EngiMix

OptiMix

User guide

engimix@gmail.com
1-13-2024

Contents

Figures

Symbols

Variables

Symbol	Name	Unit	Unit symbol
c	Coefficient / probability	-	-
C	Specific cost	Dollars / (kilowatt*Hour)	$ / kWh
E	Energy	Kilowatt*Hour	kWh
l	Learning rate	1	1
n	Number	1	1
$ObjFun$	Objective functions vector	-	-
P	Power	Kilowatt	kW
SoC	State of Charge	1	1
V	Speed	-	-
w	Inertia	-	-
X	Decision variables vector	-	-

Subscripts

Symbol	Name
$BESS$	Battery Energy Storage System
$best$	Best
CO	Cross-over
G	Grid (i.e. electricity)
GA	Genetic algorithm
GD	Gradient descent
$Global$	Global
$Guess$	Guess value
$Load$	Electrical load
M	Mutation
max	Maximum
min	Minimum
PSO	Particle swarm optimization
PV	Photovoltaic panel
ZT	Zero-test

Superscripts

Symbol	Name
k	Index

Overview

Definition

OptiMix is a multi-method and multi-objective optimizer based on the following optimization methods:

- Genetic Algorithm (GA);
- Particle Swarm Optimization (PSO) method;
- Gradient Descent (GD) method.

It optimizes (minimizes / maximizes) the objective functions, $ObjFun$, based on the decision variables X.

It can be launched directly on the web (guide), or on MATLAB (guide) and Phyton. Note that using the web version is free (limited number of particles and iterations), easier, quicker, and allows to comfortably manage different optimizations.

Characteristics

The main characteristics of OptiMix are:

- The number of decision variables can be set by the user;
- The number of objective functions can be set by the user;
- The guess value of each decision variable can be set by the user;
- The range of each decision variable can be set by the user;
- The type of the optimization of each objective function can be set by the user;
- The priority of each objective function can be set by the user;
- The number of families (i.e. independent sets of particles) can be set by the user;
- The number of particles can be set by the user;
- Multiple termination criteria (i.e. number of iterations, relative variation, average relative error) can be set by the user;
- The restart frequency can be set by the user;
- The zero-test frequency and the zero-test value of each decision variable can be set by the user. The zero-test consists in setting the decision variable equal to the zero-test value and checking if a better optimum is found. This is particularly relevant during parameters estimation problems, where the user can define a lot of coefficients and run the zero-test with a zero-test value equal to 0 to check if all the coefficients are really relevant (the non-relevant ones will be set equal to 0 after the zero-test);
- The GA cross-over and mutation probabilities can be set by the user;
- The PSO inertia and speed coefficients can be set by the user;
- The GD mutation probability and learning rate can be set by the user;

Flow

The high-level pseudo-code of the tool is:

- Configure the decision variables (OptiMix (MATLAB): *Main.m,* OptiMix (Website): Optimizer/Decision variables);
- Configure the objective functions (OptiMix (MATLAB): *Main.m* and *ObjFun_fun.m,* OptiMix (Website): Optimizer/Objective functions);
- Run the optimization function (OptiMix (MATLAB): *OptMix_v02_xxxxxx.m*, OptiMix (Website): Optimizer/Run):
    - If the iterations index is equal to 1, then initialize the decision variables' value of each particle of each family;
        - Otherwise, update the decision variables' value by using the GA, the PSO and the GD methods.
    - Calculate the objective functions' value of each particle of each family (OptiMix (MATLAB): *ObjFun_fun.m*);
    - Determine the particle-best, i.e. the optimal value of the decision variables and objective functions for each particle;

    - Determine the family-best, i.e. the optimal value of the decision variables and objective functions for each family;
    - Determine the global-best, i.e. the optimal value of the decision variables and objective functions;
    - If the termination criteria are not met, then repeat the previous steps;
        - Otherwise, interrupt the optimization and return the global-best decision variables and objective functions values;
- Show the results (OptiMix (MATLAB): dedicated script to be coded by the user, OptiMix (Website): Optimizer/Run).

OptiMix (MATLAB)

Examples

Below it is possible to observe few examples that show how the OptiMix (MATLAB) optimizer can be used. The same examples can be easily reproduced with the OptiMix (Website) optimizer.

Example 1: parameters estimation

Problem definition

This example shows how to use the OptiMix (MATLAB) optimizer to estimate the coefficients of a function to fit some experimental data.

The function considered in the example is:

$$y = f(x_1, x_2) = \frac{c_1 \cdot x_1 + c_2 \cdot x_1^2 + c_3 \cdot x_1^3 + c_4 \cdot x_2 + c_5 \cdot x_2^2 + c_6 \cdot x_2^3 + c_7 \cdot \exp(c_8 \cdot x_1 - x_2)}{10^9}$$

Where $c_1, \dots, c_8$ are the coefficients of the function, x_1 and x_2 the independent variables, and y the dependent variable.

The experimental data include $\hat{\boldsymbol{x}}_1$, $\hat{\boldsymbol{x}}_2$, and $\hat{\boldsymbol{y}}$. The number of experimental points is n.

The decision variables, $\boldsymbol{X}$, have been considered as the coefficients of the function above:

$$\boldsymbol{X} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]$$

The objective function, $ObjFun$, has been defined as the square error between the experimental and predicted data:

$$ObjFun = \min\left(\sum_{k=1}^{n} (f(\hat{x}_1^k, \hat{x}_2^k) - \hat{y}^k)^2\right)$$

Results

The first picture shows the family-best (colored lines) and the global-best (black line) values of the objective function for each family at each iteration. Note that OptiMix (MATLAB) optimizer automatically generate one figure for each objective function defined by the user.

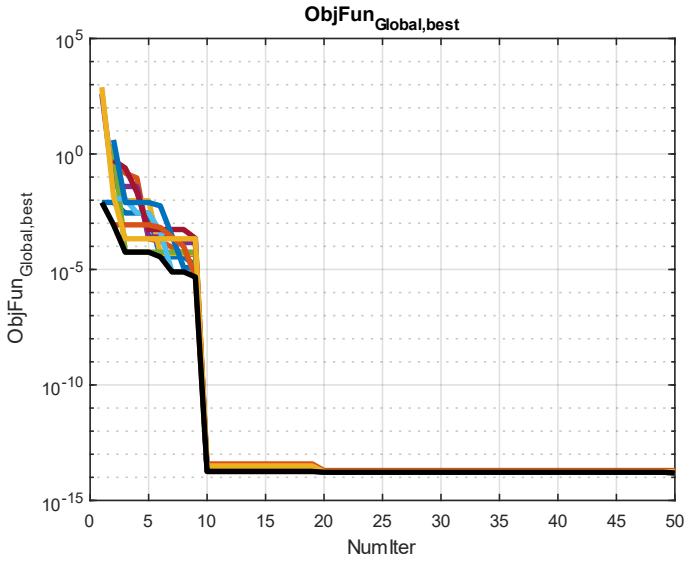

Figure 1 – MATLAB package – Example 1, global best objective function of each family.

The second picture shows the comparison between the experimental and predicted data, with the last being obtained by using the global-best values of the decision variables, $X_{Global,best}$.

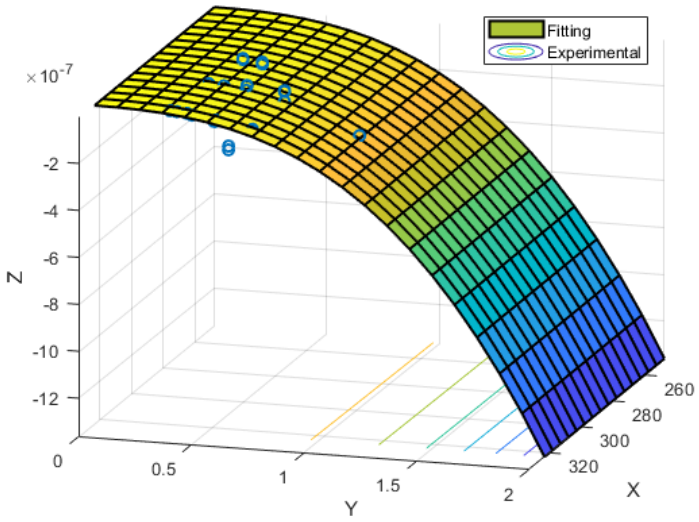

Figure 2 – MATLAB package – Example 1, comparison between experimental and predicted points.

Example 2: micro-grid daily cost minimization

Problem definition

This example shows how to use the OptiMix (MATLAB) optimizer to minimize the daily cost of a micro-grid constituted by:

- A Battery Energy Storage System (BESS);
- A PhotoVoltaic (PV) panel;

In particular, the goal is to define the BESS power (note: the BESS is the only component on which the micro-grid owner can act) during each hour of the day that:

- Minimizes the daily cost of the micro-grid;
- Guarantees that the BESS energy at the end of the day is equal to the value set by the user (the user doesn't want to get the BESS completely discharged at the end of the day);

The input data, collected in one between the *Data – 1.xlsx* and *Data – 2.xlsx* excel files, include:

- BESS capacity, $\hat{E}_{BESS}$;
- BESS minimum / maximum State of Charge (SoC), $\widehat{SoC}_{min}$ / $\widehat{SoC}_{max}$;
- BESS minimum / maximum power, $\hat{P}_{BESS,min}$ / $\hat{P}_{BESS,max}$;
- BESS minimum / maximum energy, $\hat{E}_{BESS,min}$ / $\hat{E}_{BESS,max}$;
- Load power, $\hat{P}_{Load}$;
- Photovoltaic panel power, $\hat{P}_{PV}$;
- Grid (i.e. electricity) specific cost, $\hat{C}_G$, for each hour of the day;
- Photovoltaic panel specific cost, $\hat{C}_{PV}$, for each hour of the day;
- BESS specific cost, $\hat{C}_{BESS}$, for each hour of the day;
- BESS initial energy, $\hat{E}_{BESS}(1)$;
- BESS desired energy at the end of the day, $\hat{E}_{BESS}(24)$;

The decision variables, $\boldsymbol{X}$, have been considered as the values of the BESS power during each hour of the day. Obviously, the BESS power is saturated to respect the PV panel power availability and all the other limits defined by the inputs listed above (for example, the power absorbed by the BESS cannot be higher than the power generated by the PV panel).

$$\boldsymbol{X} = [P_{BESS}(1), P_{BESS}(2), \dots, P_{BESS}(24)]$$

The first objective function, $ObjFun(1)$, has been defined as the daily cost of the micro-grid, which is given by the sum of the grid (i.e. electricity) daily cost, $Cost_G$, the PV panel daily cost, $Cost_{PV}$, and the BESS daily cost, $Cost_{BESS}$.

$$ObjFun(1) = \min(Cost_G + Cost_{PV} + Cost_{BESS})$$

The second objective function, $ObjFun(2)$, has been defined as the difference between the actual and desired BESS energies at the end of the day, $E_{BESS}(24)$ and $\hat{E}_{BESS}(24)$:

$$ObjFun(2) = \min\left(E_{BESS}(24) - \hat{E}_{BESS}(24)\right)$$

The optimization is launched three times:

- The first is based on the GA only;
- The second is based on the PSO algorithm only;
- The third is based on the GD method only.

Results

The first six figures show the family-best (colored lines) and the global-best (black line) values of the objective functions for each family at each iteration. Note that OptiMix (MATLAB) optimizer automatically generate one figure for each objective function defined by the user (since we are running the optimizer three times, there will be 6 figures).

Below the figures related to the PSO algorithm are shown:

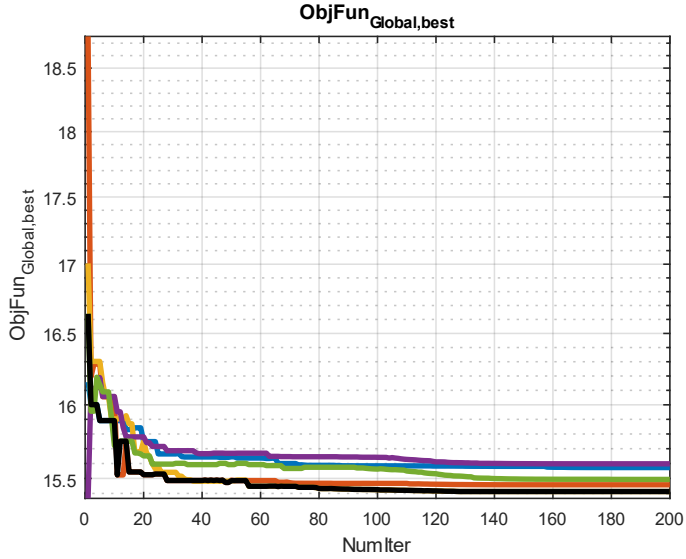

Figure 3 – MATLAB package – Example 2, global best objective function 1 of each family (PSO algorithm).

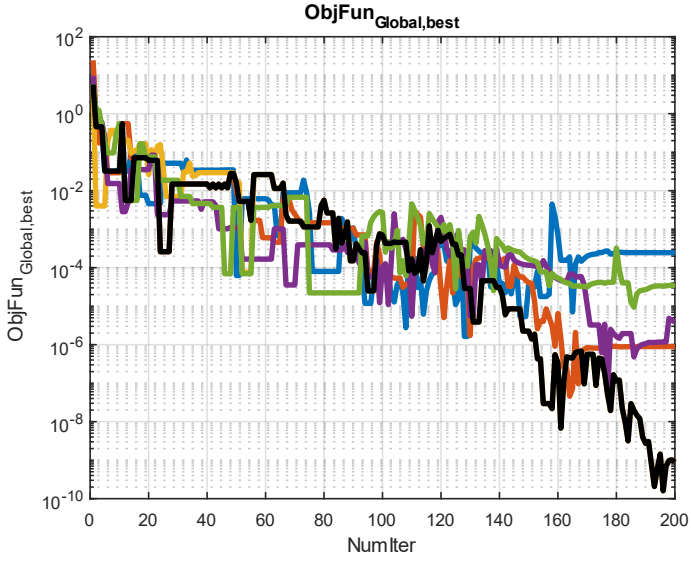

Figure 4 – MATLAB package – Example 2, global best objective function 2 of each family (PSO algorithm).

The next three figures show the Load, PV panel and BESS powers (one figure for each algorithm / method). Below the figure related to the PSO algorithm is shown:

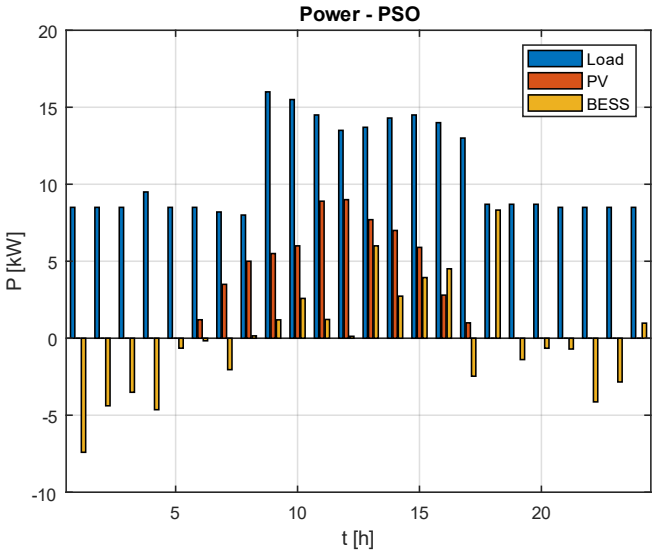

Figure 5 – MATLAB package – Example 2: load, PV panel and BESS powers (PSO algorithm).

The remaining figure show a comparison between the main quantities obtained using each algorithm individually. From the Figure 7 it is possible to observe how the BESS is discharged when the grid (i.e. electricity) specific cost is minimum.

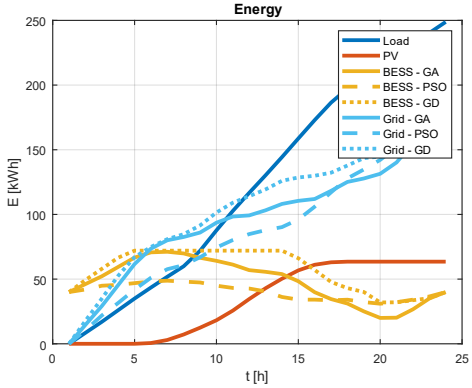

Figure 6 – MATLAB package – Example 2: energies comparison between GA, PSO and GD.

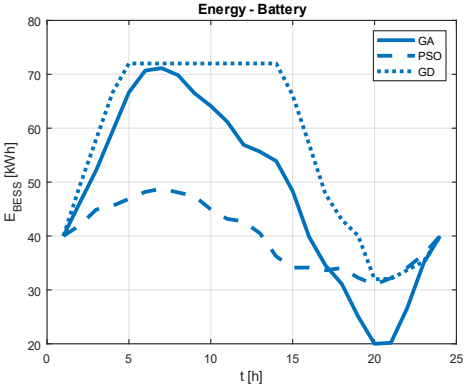

Figure 7 – MATLAB package – Example 2: BESS energy comparison between GA, PSO and GD.

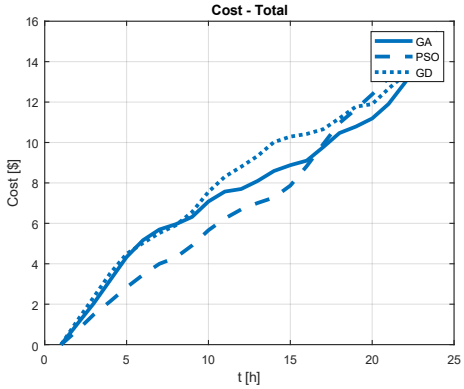

Figure 8 – MATLAB package – Example 2: total cost comparison between GA, PSO and GD.

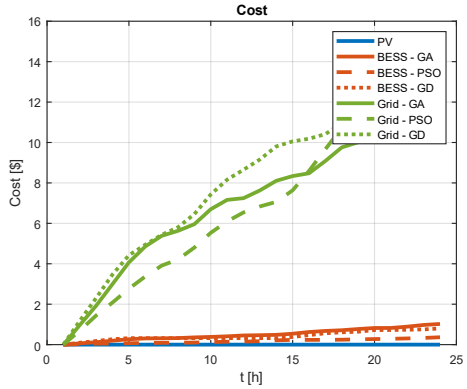

Figure 9 – MATLAB package – Example 2: individual costs comparison between GA, PSO and GD.

Files description

To run an optimization using the OptiMix (MATLAB) optimizer three files are necessary:

1. *Main.m*, in which all the external data / parameters / settings are set by the user:
    a. External data: all the external data can be passed to the following functions through a structure called $Data$. In this way it is possible to avoid calculating / importing / defining external data every time the objective function is calculated (for example, the *Data – 1.xlsx* excel file shown in the example 2 is imported only once, and its data is passed to the objective functions calculation function through the structure $Data$);
    b. Parameters / settings:
        i. The number of decision variables;
        ii. The number of objective functions;
        iii. The guess value of each decision variable;
        iv. The range of each decision variable;
        v. The type of the optimization of each objective function;
        vi. The priority of each objective function;
        vii. The number of families;
        viii. The number of particles;
        ix. The termination criteria;
        x. The restart frequency;
        xi. The zero-test frequency;
        xii. The GA cross-over and mutation probabilities;
        xiii. The PSO inertia and speed coefficients;
        xiv. The GD mutation probability and learning rate;
    c. Eventual post-processing and plotting;
2. *OptMix_vxx_FamIter.m* or *OptMix_vxx_IterFam.m*, i.e. the core of the OptiMix (MATLAB) optimizer. **Please, don't modify these files**. Note that *OptMix_vxx_FamIter.m* and *OptMix_vxx_IterFam.m* are equivalent, the results will not change if using one or the other.
3. *ObjFun_fun.m*, in which the objective functions vector, $\boldsymbol{ObjFun}$, is calculated based on the decision variables vector, $\boldsymbol{X}$, and the external data defined in the *Main.m* file, $Data$. This file must be modified case by case depending on the actual optimization problem.
4. Other files that can be added case by case (the mandatory ones to run an optimization are the ones defined above).

How to run

To run the package it is simply necessary to run the *Main.m* file. All the other two files will be called automatically.

How to get the full version

You can get the open version of the OptiMix (MATLAB) optimizer in the shop section of the following link: https://engimix.onrender.com.

OptiMix (Website)

An online version of the OptiMix optimizer is available at the following link: https://engimix.onrender.com.

The OptiMix (Website) optimizer can be used for free with limited number of particles and iterations. Tokens can be bought directly in the shop section of the link above and used to run optimizations with higher number of particles and iterations.

In addition, in the shop section of the link above it is also possible to buy the full version of the OptiMix (MATLAB) package.

How to run

Open the OptiMix (Website) optimizer

Open the link https://engimix.onrender.com.

Click on *Configure your optimization*.

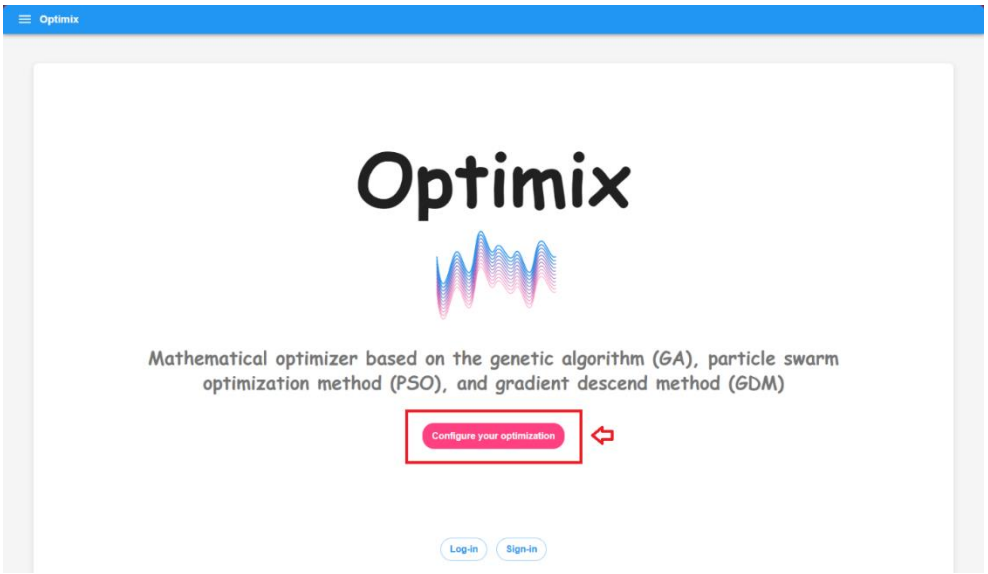

Figure 10 – Online optimizer – Main page.

Configure the decision variables

Select the number of decision variables by clicking on the *Plus* and *Delete* buttons.

Configure the parameters associated to each decision variable:

- Enable: enable / disable setting. It allows to enable / disable a decision variable. If a decision variable is disabled, it won't be considered during the optimization;
- Name: name of the decision variable. It is used to calculate the objective functions in the section below;
- X_{Guess}: guess value of the decision variable. It will be applied at the first iteration to the first particle;
- X_{min} and X_{max}: minimum and maximum value of the decision variable. The decision variable will be always limited within the $[X_{min}, X_{max}]$ range;
- V_{min} and V_{max}: minimum and maximum value of the decision variable speed. The decision variable speed will be always within the $[V_{min}, V_{max}]$ range;
- $c_{CO,GA}$: GA cross-over probability. Set it equal to 0 to deactivate the GA cross-over;
- $c_{M,GA}$: GA mutation probabilità. Set it equal to 0 to deactivate the GA mutation;
- w_{PSO}: PSO inertia. Set it equal to 0 to deactivate the PSO inertia;
- $c_{1,PSO}$: PSO coefficient applied to particle-best value to calculate the PSO speed;
- $c_{2,PSO}$: PSO coefficient applied to family-best value to calculate the PSO speed;
- $c_{CO,GD}$: GD cross-over probability. Set it equal to 0 to disable the GD;

- $\Delta X_{\%,GD}$: relative percentage increment. It is used in the calculation of the gradient for the GD method. Set it to low values;
- l_{GD}: learning rate. Set it equal to 0 to disable the GD;
- c_{ZT}: zero-test probability;
- X_{ZT}: zero-test decision variable value;

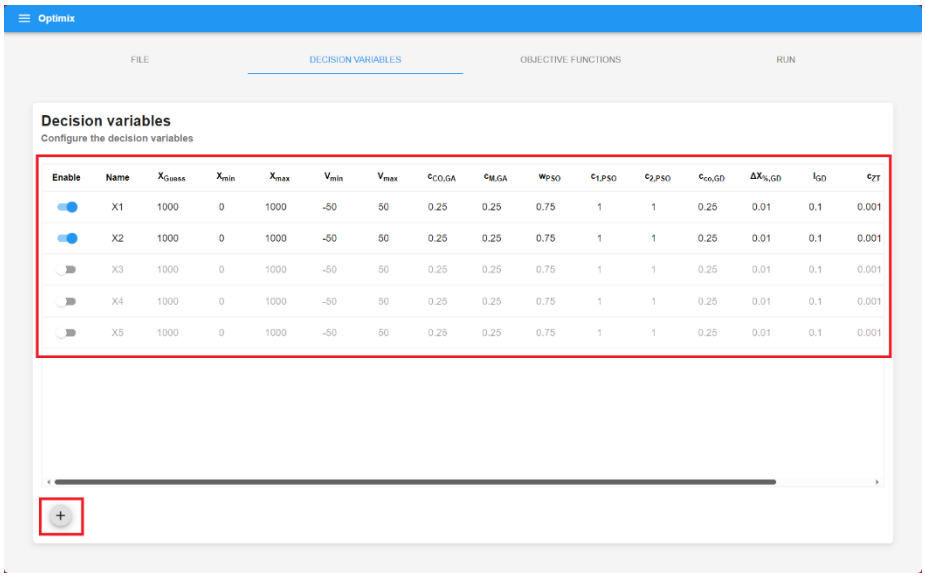

Figure 11 – Online optimizer – Decision variables configuration – 1.

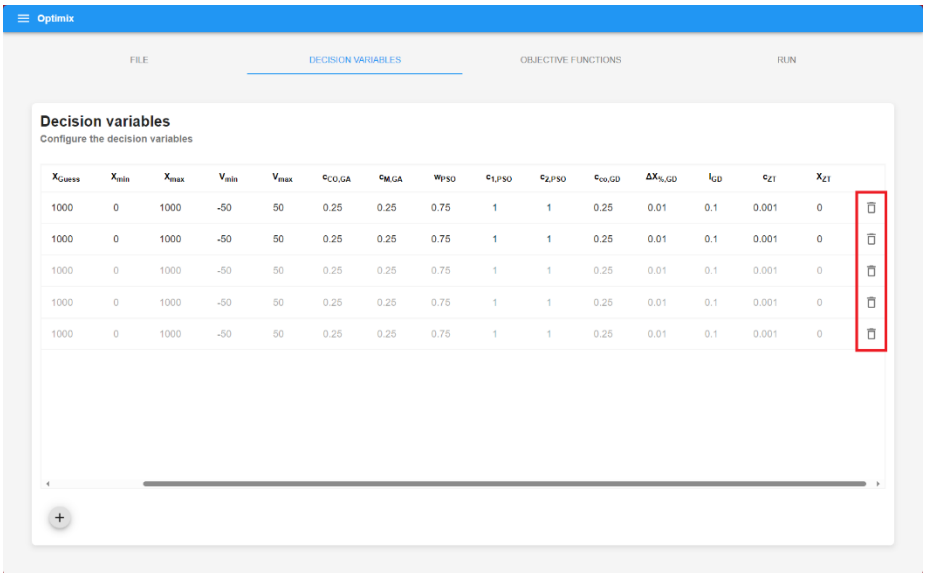

Figure 12 – Online optimizer – Decision variables configuration – 2.

Configure the objective functions

Select the number of objective functions by clicking on the *Plus* and *Delete* buttons.

Configure the parameters associated to each objective function:

- Enable: enable / disable setting. It allows to enable / disable an objective function. If an objective function is disabled, it won't be considered during the optimization;
- Name: name of the objective function. It is used to calculate the objective functions in the section below;
- Type: optimization type of the objective function;
- Weight: objective function weight. It allows to assign a higher (by increasing the weight) / lower (by reducing its weight) priority to each objective function;
- Reference: it allows to select the reference value used to normalize the objective function;

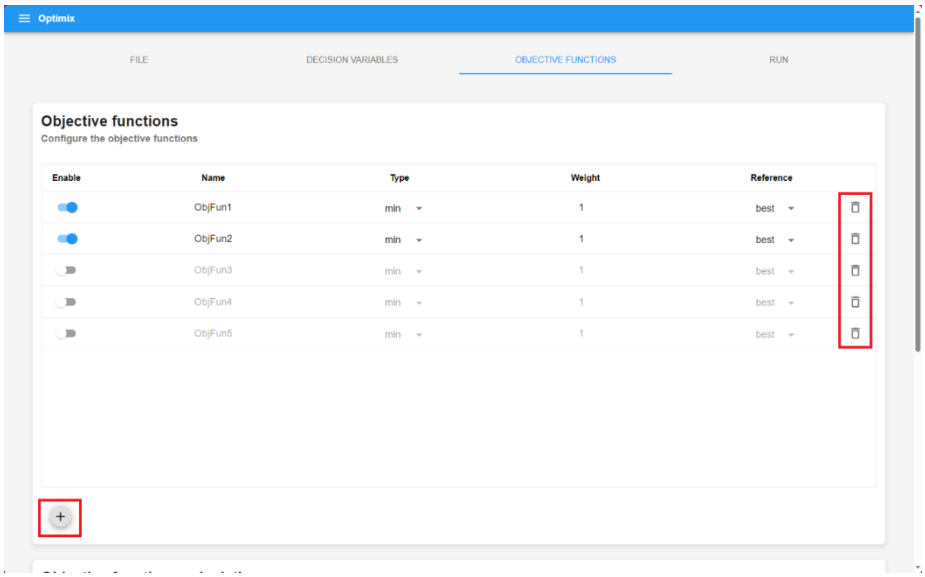

Figure 13 – Online optimizer – Objective functions configuration – 1.

Write the Javascript code that will be used to calculate each objective function based on the decision variables.

Note that there is no need to initialize the decision variables and the objective functions. However, in case additional variables are needed, these must be initialized using the let / var command.

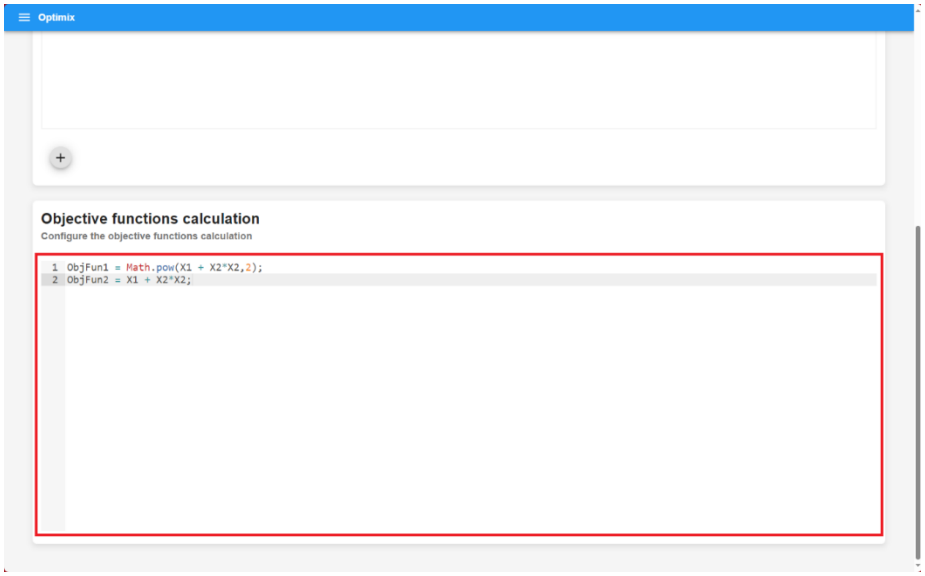

Figure 14 – Online optimizer – Objective functions configuration – 2.

Run the optimization

Select the number of particles and iterations.

Check that the number of tokens is equal to 0 (if higher than 0, then you need to Sign-in / Log-in).

Click on Run.

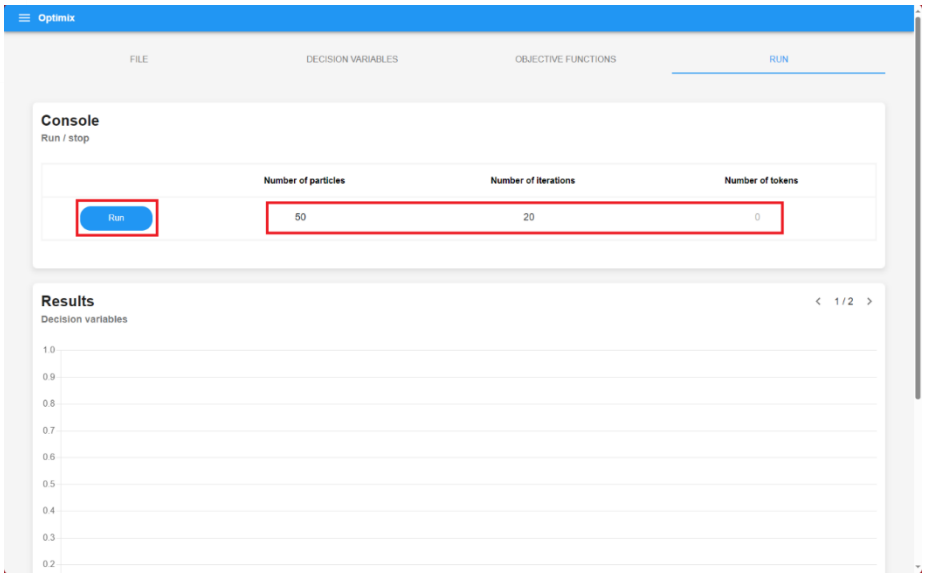

Figure 15 – Online optimizer – Run.

Analyze the results

Analyze the results through:

- The figures (one for the decision variables and one for the objective functions);
- The tables (one for the decision variables and one for the objective functions).

Figure 16 – Online optimizer – Results.

How to get tokens

In case the optimization requires more particles / iterations, than sign-in in order to get free tokens. Once these are concluded, you will need to buy new tokens in the Shop section.

How to save a file

In order to save a file you need to Sign-in / Log-in. Once logged, it will be possible to manage files for free.