongly varying in shape and adapts itself to the cost function topology.

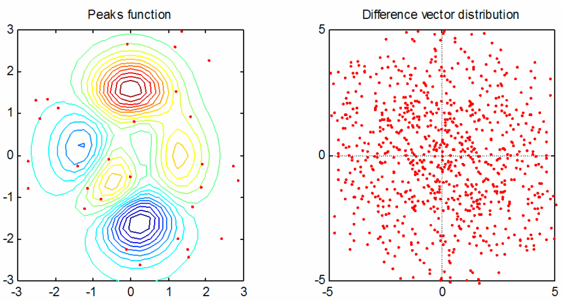


**Figure 2.3**: Generation g=1 using Np = 30 points.

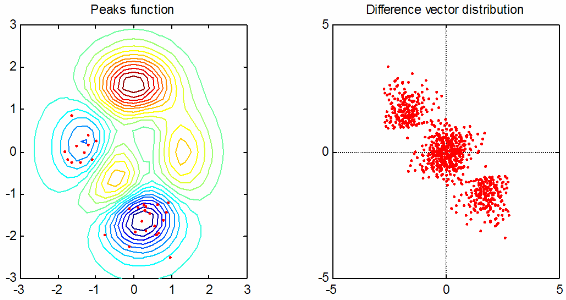


**Figure 2.4**: Generation g=12 using Np = 30 points.

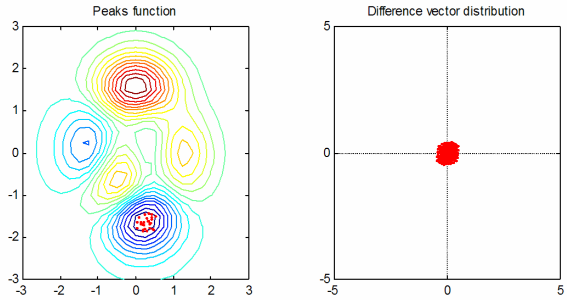


**Figure 2.5**: Generation g=20 using Np = 30 points.

This is in contrast to, for example, a Gaussian distribution which would always be represented by a single cloud of points during the entire course of the optimization.

Before the problem domain, i.e. how to deal with constraints is elaborated in chapter 4, a more detailed look at DE is in order.

## 3. DIFFERENTIAL EVOLUTION IN DETAIL

The standard version of DE can be defined by the following constituents:

1) The population

$$\mathrm{P}_{\mathbf{x},g} = \left(\mathbf{x}_{i,g}\right), \quad i = 0,1,\ldots,Np-1, \quad g = 0,1,\ldots,g_{\max}, \qquad (3.1)$$
$$\mathbf{x}_{i,g} = \left(x_{j,i,g}\right), \; j = 0,1,\ldots,D-1.$$

where Np denotes the number of population vectors $\mathbf{x}_{i,g}$, g defines the generation counter, and D the dimensionality, i.e. the number of parameters.

2) The initialization of the population via

$$x_{j,i,0} = \mathrm{rand}_j[0,1) \cdot \left(b_{j,\mathrm{U}} - b_{j,\mathrm{L}}\right) + b_{j,\mathrm{L}}. \qquad (3.2)$$

The D-dimensional initialization vectors, $b_{j,L}$ and $b_{j,U}$ take the lower and upper bounds of the parameter vectors $x_{i,j,g}$ into account. The random number generator, $\mathrm{rand}_j[0,1)$, returns a uniformly distributed random number from within the range [0,1), i.e., $0 \leq rand_j[0,1) < 1$. The subscript, j, indicates that a new random value is generated for each parameter.

3) The relocation of a, yet to be defined, *base vector* $\mathbf{y}_{i,g}$ by using a difference vector based mutation

$$\mathbf{v}_{i,g} = \mathbf{y}_{i,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}). \tag{3.3}$$

to generate a *mutation vector* $\mathbf{v}_{i,g}$. Setting $\mathbf{y}_{i,g} = \mathbf{x}_{r0,g}$ defines what is often called *classic DE* where the *base vector* is also a randomly chosen population vector. The random indexes r0, r1, and r2 should be mutually exclusive. The *difference vector* indices, r1 and r2, are randomly selected once per base vector. The *scaling factor* F is adding diversity by preventing $\mathbf{v}_{i,g}$ to be located at the coordinates of an already existing vector. Another popular variant for relocation is

$$\mathbf{v}_{i,g} = \mathbf{y}_{best,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}). \tag{3.4}$$

where the currently best vector is used as a base vector. This choice makes the search more greedy and results in faster convergence if the cost function to minimize is not deceptive. The downside of increased greediness is an increased possibility for getting trapped in a local minimum.

Relocation by differential mutation is actually the crucial ingredient of DE giving the method its name. Using difference vectors for relocation leads to an effect called *contour matching* [2], i.e. the vector population adapts itself automatically to the contours of the cost function as seen in Figs. 2.3 to 2.5. A closer look at these figures reveals that the distribution reinforces the search around the most promising regions. Also as the population converges the difference vectors automatically become shorter and hence foster a more refined search.

4) Diversity enhancement

DEs method of determining the relocation from within the population comes at the price of a low number of difference vectors, and hence low diversity if Np is small. It has proven advantageous to add diversity to the relocation computation. The classic variant of diversity enhancement is *crossover* which mixes parameters of the mutation vector $\mathbf{v}_{i,g}$ and the so-called *target vector* $\mathbf{x}_{i,g}$ in order to generate the *trial vector* $\mathbf{u}_{i,g}$. The most common form of crossover is uniform and is defined as

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } (\text{rand}_j[0,1) \leq Cr) \\ x_{j,i,g} & \text{otherwise.} \end{cases} \tag{3.5}$$

In order to prevent the case $\mathbf{u}_{i,g} = \mathbf{x}_{i,g}$ at least one component is taken from the mutation vector $\mathbf{v}_{i,g}$, a detail that is not expressed in Equation (3.5). In [2] it is shown, however, that crossover should only be used lightly because it has the potential to destroy DE's contour matching property.

There are various other possibilities of further introducing diversity. One common method is called dither [2] and works according to

$$F_{dither} = F_l + rand_g(0,1) \cdot (F_h - F_l) \tag{3.6}$$

Dither randomly selects the weighting factor for the difference vector within a predefined range. Chakraborty showed that indeed DE is improved if dither is added to classical DE, especially if the cost functions are noisy [18]. Another advantage of dither is that it is neutral with respect to the contour matching property of DE. In Figure 3.3 the values $F_h$=1 and $F_l$=0.5 are used.

Another interesting diversity-enhancing method is called jitter [2]. In jitter F is randomized for each single parameter j=0, 1, …, D-1 and for every new mutant vector i according to

$$F_{jitter,i} = F \cdot (1 + \delta \cdot (rand_j[0,1) - 0.5)) \tag{3.7}$$

For jitter it seems to be very important that δ be small, e.g., δ=0.001 to not affect contour matching excessively.

```
while (convergence criterion not yet met)
{
    //---xi defines a vector of the current vector population-----
    //---yi defines a vector of the new vector population---------
    Fd = 0.5*(1+rnd());  //Dither factor Fd ex [0.5, 1] increases
                         //diversity
    for (i=0; i<Np; i++) //there are Np vectors in a population
    {
       r1 = rand(Np); //select a random index from 1, 2, ..., Np
       r2 = rand(Np); //select a random index from 1, 2, ..., Np
       r3 = rand(Np); //select a random index from 1, 2, ..., Np
       ui = xr3 + Fd*(xr1 - xr2);//add weighted difference vector
       if (f(ui) <= f(xi))       //trial vector ui better than xi ?
       {
         yi = ui;                //if yes ui wins
       }
       else
       {
         yi = xi;
       }
    }
    swap(Y,X); //new population Y becomes current one
    ...
}//end while
...
```

**Figure 3.3:** Basic version of DE in pseudo code using dither for diversity enhancement.

5) Selection

DE uses simple one-to-one survivor selection where the trial vector $\mathbf{u}_{i,g}$ competes against the target vector $\mathbf{x}_{i,g}$. The vector with the lowest objective function value survives into the next generation g+1. A mathematical representation of this selection strategy is provided in Equation (3.8)

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{3.8}$$

Figure 3.3 shows a simple pseudo-code version of DE which uses dither as a diversity enhancement method.

## 4. HANDLING OF CONSTRAINTS

Constraints are present very often in practical optimization problems, and DE offers several convenient ways to handle them [2]. Constraints on the parameters are best handled with a method called *bounce back* which is shown in Figure 4.1.

```
...
u_i = x_r3 + F*(x_r1 - x_r2);
for (j=0; j<D; j++) //----BOUNCE BACK-----
{
   if (u_j,i<x_j,L) //if child parameter exceeds lower bound
   {
      u_j,i = x_j,r0 + rand(0,1)*(x_j,L-x_j,r0);
   }
   if (u_j,i>x_j,U) //if child parameter exceeds upper bound
   {
      u_j,i = x_j,r0 + rand(0,1)*(x_j,U-x_j,r0);
   }
}
if (f(u_i) <= f(x_i))
...
```

**Figure 4.1:** Parameter constraints are best handled after mutation and before evaluation .

In bounce back parameters which go astray after mutation are placed randomly between the base vector and the bound and hence the trial vector is placed automatically within bounds. In order to handle inequality constraints these are best brought to normal form, i.e

$$\gamma_m(\mathbf{x}) \le 0, \quad m = 1,2,\ldots,M \tag{4.1}$$

All that needs to be changed then is the selection criterion of DE where a trial vector now wins against the target vector only if either:

1. The trial vector **u** and target vector **x** fulfill all constraints but the trial vector **u** has the lower cost function
2. or the trial vector **u** fulfills all constraints but the target vector **x** doesn't
3. or at least each $\gamma_m(\mathbf{u}) < \gamma_m(\mathbf{x})$, i.e. the trial vector fulfills all constraints better than the target vector.

For inequality constraints it is convenient to write

$$\varphi_n(\mathbf{x}) = 0, \quad n = 1,2,\ldots,N \tag{4.2}$$

and work with an extended cost function

$$f'(\mathbf{x}) = f(\mathbf{x}) + \sum_{n=1}^{N} w_n \cdot \varphi_n^2(\mathbf{x}) \tag{4.3}$$

where $w_n$ denote weights which, however, can usually be set to unity.

## 5. DIGITAL FILTER DESIGN

As an example for an optimization task belonging to the wireless communication domain the design process for a digital filter employing DE is described. The digital filter is used at Rohde & Schwarz in a channel simulator for wireless applications. The tolerance scheme of the filter is depicted in Figure 5.1 which shows a filter with Gaussian magnitude of extremely narrow bandwidth, i.e. $\Omega \in [0, 0.0046]$, that was set out to be implemented by an FPGA-based IIR-filter of $8^{th}$ order and a wordlength of 32 bits.

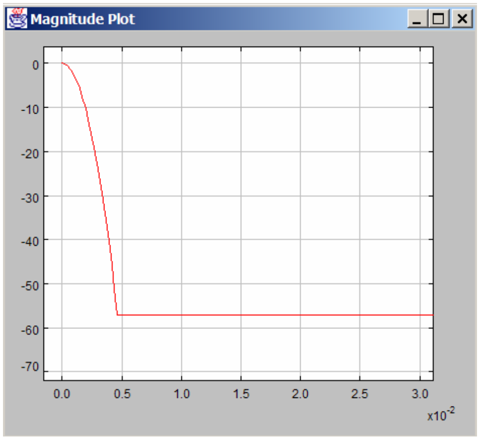


**Figure 5.1:** Tolerance scheme for the Gaussian magnitude response $A(\Omega)$ where the upper constraint curve $C_{A,U}(\Omega)$ equals the lower constraint curve $C_{A,L}(\Omega)$ for $\Omega \in [0, 0.0046]$.

For ease of implementation the filter structure was set out to consist of four biquad stages. The magnitude at $\Omega = 0.0046$ was set to be -57dB.

All endeavours to use standard filter design tools failed due to strong violations of the constraint curves when the coefficients were set to their finite wordlength. This failure was due to the two-step approach taken by the tools where the coefficients were determined with infinite precision in the first step and quantization in the second. So a design using DE was undertaken where the coefficient quantization was taken directly into account.

### 5.1 DETERMINING THE COST FUNCTION

The first and foremost step in transforming a design task into a minimization task is to find the pertinent cost function. The transfer function in Z-transform domain is given by

$$H(z)=\frac{U(z)}{D(z)}=\frac{\sum_{n=0}^{N} a_n\cdot z^{-n}}{1+\sum_{m=1}^{M} b_m\cdot z^{-m}}=A_0\,\frac{\prod_{n=0}^{N-1}\left(z-z_{0,n}\right)}{\prod_{m=0}^{M-1}\left(z-z_{p,m}\right)} \tag{5.1}$$

where the magnitude is defined by

$$A(\Omega)=\left|H\left(e^{\iota 2\pi\cdot\Omega}\right)\right|=\sqrt{\mathrm{Re}\left(H\left(e^{\iota 2\pi\cdot\Omega}\right)\right)^2+\mathrm{Im}\left(H\left(e^{\iota 2\pi\cdot\Omega}\right)\right)^2} \tag{5.2}$$

In Figure 5.1 the magnitude $A(\Omega)$ has to obey a certain tolerance scheme defined by an upper constraint $C_{A,U}(\Omega)$ and a lower constraint $C_{A,L}(\Omega)$. The deviation from these constraints are measured by equidistant samples along the $\Omega$-axis following the principle shown in Figure 5.2.

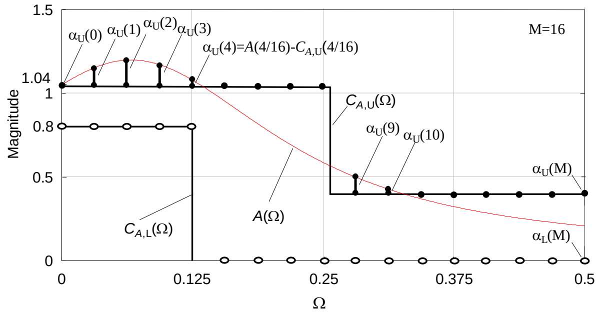


**Figure 5.2:** Example tolerance scheme for the magnitude $A(\Omega)$ to illustrate the constraints $C_{A,U}(\Omega)$ and $C_{A,L}(\Omega)$ [13].

The cost function is constructed according to

$$f(\mathbf{x})=\sum_{i=1}^{i_{\max,U}} s\left(\alpha_U(i)\right)\cdot\alpha_U^2(i)+\sum_{j=1}^{j_{\max,L}} s\left(\alpha_L(i)\right)\cdot\alpha_L^2(i) \tag{5.3}$$

with the deviations

$$\alpha_U(i)=\max\left(20\cdot\log_{10}\left(C_{A,U}\left(\Omega_{i,U}\right)\right)-20\cdot\log_{10}\left(A\left(\Omega_{i,U}\right)\right),\quad 0\right) \tag{5.4}$$

$$\alpha_L(i)=\max\left(20\cdot\log_{10}\left(A\left(\Omega_{j,L}\right)\right)-20\cdot\log_{10}\left(C_{A,L}\left(\Omega_{j,L}\right)\right),\quad 0\right)$$

and the step function

$$s(\delta)=1 \quad \text{for} \quad \delta\geq 0 \quad \textit{and} \quad 0 \quad \text{otherwise.} \tag{5.5}$$

The parameters to be adjusted were chosen to be the poles $z_{p,m}$ and the zeros $z_{0,n}$ rather than the coefficients themselves since stability could be ensured simply by using bounce back. The cost function was computed according to Equation (5.3) where the deviation was determined using a quantized version of $A(\Omega)$. This was achieved by determining the biquad coefficients from the poles and zeros and quantizing them before plugging the result into Equation (5.4), and (5.3).

The ensuing cost function f(**x**) had many of the unpleasant properties mentioned above, i.e. a high dimensionality D=17 due to 4 pole angles, 4 pole radii, 4 zero angles, and 4 zero radii, as well as one gain factor $A_0$. In addition f(**x**) exhibited non-differentiality due to the constraints and the coefficient quantization, as well as multimodality.

## 5.2 RESULTS

For convenience the tool FIWIZ [19] was used to solve the Gaussian filter problem as it already provides DE-based filter design capability. In order to achieve fast convergence the base vector in this tool is chosen according to Equation (3.4). In addition both dither and jitter as in Equation (3.6) and Equation (3.7) are employed. Crossover as defined in Equation (3.5) is only used lightly by setting Cr=0.95. Due to these diversity enhancement measures the population size could be kept low at Np=30. With 28230 function evaluations the result shown in Figure 5.3 has been achieved.

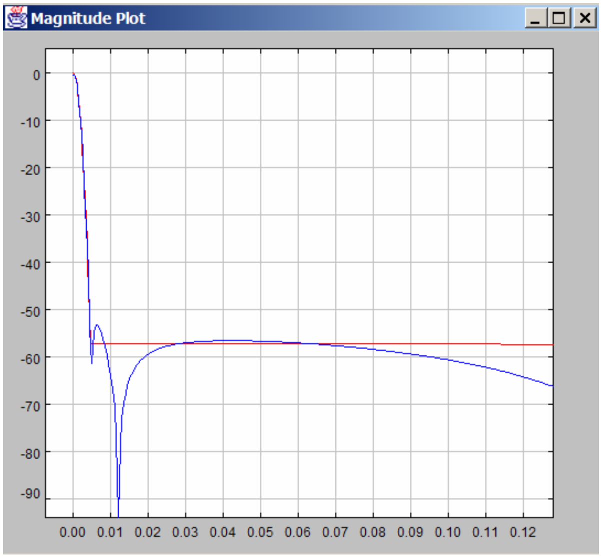


**Figure 5.3:** Result of the magnitude transfer function after 28230 function evaluations for an 8th order biquad structure with 32 bits as a coefficient wordlength.

It can be seen in Figure 5.3 that the constraints in the stopband could not be completely fulfilled. Still this result was more acceptable than an increase in filter order which would have been necessary otherwise.

The filter coefficients for the ensuing biquad stages are listed below in Matlab-compatible format.

Numerator coefficients:
a0=[1.0 -1.9911251696757972 0.9926184536889195];
a1=[1.0 -0.721676564309746 0.18678478291258216];
a2=[1.0 -1.9880436742678285 12.484005219303071];
a3=[1.0 -3.3593827835284173 2.8323051296174526];

Overall amplification according to Equation (5.1):
A0=7.229733026078507E-6;

Denominator coefficients:
b0=[1.0 0.9818580755963922 0.7655303096398711];
b1=[1.0 -1.9761409498751163 0.9766676407307386];
b2=[1.0 -1.978497477248311 0.9786536265164614];
b3=[1.0 -0.3446712060831487 0.029700732324272394];

## 6. CONCLUSION

The global optimization method *Differential Evolution* (DE) has been introduced and numerous references for its applicability to design problems in the wireless communication domain have been provided. By example it has been shown that DE is suitable to find the global mimimum of constrained cost functions with several or even many minima, non-differential regions and high nonlinearity. One of the biggest assets of DE are its simplicity and its ease of use since DE is mostly self-steering. As a concrete example the DE-based design of an $8^{th}$ order digital filter with Gaussian magnitude response, used for a channel simulator, and exhibiting 17 free parameters and a coefficient wordlength of 32 bits has been laid out. Other approaches that first designed the filter with infinite precision and then applied the coefficient quantization afterwards have failed altogether.

## 7. REFERENCES


[1] L.E. Scales, *Introduction to Non-Linear Optimization*, MacMillan, London, 1985.

[2] K. V. Price, R. Storn, J. Lampinen, *Differential Evolution – A Practical Approach to Global Optimization*, Springer, Berlin, Heidelberg, New York, 2005.

[3] D. Kurup, *Active and Passive Unequally Spaced Reflect-Arrays and Elements of RF Integration Techniques*, PhD Dissertation, Uppsala, 2003.

[4] S. Yang, Y. B. Gan, and A. Qing, “Sideband suppression in time-modulated linear arrays by the differential evolution algorithm,” *IEEE Antennas Wireless Propag. Lett.*, vol. 1, pp. 173–175, 2002.

[5] Kurup D G, Himdi M, Rydberg A. “Synthesis of uniform amplitude unequally spaced antenna arrays using the differential evolution algorithm”, *IEEE Trans. Antennas Propagation*, vol. 51(9), pp. 2 210-2 217, 2003.

[6] S. Yang, Y. B. Gan, A. Qing, “Antenna array pattern nulling using a differential evolution algorithm”. Int. J. RF and Microwave Comput. Aided Eng., vol. 14(1), pp. 57-63, 2004.

[7] P. Vancorenland, et alii, “A Layout-Aware Synthesis Methodology for RF-Circuits”, *ICCAD 2001*, pp. 358, 2001.

[8] K. Francken, P. Vancorenland, G. Gielen, “DAISY: a simulation-based high-level synthesis tool for ΔΣ-modulators”, *IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000*. pp. 188 - 192, 2000.

[9] D. A. Crutchley, M Zwolinski, "Globally Convergent Algorithms for DC Operating Point Analysis of Nonlinear Circuits", IEEE *Trans. Evol. Comp*., Vol. 7, No. 1, pp. 2-10, 2003.

[10] Dong Mei Shan, Zhi Ning Chen, Xuan Hui Wu, "Signal optimization for UWB radio systems", *IEEE Trans, Antennas and Propagation*, Vol. 53, No. 7, pp. 2178- 2184, 2005.

[11] P. Limpaphayom, K.A. Winick, "Power- and bandwidth-efficient communications using LDPC codes" *IEEE Trans. Communications*, Vol. 52, No. 3, pp. 350-354.

[12] P. Weitkemper, K. Zielinski, K.-D. Kammeyer, R. Laur: "Optimization of Interference Cancellation in Coded CDMA Systems by Means of Differential Evolution", *Proc. of the 4th Intl. Symp. on Turbo Codes & Related Topics*, 2006, http://www.item.uni-bremen.de/staff/zilli/.

[13] R. Storn, “Designing Nonstandard Filters with Differential Evolution”, *IEEE Signal Processing Magazine*, pp. 103 – 106, 2005.

[14] R. Storn, „Digital Filter Design Software FIWIZ“, http://www.icsi.berkeley.edu/~storn/fiwiz.html, 2000 - 2007.

[15] N. Karaboga, "Digital IIR Filter Design Using Differential Evolution Algorithm", *EURASIP Journal on Applied Signal Processing*, Vol. 2005, No. 8, Pages 1269-1276, 2005.

[16] J. Vondras and P. Martinek, “Multi-criterion filter design via differential evolution method for function minimization”, *In Proceedings. ICCSC ’02. 1st IEEE International Conference on Circuits and Systems for Communications*, pp. 106–109, 2002.

[17] P. Weitkemper, V. Kuhn and K.-D. Kammeyer, "Analysis of Iterative Successive Interference Cancellation in SC-CDMA Systems", in *book "Multi-Carrier Spread-Spectrum"*, Springer, pp. 147-155, 2006.

[18] Das, S., Konar, A., and Chakraborty, U.K. (2005), “Improved differential evolution algorithms for handling noisy optimization problems”, *Proc. IEEE Congress on Evolutionary Computation*, Edinburgh, September 2005.

[19] Storn, R., Digital Filter Design Software FIWIZ, http://www.icsi.berkeley.edu/~storn/fiwiz.html.