CentraleSupélec
Mention Control Engineering
2021-2022

Engineering degree
CentraleSupélec
Second year, ST7
2021-2022

Optimization

Tutorial 5 – Metaheuristics Optimization

Example of the Differential Evolution Algorithm

1 Objectives of the tutorial and presentation of metaheuristics

The objective of this tutorial is to provide a brief introduction on a class of optimization algorithms named "Metaheuristics", or stochastic optimization algorithms. The idea is to add a part of random search so that the algorithm can escape from local minima.

Some common features can be observed for these algorithms:

- There are usually inspired by some physical or biological phenomena;
- They use a part of random search;
- They only use the evaluations of the cost function and do not require some local information such as gradient or Hessian computation.

Of course, it has to be noticed that these algorithms should not be used when deterministic and efficient algorithms do exist to solve the optimization problem. They have to be considered as an alternative to face with difficult optimization problems (numerous local minima, lack of analytical expression for cost and constraint functions…).

There are some counterparts in the use of these algorithms:

- Usually, there is no guarantee to obtain (at least) a local minimum;
- Due to the random process, the algorithm may provide different solutions for two runs on the same problem.

It exists several metaheuristics algorithms (new ones appear each year). Among the most famous ones are the Genetic Algorithm, the Differential Evolution, the Particle Swarm Optimization, the Ant Colony algorithm…

2 Differential Evolution

In this tutorial, we are going to use the Differential Evolution (DE). The description of the algorithm is given in Appendix. The notations in the document will be those used in the Appendix.

In order to get feedback while the algorithm is running, it is recommended to have some information displayed in real time[1]:

- The g value of the current generation (possibly as a percentage of the total number of generation G) ;
- The best value of the criterion found since the beginning of the search;
- The index n which corresponds to the best value found at generation g (the frequent changes of indexes provide information on the richness of the research).

The parameter values to be used to start this exercise are given in the Appendix. Unless otherwise stated, the variables to be optimised are between -10 and +10 and the examples are to be treated in dimension 8.

[Q01] Implement the algorithm described in Appendix[2]. At the end of the run, display the calculation time, the optimum found and the associated criterion value. Illustrate your results with an example.

To do this, fill in the main_DE.m file where the key variables are defined and the first parameter values are given.

3 Test functions

Some test functions $\mathbb{R}^D \to \mathbb{R}$ are given:

- f_sphere.m
- f_nondiff.m
- f_nondiff2.m
- f_nondiff2asym;
- f_rastrigin.m

The program MAIN_visufunction2D.m can be used for visualisation.

[Q02] Analyse these functions. Give their expression in concise form. What are their minima? For each function, what is its interest in testing optimisation algorithms?

Formal demonstrations of properties are not required here. However, factual arguments to support the stated results are welcome (the kind of argument that can be formalised into a proof elsewhere).

4 Behavioural analysis of the criterion

It is important to visualise the behaviour of the algorithm in more detail[3] in order to get an idea of its behaviour. This makes it possible to better judge the quality of the result and, possibly, helps to adjust parameters. For this purpose, the following values can be observed as a function of the generation g :

$$f(P_g) = \left\{ f\left(x_1^{(g)}\right), f\left(x_2^{(g)}\right), \cdots, f\left(x_n^{(g)}\right), \cdots, f\left(x_N^{(g)}\right) \right\}$$

[1] *Available functions: fprintf, disp… or simply do not put a ";".*

[2] *For debugging, it is advisable to test as you go along using the functions that will be studied more closely in the following questions. Typically, use the function f_sphere*$(x) = \sum_{n=1}^{N} (x_i - 1)^2$ *from the "f_sphere.m" file.*

[3] *For this project, this can be done a posteriori, after the end of the optimisation phase (this should allow you to keep your code simpler).*

It is possible to plot for each individual in the population:

$$g \to f\left(x_n^{(g)}\right)$$

However, this gives a set of N curve which is difficult to use. It is often preferable to plot a limited number of relevant quantities characterising the population at each generation: mean, median, extreme values, sigma…

[Q03] Plot, as a function of the iteration number g, the best and the worst value known so far and the mean value of the costs in the population. Enrich this plot so that it also includes information representing the population spread. Illustrate with some examples[4] in dimension 8.

Work on your graphic representation to make it meaningful: this will help you to better analyse the behaviour.

5 Behavioural analysis of the population

The representation of the evolution of the population is less simple since[5] we are dealing with a collection of vectors:

$$P_g = \left\{x_1^{(g)}, x_2^{(g)}, \cdots, x_n^{(g)}, \cdots, x_N^{(g)}\right\}$$

The idea is to reduce it to a family of scalar quantities:

$$h_g(P_g) = \left\{h_g\left(x_1^{(g)}\right), h_g\left(x_2^{(g)}\right), \cdots, h_g\left(x_n^{(g)}\right), \cdots, h_g\left(x_N^{(g)}\right)\right\}$$

then apply the same approach as for the analysis of the evolution of the criterion.

Let us define the distance to the mean position at generation g :

$$\overline{x^{(g)}} = \frac{1}{N}\sum_{n=1}^{N} x_n^{(g)} \quad and \quad h_g(x) = \left\| x - \overline{x^{(g)}} \right\|$$

[Q04] Interpret this indicator $h_g(x)$. Develop a graphical representation to analyse the evolution of the population as a function of g. Illustrate with some examples[6] in dimension 8.

6 Behaviour in the presence of local minima

Use function f_nondiff2. Define G to ensure that the algorithm "almost always" converges [7] (aim for a success rate of around 95%). Use the simple criterion: $\left|f_{opt} - 0\right| < 0.01$ to check for convergence.

Several optimisations will always be observed to get a global view of the behaviour. Identify the frequent cases, the unusual behaviours...

[4] *Use function f_sphere(x) in* $\mathbb{R}^8$.

[5] *Except in dimension 1 or 2.*

[6] *Use function f_sphere(x) in* $\mathbb{R}^8$.

[7] *Do not try to have "always" the convergence, because it would be very expensive in time.*

[Q05] **In cases of convergence, does the criterion always converge to the expected value? Explain and justify. Calculate a statistic of the number of cases converging for your parameter values.**

The notion of convergence of a group of vectors can take several aspects. All the vectors may converge more or less quickly to the same value (possibly not the one desired !) or only a part of them may be concerned by the convergence… Use the analysis tools you have developed to understand what the behaviour is in this case.

[Q06] **In cases of convergence, analyse the final behaviour of the population. Does it always converge to the same value? Produce statistics to quantify your conclusions.**

Sometimes the population remains very spread out before suddenly converging.

[Q07] **Give a graphical example of such behaviour. Give an interpretation.**

7 Effects of parameters

The different effects of the parameters are not decoupled. Also, their choice is not trivial. For example, a large increase in the population provides better coverage (or exploration) but requires many more generations to mix all the genes in the whole population and finally obtain convergence. To obtain a better overall behaviour, it would then be necessary to simultaneously adjust the other parameters... For this study, we will take a pragmatic approach and vary only one parameter at a time.

For each parameter N, F, C (individually), test large and small values (see table below).

Parameter	Nominal (for this study)	Small values	Large values
N	$\text{floor}\left(10+\sqrt{D}\right)$	$< D$	$10 \times D$
F	0.8	0.1	0.95
C	0.7	0.1	0.98

G will not be modified (although this may sometimes allow for convergence). Instead, try to analyse how the population evolves (coverage of space) and how the criteria evolve (be careful, it may not converge towards the optimum).

For this question, use function f_nondiff2 is always used.

[Q08] **Vary the parameters shown in the table above. Observe the behaviour. Propose an analysis.**

8 For fun!

[Q09] **Use the DE algorithm on function f_nondiff2asym. Adjust the parameters to almost always achieve convergence to the global optimum. Illustrate your results.**

[Q10] **Use the DE algorithm on function f_rastrigin. Adjust the parameters to almost always achieve convergence to the global optimum. Illustrate your results. What is the price to be paid to improve comportment?**

9 Optimization of a controller for a magnetic levitation system

9.1 The system

We consider now a magnetic levitation system such as the one represented in figure 1 (photograph of the real system, and schematic representation). The goal is to control the position of a pendulum thanks to the magnetic force of a magnet.

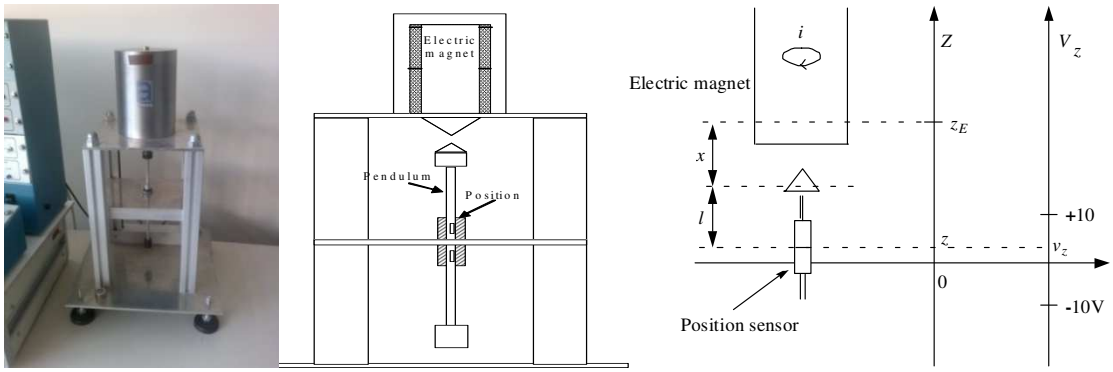

Fig. 1 – Magnetic levitation system

A linear model and block diagram of the system is given in figure 2. s is the Laplace variable.

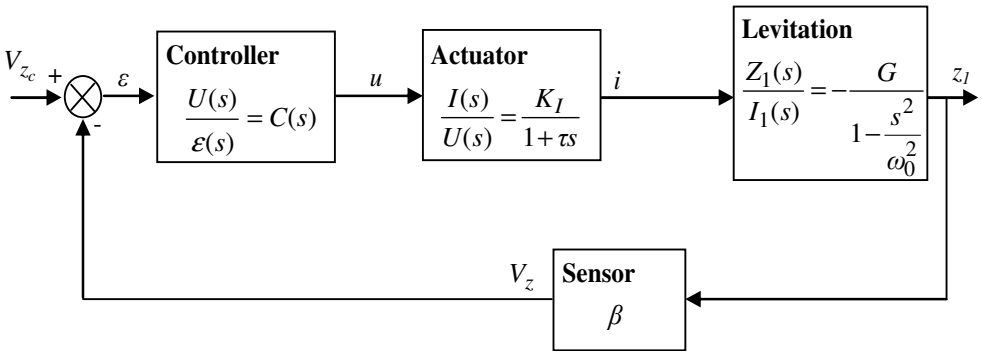

Fig. 2 - Linear model of the magnetic levitation in a closed loop framework.

The current in the levitation system is delivered by an actuator whose transfer function is:

$$\frac{I(s)}{U(s)} = \frac{K_I}{1+\tau s}$$

The transfer function of the magnetic levitation is given by:

$$\frac{Z(s)}{I(s)} = -\frac{G}{1-\frac{s^2}{\omega_0^2}}$$

The position of the pendulum is measured by a sensor considered as a gain β .

Numerical values of the model parameters are: $K_I = 0.1A/V$, $\tau = 5.10^{-4}s$, $G = 0.01m/A$, $\omega_0 = 40rad/s$, $\beta = 4000V/m$. To control the system, we look for a classical PID controller with high frequency filtering:

$$C(s) = K\left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \tau_d s}\right) \cdot \frac{1}{1 + T_f s} \quad \text{with } \tau_d = T_d/10$$

The parameters to be tuned are thus $\theta = \left[K, T_i, T_d, T_f\right]^{\mathsf{T}}$.

To enhance the sensitivity of the algorithm with respect to small values of the parameters, we consider the optimization variables defined by a logarithmic change of variables, that is:

$$x = log_{10}(\theta) = \left(log_{10}(K), log_{10}(T_i), log_{10}(T_d), log_{10}(T_f)\right)^T$$

The following Matlab function is provided:

[stab, tr, D, delta_phi, u_max] = simul_magnetic(x, trace)

With:

- x: logarithmic values of the parameters;
- trace: Boolean value. If trace is equal to 1, the step response of the closed loop, the corresponding control signal u and the corrected open-loop Bode diagram are plotted. Otherwise, the performance indicators are evaluated without any plotting.
- stab: maximum real value of the closed loop poles. Consequently, this value is positive if the closed loop is unstable, negative otherwise.
- tr: 5% time response if the closed loop is stable, $+\infty$ otherwise.
- D: overshoot of the step response if the closed loop is stable, $+\infty$ otherwise.
- delta_phi: phase margin of the controlled system if the system is stable, $-\infty$ otherwise.
- u_max: $\max_t |u(t)|$ for the step response if the system is stable, $+\infty$ otherwise.

9.2 Initial observations

We give the following stabilizing controller:

$$C_0(s) = \left(1 + \frac{1}{0.1s} + \frac{0.02s}{1 + 0.002s}\right) \cdot \frac{1}{1 + 0.001s}$$

that is $K = 1, T_i = 0.1s,\ T_d = 0.02s,\ T_f = 0.001s$.

Utiliser le fichier : test_function.m.

[Q11] Evaluate the performance of the corresponding closed loop (time-response, overshot, phase margin and maximum of the control signal).

9.3 Optimize the time response

[Q12] Optimize the time response of the closed loop system thanks to your DE algorithm. What is the obtained maximum control value?

The initial population should be defined in the following range:

$$K \in [0.1,100], T_i \in [0.01,1],\ T_d \in [0.001,0.1],\ T_f \in [0.0001,0.01]$$

Be careful! The time response can only be evaluated if the closed system is stable. In addition, it is important to give a relevant movement direction when the system is unstable. It is suggested to use the following function to be optimized (give the corresponding justification):

$$f_1(x) = \begin{cases} \max(\, Re(\,\text{poles}(BF(x)))) & \text{if the closed loop is unstable} \\ -\dfrac{1}{t_r(x)} & \text{if the closed loop is stable} \end{cases}$$

9.4 Add a physical limitation

The physical limitations of the system are given by $\max|u(t)| \leq 10V$. We propose to use a penalty approach to take into account this constraint. We consider the following cost function:

$$f_2(x) = \begin{cases} \max(\, Re(\,\text{poles}(BF(x)))) & \text{if the closed loop is unstable} \\ -\dfrac{1}{t_r(x)+e^{\left(\lambda\frac{\max|u(t)|-10}{10}\right)}} & \text{if the closed loop is stable} \end{cases}$$

Justify the choice of this function, and optimize it with your DE algorithm (choose for instance $\lambda = 100$).

9.5 Extension

Extend the previous methodology to take into account the constraint $\Delta\varphi \geq 45°$.

Appendix : Differential Evolution[8,9] (DE)

Differential evolution (DE) searches for the optimum of a function $f:\Omega \subset \mathbb{R}^D \to \mathbb{R}$ by evolving a population of N vectors in the search space Ω for G generations. The size N of the population is fixed throughout the minimization process.

Notations. For each generation $g \in \{1,2,\dots,G\}$, the population is noted:

$$P_g = \left\{x_1^{(g)}, x_2^{(g)}, \cdots, x_n^{(g)}, \cdots, x_N^{(g)}\right\} \in (\mathbb{R}^D)^N$$

The D components of the vector n of the generation g are noted[10]:

$$x_n^{(g)} = \left[x_{n,1}^{(g)}, \dots, x_{n,i}^{(g)}, \dots, x_{n,D}^{(g)}\right]$$

Feasible space. Ω is a parallelogram defined by the minimum m_i and maximum M_i values characterizing its boundary in each direction $i \in \{1,2,\dots,D\}$ of the space. Thus $x_n^{(g)} \in \Omega$ verifies:

$$\underbrace{\forall g \in \{1,\dots,G\} \quad \underbrace{\forall n \in \{1,\dots,N\} \quad \underbrace{\forall i \in \{1,\dots,D\} \; m_i \le x_{n,i}^{(g)} \le M_i}_{All\ x_{n,i}^{(g)}\ components\ of\ x_n^{(g)}}}_{All\ vector\ x_n^{(g)}\ of\ P_g\ population}}_{All\ P_g\ population \Leftrightarrow All\ successive\ generations}$$

Initial population. The initial population[11] $P_1 = \left(x_1^{(1)}, \cdots, x_n^{(1)}, \cdots, x_N^{(1)}\right)$ is randomly chosen in Ω. For this choice, and more generally for all random choices involved in DE; the distributions to be used are uniform.

Overview of algorithm. The algorithm evolves the population by a sequence of three transformations:

$$P_g \longmapsto \boxed{[1]\ Mutation} \longmapsto \boxed{[2]\ Crossover} \longmapsto \boxed{[3]\ Selection} \longmapsto P_{g+1}$$

The cycle of these three transformations is repeated G times to generate all generations:

$$P_1 \longmapsto P_2 \longmapsto \cdots \longmapsto P_g \longmapsto \cdots \longmapsto P_G$$

The following gives details of these transformations.

---

[8] *STORM R. M, PRICE K., "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR9 -012, International Computer Science Institute, Berkeley, California, 1995.*

[9] *STORM R. M, PRICE K., "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". Journal of Global Optimization 11, 341–359 (1997).*

[10] *Note. For a given generation g, the family P_g can be stored in a matrix* $M^{(g)} = \begin{bmatrix} x_{1,1}^{(g)} & x_{1,2}^{(g)} & \cdots & x_{1,D}^{(g)} \\ x_{2,1}^{(g)} & x_{2,2}^{(g)} & \cdots & x_{2,D}^{(g)} \\ \vdots & \vdots & & \vdots \\ x_{N,1}^{(g)} & x_{N,1}^{(g)} & \cdots & x_{N,D}^{(g)} \end{bmatrix} \in \mathbb{R}^{N \times D}$

[11] *In the case where a preliminary solution is available, it is possible to take it into account by including it in the initial population P_1 which will be completed by randomly chosen vectors in Ω (either [1] uniformly in Ω, or [2] around the preliminary solution – in this latter case, the deviations from this preliminary solution follow a normal distribution).*

[1] Mutation for each $n \in \{1, \dots, N\}$:

— Random selection of 3 different vectors of P_g: $X_1 = x_{r_1}^{(g)}, X_2 = x_{r_2}^{(g)}, X_3 = x_{r_3}^{(g)}$.
  - To ensure that these vectors are always different from $x_n^{(g)}$, one must choose r_1, r_2, r_3 different from each other and different from n. The selection must be equiprobable sur $\{1,2, \dots, D\}\backslash\{n\}$.
  - Note that this implies $N \geq 4$.

— Generation of a mutant vector $\Rrightarrow v_n^{(g+1)} = X_1 + F(X_2 - X_3)$.
  - F governs the use of the differential variation $X_2 - X_3$. Guidelines for the choice of F are given in the following.

[2] Crossover for each $n \in \{1, \dots, N\}$:

— Principle : creation of a test vector: $u_n^{(g+1)} = \left[u_{n,1}^{(g+1)}, u_{n,2}^{(g+1)}, \dots, u_{n,D}^{(g+1)}\right]^{\mathsf{T}}$ by crossing the original vector $x_n^{(g)}$ and the mutant vector $v_n^{(g+1)}$.

— To ensure that there is always at least one active mutation, a particular index $\mathbb{I}$ is chosen first (this index will necessarily be associated with a mutation).

— For each component $i \in \{1, \dots, D\}$:
  - $u_{n,i}^{(g+1)} = \begin{cases} v_{n,i}^{(g+1)} & si\ (rand \leq C)\ or\ i = \mathbb{I} \\ x_{n,i}^{(g)} & si\ (rand > C)\ and\ i \neq \mathbb{I} \end{cases}$
    with $rand \in [0,1]$ uniform random generator and C crossover constant used to set the average mutation rate. Guidelines for the choice of C are given in the following.

[3] Selection for each $n \in \{1, \dots, N\}$:

— Orthogonal projection of $u_n^{(g+1)}$ onto Ω
  - for each component $i \in \{1, \dots, D\}$:
    update of values → $u_{n,i}^{(g+1)} = \max(m_i, \min(M_i, u_{n,i}^{(g+1)}))$

— Perform greedy selection (selection of the surviving vector with the best criterion):

$$x_n^{(g+1)} = \begin{cases} u_n^{(g+1)} \text{ if } f\left(u_n^{(g+1)}\right) < f\left(x_n^{(g)}\right) \\ x_n^{(g)} \quad \text{else} \end{cases}$$

Setting the control parameters N, F and C.

The method is tolerant[12] to the setting of these parameters. The following table provides information on the settings and the effects of the settings on the behaviour (there is no consensus on this issue).

Parameters	**Not unreasonable choice !**	**To start the present work**	**↗ (increased value, volume of space)**	**↘ (decrease value, volume of space)**
Ω	As small as possible (being sure that it contains the solution)	$\forall i \in \{1,..,D\}$ $\begin{cases} m_i = -10 \\ M_i = +10 \end{cases}$	larger space: more exploration, generally slower convergence.	smaller space: less exploration, generally faster convergence.
N	If calculation time is not prohibitive: $N \in [5D, 10D]$. Note that lower values often work, and the choice $N =$ floor $(10 + \sqrt{D})$ is often relevant. It is necessary to have $N \geq 4$	To start use $N =$ floor $(10 + \sqrt{D})$.	N ↗: more exploration at each generation. More calculation for each generation. Possibility of parallelizing all the calculations of a generation (reduction of the calculation time by a factor N)	N ↘ less exploration at each generation. Less calculation for each generation.

[12] *This is one of the important qualities that a good optimisation algorithm should have!*

Parameters	Not unreasonable choice !	To start the present work	↗ (increased value, volume of space)	↘ (decrease value, volume of space)
F	$F \in [0.5, 1]$ is often a good initial choice[13].	$F = 0.8$	$F \nearrow$ Increase exploratory behaviour. Slows down convergence (need more generation).	$F \searrow$ Decrease exploratory behaviour and often faster convergence. Greater risk of being trapped in a local minimum.
C	Typical: $C \in [0.1, 0.9]$ C around 0.5 is often a good initial choice. If no convergence can be achieved, $C \in [0.9, 1]$ often helps. On a new problem, it is advisable to test $C = 0.9$ to get a first idea of the behaviour. If a satisfactory quick solution is not achievable → decrease C	$C = 0.7$	$C \nearrow$ large CR often speeds convergence, but give a more chaotic behaviour.	$C \searrow$ More regular behaviour.
Preliminary solution	If available gives reference(s) value(s) of criterion that can only be improved, but by attractiveness of preliminary solution may reduce exploration of the space	Not used	Use of several preliminary solutions: competition and search "around" the combinations of these solutions. Beware, all preliminary solutions may reduce the exploration of space.	To ensure a good exploration of the space, it is necessary to maintain a small number of preliminary solutions compared to N

[13] *STORM R., "On the usage of Differential Evolution for Function Optimization", North American Fuzzy Information Processing Society, p. 519-523, 1996*