Differential Evolution and its Application for the Proposed RELD

Rishabh Kanchu, Anoop Chandrika Sreekantan, Vineeth Bala Sukumaran

Department of Avionics
Indian Institute of Space Science and Technology

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY
THIRUVANANTHAPURAM

August 2025

Contents

Chapter 1

Introduction

Differential Evolution

The differential evolution algorithm is a population-based optimization algorithm designed to solve global optimization problems. The core idea of DE revolves around iteratively improving a population of candidate solutions through a process of mutation, crossover, and selection. Differential evolution (DE) is a kind of simple yet efficient evolutionary computation (EC) algorithm. Due to its simple algorithm implementation and efficient performance in many kinds of optimization problems, the DE algorithm and its variants have been extensively studied [4] [3] and have been applied to many real-world optimization problems [5] [6].

The basic DE algorithm has three key parameters: (a) population size NP i.e., the number of candidate solutions or "parents" maintained in each generation, (b) crossover probability CR which controls the mixing of solutions during recombination, and (c) mutation factor F which determines the scale of differential variation added to candidate solutions. The typical values of these parameters are $NP \geq 4$, $CR \in [0, 1]$, and $F \in [0, 2]$. The typical setting for population size is $10n$, where n is the number of decision variables (the dimensions of the optimization problem). [2]

1.1 Unconstrained Optimization Problem Formulation

An unconstrained optimization problem deals with finding the local minimizer x^* of a real valued objective function $f(x)$, given by $f : R^D \to R$, formulated as,

$$x^* \in R^D \quad \ni \quad f(x^*) \leq f(x) \quad \forall\, x \in R^D \tag{1.1}$$

where D denotes the number of decision variables. In this case, no additional restrictions or conditions are imposed on the variables, and the search is carried out over the entire D-dimensional space.

1.2 Constrained Optimization Problem Formulation

For an objective function $f : X \subseteq R^D \to R$ where the feasible region $X \neq \phi$, the minimization problem is to find

$$x^* \in X \quad \ni \quad f(x^*) \leq f(x) \quad \forall\, x \in X \tag{1.2}$$

subject to one or more constraint functions:

$$g_j(x) \leq 0, \quad j = 1, \ldots, m \tag{1.3}$$

where m is the number of constraints and $f(x^*) \neq -\infty$. Here, the search is limited to the feasible region X that satisfies all given constraints.

1.2.1 Notation

To optimize a function with D real parameters, the population size $NP \geq 4$. The parameter vectors have the form:

$$x_{i,G} = [x_{1,i,G}, \ldots, x_{D,i,G}] \text{ i = 1,...,N} \tag{1.4}$$

where G is the generation number.

1.2.2 Handling Constraint functions

The most commonly used penalty function method is combined with DE to handle constraint functions. The value of the function $f'(x)$ (where $f'(x)$ is the constained objective fucntion) to be minimized by DE can be computed in its simplest form by penalizing the objective function value with a weighted sum of constraint violations, as shown below. [7]

$$f'(x) = f(x) + \sum_{j=1}^{m} W_j \times max(0, g_j(x)) \tag{1.5}$$

The penalty function approach effectively converts a constrained problem to an unconstrained one, and then $f'(x)$ is used as the objective function instead of $f(x)$. This method will add one or more control parameters, penalty parameters, or the weights shown in equation 1.5. The user has to pre-define the suitable control parameter values. This level of implementation always requires some additional work, and selecting a good setting may be a challenging, if not impossible, process. Several DE trial runs are required to set the penalty parameters through trial and error.

1.2.3 Initialization

The upper and lower bounds for each parameter are given by

$$x_j^L \leq x_{j,i,1} \leq x_j^U \tag{1.6}$$

A population of vectors (candidate solutions) is randomly generated with a uniform distribution within specified bounds $[x_j^L, x_j^U]$.
Each vector represents a possible solution for the parameter set.

1.2.4 Mutation

Each of the N parameter vectors undergoes mutation, recombination, and selection.

For a given parameter vector $x_{i,G}$ randomly select three vectors $x_{r_1,G}$, $x_{r_2,G}$ and $x_{r_3,G}$ such that the indices i, r_1, r_2 and r_3 are distinct. The vector obtained by adding the weighted difference of two of the vectors to the third is known as the donor vector and is given by [8]

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} - x_{r_3,G}) \tag{1.7}$$

1.2.5 Recombination

The trial vector $u_{i,G+1}$ is developed from the target vector, $x_{i,G}$, and the donor vector, $v_{i,G+1}$. Elements of the donor vector enter the trial vector with probability CR. [8]

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } rand_{j,i} \leq CR \text{ or } j = I_{rand} \\ x_{j,i,G} & \text{if } rand_{j,i} > CR \text{ and } j \neq I_{rand} \end{cases} \tag{1.8}$$

$$i = 1, \ldots, N; \quad j = 1, \ldots, D \tag{1.9}$$

Where $rand_{j,i} \sim U[0,1],\ I_{rand}$ is a random integer from [1, 2, ..., D].

1.2.6 Selection

The target vector $x_{i,G}$ is compared with the trial vector $v_{i,G+1}$, and the one with the lowest function value is given to the next generation. [8]

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \quad \text{where } i = 1, \ldots, N \tag{1.10}$$

Mutation, recombination and selection continue until some stopping criterion is reached.

1.2.7 Termination Conditions

Termination conditions in an optimization algorithm, including Differential Evolution, are criteria used to determine when the algorithm should stop running. The choice of termination conditions depends on the specific problem being solved and the goals of the optimization. We usually want a termination condition that ensures our solution is close to optimal at the end of the run. The possible termination conditions are given:

1. The algorithm stops after a predefined number of iterations have been completed.
2. When the population has not improved after X iterations. The algorithm can be set to terminate when the population converges, meaning that there is little or no improvement in the fitness of the best solution over a certain number of iterations/generations.
3. When the constraint violation occurs.
4. When an absolute number of generations is reached.
5. Termination can be triggered when the fitness of the best solution in the population reaches a predefined threshold. This threshold represents a desired level of performance or the solution quality you aim to achieve.

1.3 DE Algorithm Pseudocode

Algorithm 1 Pseudocode of the DE Algorithm for a Minimization Problem

```
Randomly generate the population of size $NP$
Do while
For each individual $j$, in the population undergoes mutation, crossover, and selection
**Mutation**
Generate three random integers, $r_1, r_2, r_3 \in (1, NP)$, with $r_1 \neq r_2 \neq r_3 \neq j$
$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} - x_{r_3,G}) \tag{1.11}$$
**Recombination**
Generate a random integer $I_{rand} \in (1, N)$
For each parameter $i$
$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } rand_{j,i} \leq CR \text{ or } j = I_{rand} \\ x_{j,i,G} & \text{if } rand_{j,i} > CR \text{ and } j \neq I_{rand} \end{cases} \tag{1.12}$$
$$i = 1, \ldots, N; \quad j = 1, \ldots, D \tag{1.13}$$
**Selection**
The target vector $x_{i,G}$ is compared with the trial vector $u_{i,G+1}$.
$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \tag{1.14}$$
**if** $u_{i,G+1}$ **is better then**
    Replace $x_{i,G}$ with the $u_{i,G+1}$
**else**
    Repeat mutation, recombination and selection until the Termination Criteria is satisfied
```

Chapter 2

Application of DE in Proposed RELD

Linear Variable Differential Transformers (LVDTs) are electromechanical sensors widely employed for precise linear displacement and position measurement across a broad spectrum of industries. With appropriate accessories, they can also be adapted for applications such as pressure measurement, weight determination, and liquid level sensing.[1] The LVDTs are inherently nonlinear over a wide displacement span. To overcome this, the proposed Range-Extending LVDT Digitizer (RELD) introduces a linearizing function whose parameters are optimized using the Differential Evolution (DE) algorithm.

2.1 Goal of Optimization

The goal is to improve the linearity of the LVDT's output over a wider span without modifying the physical structure of the LVDT. Instead, we apply mathematical compensation via a linearizing function, whose parameters are tuned for best performance.

2.2 Optimization problem of RELD

The digitizer output is expressed as a ratiometric nonlinear function:

$$F = \frac{T_2}{T_1} \cdot \mathrm{sgn}(V_c(T_1))$$

where T_1 and T_2 are integration and deintegration periods, respectively. However, F becomes nonlinear over a large displacement range. To correct this, a composite function F_L is introduced:

$$F_L(x) = K_1 \left(\frac{F}{F + A_1} \right) + K_2 \left(\frac{F}{F + A_2} \right)$$

Here, A_1, A_2 are shaping constants, and K_1, K_2 are gain weights. These four parameters form the optimization vector:

$$P = (A_1, A_2, K_1, K_2)$$

2.3 Motivation behind using DE for RELD optimiztion

The relationship between F and displacement x is nonlinear, making it difficult to directly invert or apply standard analytical curve-fitting methods to find an exact transformation for linearization. The chosen linearizing function $F_L(x; P)$ depends on multiple coupled parameters, and their effect on the output is highly nonlinear. Hence, traditional gradient-based optimization methods are not suitable, while a derivative-free global search method such as Differential Evolution can efficiently determine the optimal parameters.

DE is ideal because:

- It does not require derivatives of the objective function.
- It can efficiently explore high-dimensional, non-convex, and multi-modal landscapes, reducing the risk of getting trapped in local minima.
- It handles constraints well.

However, some limitations of DE include:

- Its convergence speed can be slower than that of gradient-based methods when the objective is smooth and differentiable.
- DE has a strong ability to find near-global optima in practical applications, but it does not offer a mathematical guarantee of convergence to the exact global optimum. Its performance depends on adequate population diversity, control parameter tuning (mutation factor F, crossover probability CR, population size NP), and the complexity of the search space.

2.4 Optimization Objective

We define the objective function to be the normalized peak nonlinearity:

$$NL(P) = \frac{\max_i |F_L(x_i; P) - y_{BF}(x_i)|}{\max_i F_L(x_i; P) - \min_i F_L(x_i; P)} \times 100$$

where $y_{BF}(x_i)$ is the best-fit straight line (with slope m and intercept c) through the data points $F_L(x_i; P)$.

This objective function reflects how closely the transformed output F_L aligns with a perfect linear behavior.

2.5 Constraints in Optimization

Two key constraints are imposed:

- $|m| \geq 0.1$ — ensures that the optimized output does not result in a near-flat line, which would lose sensitivity.
- $0.0001 \leq NL(P) \leq 100$ — ensures valid nonlinearity computation and avoids degenerate solutions.

2.6 Optimization Strategy Using DE

The DE algorithm explores the parameter space of $P = (A_1, A_2, K_1, K_2)$ to minimize $NL(P)$. It maintains a population of solutions, applies mutation and crossover, and selects candidates that yield lower nonlinearity. Over several generations, it converges toward a globally optimal (or near-optimal) solution.

The Python implementation is configured as: [9]

```
bounds = [(-100, 100), (-100, 100), (-1, 1), (-1, 1)]
constraints = (nlc1, nlc2)
result = differential_evolution(to_minimise,
                                bounds,
                                constraints=constraints,
                                maxiter=500,
                                popsize=25,
                                polish=False)
```

- `bounds`: These define the search space for each of the four optimization parameters:
  - $A_1, A_2 \in [-100, 100]$: These parameters influence the curvature of the linearizing function. The wide bounds allow DE to explore both small and large shaping constants.
  - $K_1, K_2 \in [-1, 1]$: These parameters serve as gain weights for each term in the linearizing function. Gains beyond this range are unnecessary due to the bounded nature of the function $\frac{F}{F+A}$, and to maintain numerical stability.
- `constraints`: A tuple containing nonlinear constraints applied during the optimization process.
  - `nlc1`: Ensures the slope of the best-fit line $|m| \geq 0.1$, so that the output retains sensitivity and doesn't flatten.
  - `nlc2`: Ensures the computed nonlinearity stays within a valid range $[0.0001, 100]$, rejecting degenerate or invalid solutions.
- `maxiter`: This is the maximum number of generations (iterations) the DE algorithm will execute. A higher number ensures thorough exploration of the search space and convergence toward optimality.
- `popsize`: Multiplier for the population size. The total number of candidates in the population is `popsize × len(bounds)` (4 in this case). Hence, actual population size is $4 \times 25 = 100$. A large population helps avoid premature convergence and allows robust search in high-dimensional or rugged landscapes.
- `polish`: Setting `polish=False` disables post-processing using a local optimizer. This ensures the final solution is purely the result of global DE search.

Chapter 3

Results and Discussion

The optimized parameters (example values: $A_1 = 0.856$, $A_2 = -1.151$, $K_1 = 0.401$, $K_2 = -0.964$) yield a linearized output with a nonlinearity less than 0.18% over an 80 mm displacement span. This is an 16-fold improvement in linearity and 2-fold enhancement in usable range compared to the native LVDT output. The optimization process is performed separately for each LVDT model, enabling the linearizing function to be precisely matched to the unique characteristics of the sensor and ensuring consistent high performance across different LVDT types and applications. The DE-based approach enabled these results without requiring complex analog circuits, additional ADCs, or logarithmic amplifiers — making it low-cost, simple, and effective.

Differential Evolution has played a critical role in the RELD architecture by providing a means to mathematically correct nonlinearity through efficient global optimization. This approach enhances LVDT applicability in precision displacement measurement systems.

Bibliography

[1] Techno Instruments, “Linear Variable Differential Transformer,” Accessed: May 22, 2024. [Online]. Available: `https://technoinstruments.in/product-details/178/LVDT-01`

[2] K. Storn and R. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Optimization*, vol. 11, pp. 341–359, 1997.

[3] J. Zhang, H. Chung, and W. L. Lo, “Clustering-based adaptive crossover and mutation probabilities for genetic algorithms,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 326-335, Jun. 2007.

[4] Z H. Zhan and 1. Zhang, "Enhance differential evolution with random walk," in *Proc. Genetic Evol. Comput. Coni*, Philadelphia, America, Jul. 2012, pp. 1513-1514.

[5] Z. -H. Zhan and J. Zhang, "Differential evolution for power electronic circuit optimization," *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Tainan, Taiwan, 2015, pp. 158-163, doi: 10.1109/TAAI.2015.7407129.

[6] P. P Menon, 1. Kim, D. G. Bates, and L Postlethwaite, "Clearance of nonlinear flight control laws using hybrid evolutionary optimization," *IEEE Trans. Eva!. Comput.*, vol. 10, no. 6, pp. 689-699, Dec. 2006.

[7] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Honolulu, HI, USA, 2002, pp. 1468-1473 vol.2, doi: 10.1109/CEC.2002.1004459.

[8] Wong, Kit & Dong, Z.Y.. (2005). Differential Evolution, an Alternative Approach to Evolutionary Algorithm. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 73–83. IEEE, 2005. 10.1109/ISAP.2005.1599244.

[9] P. Virtanen, et al., “scipy.optimize.differential_evolution,” *SciPy v1.14.0 Manual*, 2025. [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html.`