
Polytechnique Montréal - École Centrale Nantes - INP Grenobles

TFE REPORT

---

# Optimization Benchmark Library

---

Constitution of an optimization problem library for Ibex and NOMAD

Directed by Pr. Jean Bigeon, Raphaël Chenouard, Sébastien Le Digabel and Ludovic Salomon

Maxime Gras

Option Industrial Engineering

02/02/2022

## Thanks

Before anything, I would like to send my most sincere thanks to everyone who helped me during this Internship.

In first place to Mr. Salomon who always took the time to answer my numerous questions and help me when I needed it.

My thanks also go to Pr. Le Digabel who offered me the precious opportunity to work in a place I already feel like home, Québec.

Of course, I want to thank Pr. Chenouard whose help was inestimable in the completion of the mission which was given to me.

And last but not least, my thanks go to Pr. Bigeon without whom nothing of this would have been possible.

All together you allowed me to realise a 10 years old dream and offered me the opportunity to have the life I always desired.

# Contents

# List of Figures

# List of Algorithms

# 1 Introduction

## 1.1 Summary of the Internship

The Internship can be split into two distinct parts regarding the place of work at that time. The Internship started on the 1st of September 2021. At that time I was in Nantes and hence focused on the work on `Ibex` and the coding of what we will call White Box. At my arrival in Canada, on the 4th of October 2021, the objectives of `NOMAD` were added. I started working on what we will call Black Box. The Internship will end on the 28th of February.

## 1.2 Entities of the Internship

In order to fully understand the different objectives of the Internship, I think it necessary to present each entity involved together with their expectations in the mission I was to accomplish.

### 1.2.1 INP Grenoble


Figure 1: Logo of INP Grenoble

The INP Grenoble 1 was my main employer during this Internship, it was them who edited the Internship convention. However the INP in itself had little to no implication in the decisions taken during the Internship and the objectives fixed as Pr. Bigeon (who was the link with INP Grenoble) was already in post in Centrale Nantes at that time. The two following entities we will discuss are places where I was sent in mission by INP Grenoble, known as Centrale Nantes and Polytechnique Montréal.

### 1.2.2 Centrale Nantes


Figure 2: Logo of Ecole Centrale de Nantes

Centrale Nantes 2 is one of the two main entities involved in the Internship. As said, I spent about one month there. It is important to know that Centrale Nantes is the institution which employs Pr. Chenouard and Pr. Bigeon but also the host of the OGRE (for Global Optimization and Set Resolution) team that is developing `Ibex`.

### 1.2.3 Polytechnique Montréal

Polytechnique Montréal 3 is the last entity involved in the Internship. It is the place of work of Pr. Le Digabel and Mr. Salomon and the host or the GERAD Laboratory (for Decision Analysis and Research Study Group) which is where the `NOMAD` software is developed. I spent most of my Internship there (from the 4th of October to the end of February).


Figure 3: Logo of Polytechnique Montréal

Due to the presence of two different entities, it is easy to understand that two differents objectives were at stake during this Internship. Centrale Nantes was expecting work on `Ibex` and Polytechnique Montréal was expecting work on `NOMAD`. We will clarify these expectations later. For now I think it important to explain what exactly are `Ibex` and `NOMAD`.

### 1.3 Context and Objectives

First of all, we will establish the general context of work. We are using differents algorithms to find the optimum of a function over a defined decision space, that is to find the values of the variables of a function which minimize it. This is what we call an optimization problem which can be formulated as :

$$\begin{gathered}\Omega \subset A \\ f : A \longrightarrow \mathbb{R} \\ c : A \longrightarrow \mathbb{R}^n \\ c' : A \longrightarrow \mathbb{R}^m \\ \min_x \{f(x) \ : \ x \in \Omega \ ; \ \forall i, j \in \mathbb{R}^n \times \mathbb{R}^m \ , \ c_i(x) = 0 \wedge c'_j(x) \leq 0\}\end{gathered}$$

In this definition, $x$ represents the variables defined over the studied space A. The $f$ function is the objective function and the $c$ and $c'$ functions are respectively the functions calculating the equality and inequality constraints.

The variables are defined over a pre-established space (by bounds) and can also be used to calculate auxiliary functions that can be used as constraints.

For example, one can want to find the proportions of ingredients that will maximize the $kcal$ given by a cake while keeping its mass under a certain value. In this case, each variable would be bounded by 0 and the amount available at the groceries store, the objective function would be the one giving the $kcal$ quantity and an auxiliary function used as a constraint would be the mass of the cake.

In the above example, the objective function is what we call a White Box because chemistry and mathematics can give us an analytical expression of the $kcal$ of the cake as a function of the amount of each ingredient. But if the objective function is the taste, we have no function which can calculate it and optimizing it would require to gather tasters, establish a protocol to note each cake, bake them, submit them to notation etc. The process is quite longer and more difficult to put into practice, this is what we call a Black Box.

We will now formalize those therms :

Before diving in a list of the boxes we will define the notion of Black Box or White Box :

- **White Box :** What we call a White Box is an optimization model where we can see inside the model, that is to say that we know the equations between the point given as an input and the output (the objective functions and the potential constraints). In practice this kind of model is the one that will be treated by `Ibex` because, as we said earlier, `Ibex` needs the access to the equations of the model to function.
- **Black Box :** In optimization, a Black Box is any process that when provided an input, returns an output, but the inner workings of the process are not analytically available [13]. The most common form of Black Box is computer simulation, but other forms exist, such as laboratory experiments for example. In practice this kind of model is the one that will be treated by `NOMAD` because we cannot have access to the functions used by the model. Moreover, it is frequent that the Black Box is a specific program (instead of equations like a White Box).

There already exists some libraries of models like COCO for Comparing Continuous Optimizers [6]. As stated on their website, the COCO framework is an open-source library aiming to provide a way to test optimizers. It is quite similar to our library in its objective. However, it provides a more complete framework with :

- An interface to several languages in which the benchmarked optimizer can be written, currently `C/C++`, `Java`, `Matlab/Octave`, `Python`.
- Several benchmark suites or testbeds, currently all written in `C`.

- Data logging facilities via the Observer.
- Data post-processing in `Python` and data browsing through html.
- Article LaTeX templates.

While being quite useful and more complete than our library as well as accepting Black Boxes, the COCO library presents one downside, it is not simple. Indeed one of the main objectives set for our work here is to be simple and fast to install, following the idea of "one-click" for online shopping. Even if the installation of COCO framework is not especially complex, it might discourage potential users who only want to test their solver on a few models and do not want to bother installing the quite complete but heavy COCO framework and setting it up.

That is where our work will take place. Indeed, we aim to provide a library of models easily and instantly available for the user without having to install a whole application (only commonly used packages). By limiting ourselves to `python`, `c++` and packages for `python` (described in the readme document of each Black Box), models are directly available and usable upon downloading to fit the most occasional use a programmer could have. Moreover, our multiple examples of Black Boxes and a `json` (which is a data structure standard [14]) framework will allow a user to easily implement a new model and run it with `NOMAD` or `Ibex` (according to the type of model and his needs).

We will now present the `Ibex` and `NOMAD` tools and what differences we can remark between them.

### 1.3.1 `Ibex`

`Ibex` [1] is a global optimization solver based on interval calculus using gradients methods. It relies on a Branch and Bounds algorithm [16] with successive iterations on intervals (research space) containing a constrained space (or part of it). At each iteration, `Ibex` will "shave" a part of the box to end up as close as possible to the constrained space.

We will now dive further into details on how it is done with the help of some figures and starting with `IbexSolve`, the algorithm of problem solving of `Ibex`.

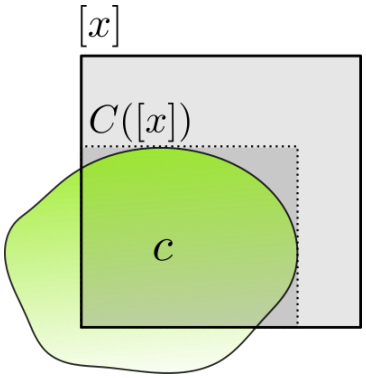



Figure 4: Representation of the beginning of the contraction step of the algorithm [2]

In the Figure 4 above can be seen the first step of `Ibex`. The green shape is the constrained space (that is to say the values of the variables for which the constrains are satisfied). The [x] box is the interval studied at the beginning of the algorithm while the C([x]) box is a contraction of the uncertain [x] box. We call an uncertain box one that is partly contained in the constrained space and we call a valid box one that is fully contained in the constrained space. This is a Branch-and-Bound algorithm that can be summarized with the Algorithm 1 (prune is a way to say discard the box because its lower bound is superior to the current solution box' upper bound).

Upon iterations, `Ibex` will split every uncertain boxes in smaller ones, keep the valid boxes, elimi-

**Algorithm 1** Branch-and-Bound($X, f$) [16]

**while** $L \neq \emptyset$ **do**
  Select a subproblem $S$ from $L$ to explore.
  **if** a solution $\hat{x}' \in \{x \in S \mid f(x) < f(\hat{x})\}$ can be found **then**
    Set $\hat{x} = \hat{x}'$
  **end if**
  **if** $S$ cannot be pruned **then**
    Partition $S$ into $S_1$,$S_2$,...,$S_r$
    Insert $S_1$,$S_2$,...,$S_r$ into $L$
  **end if**
  Remove $S$ from $L$
**end while**
Return $\hat{x}$

nate the invalid ones (the boxes completely outside the constrained space) and contract the uncertain box with different mathematical tools called contractors.

Eventually, when the boxes are smaller than a defined parameter, they will not be treated again even if they are uncertain (to ensure the convergence of the algorithm) and we might end up with a result that can be represented as this one :

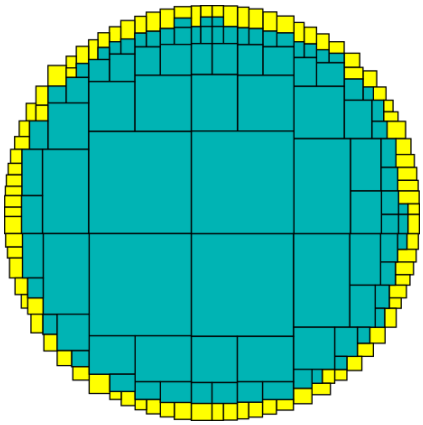


Figure 5: Representation of the studied space for `IbexSolve` at the end of the algorithm [1]

Here on Figure 5, valid boxes are in blue and the mixed boxes are yellow and small enough for the algorithm to not treat them again (the constrained space is here a circle). This kind of algorithm is called Branch and Prune.

Now we'll talk a bit about how `IbexOpt` works. While the Solve part uses a Branch and Prune algorithm, the Opt part relies on a Branch and Bound algorithm.

The Branch and Bound method is quite similar to the Branch and Prune, given a function defined over a 2-dimensional space and minimized under a constraint represented by the ovoid shape in the Figure 6:

We will then define an initial box covering all the represented space and contract it over the constrained space with the help of contractors like on Figure 7.

This step is followed by a sampling of a point inside the box (let's call it X) and add the constraint to be lower than f(X) for the following iteration, together with box splitting just like the Branch and Prune algorithm.

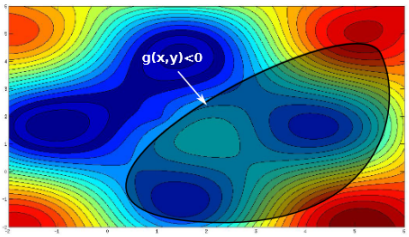



Figure 6: Representation of the starting space for IbexOpt [5]

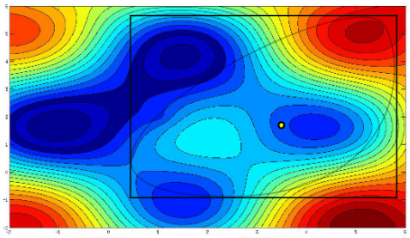


Figure 7: Studied space for IbexOpt after contraction and sampling [5]

So we have now explained how the Ibex algorithm works but we have one final point to mention before concluding on its limitations. We mentioned a bit earlier that the contraction of the boxes was done through the usage of contractors and even though we will not detail every contractor used nor the detail of the algorithm, the limitations of Ibex are clearly linked to their functioning so we will expand a bit on it.

Like we said, contractors are algorithms built with constraints relative to a set of variables. A contractor takes as input a box (that is to say a set of intervals for each variables). The contractors will return as output a box smaller or equal to the input box, ensuring that the output box respects the constraints. As a consequence, contractors need to have access to the functions involved in the delimitation of the constrained space.

Now that we explained how Ibex works, we can state some characteristics of this solver:

- First, Ibex might not provide an optimum point or a set of points respecting constraints if the algorithm had not enough time to complete. In the worse case, Ibex will return boxes containing a point (for IbexOpt) or a set (for IbexSolve) that satisfies the constraints.
- The solution provided are boxes containing a global optimum (for IbexOpt) or the full solution set (for IbexSolve) over the starting set. We miss no solutions of the problem.
- Because contractors need to have access to the constraints and/or the objective function, Ibex is not adapted to the optimization of Black Box.

### 1.3.2 NOMAD

NOMAD [17] is an implementation of MADS (for Mesh Adaptive Direct Search [9]) algorithm depicted in Figure 2. Basically its functioning can be vulgarized as a ball rolling down a pit. The algorithm starts with a point and will try to find a point where the objective function is lower at each iteration. We will now detail how the MADS algorithm works to once again get a good grip at what are the limitations of NOMAD and why.

First of all, NOMAD needs a starting point that is feasible, which means that satisfies the constraints of the model, we will detail why later. Than two other parameters will be taken into account: the frame size and the mesh size.

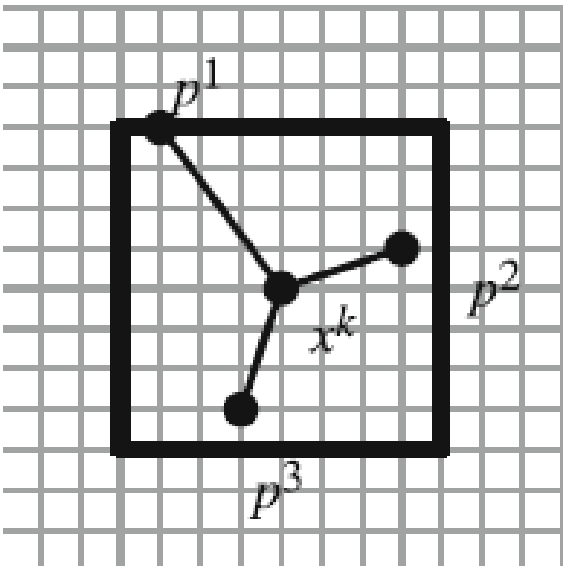



Figure 8: A poll step executed by NOMAD in a 2D decision space [13]

Basically, the point is to create a grid around the considered point like Figure 8. The grid is as wide as the frame size parameter, and each node is separated from its neighbours by the mesh size parameter. The NOMAD algorithm will then select some points on the mesh and try to find a better point than the first one, this is called the poll step. Once it is done, it will create a new mesh and frame around this point and continue the optimization.

How the next point is chosen depends whether NOMAD is set in Extreme Barrier or Progressive Barrier [18] mode, the first considering that the constraints cannot be violated and the second considering that it is allowed.

Before detailing the differences, we will define the h-value which is a measure of the feasibility of a point. The bigger it is, the more constraints the point violates. It is good to note that the h-value does not take into account the objective function value.

- **Extreme Barrier :** In this case, constraints are considered as non-relaxable. If an unfeasible point is given at the beginning of the algorithm, NOMAD will first try to find a feasible one by lowering the h-value. It will than proceed to the optimization and consider every point with $h \neq 0$ to have an objective function value of $+\infty$. This step is only executed if the first point given is feasible. In practice finding a feasible point by lowering the h-value is quite unlikely to happen, and the algorithm will waste all its allowed iteration trying. That i why we set as a strong constraint for the first point to be feasible.
- **Progressive Barrier :** In this case, constraints are relaxable and NOMAD will optimize a set of two points. One feasible (with $h = 0$) and one unfeasible, that is to say with $h \in ]0; h_{threshold}]$. The value of $h_{threshold}$ will decrease with the pursue of the algorithm.

For the Algorithm 2, we define as a failed operation one that did not manage to update the studied point. In this case the frame size is reduced (the mesh size is always inferior to the frame size). A successful operation is defined as one that managed to find a better point. In this case, the frame size and mesh size increase.
Now that we know approximately how NOMAD works, we can make some deductions on its limitations :

- As we could suppose it with the metaphor of the ball rolling down a hill, it is possible for NOMAD to be stuck in a local minimum. Unlike Ibex, we are not sure to find the global minimum of the

**Algorithm 2** High-level presentation of MADS [9]

Initialization : Let $x_0 \in \mathbb{R}^n$ be an initial point and set the iteration counter $k \leftarrow 0$.
**repeat**
  SEARCH on the mesh to find a better solution than $x_k$
  **if** the SEARCH failed **then**
    POLL on the mesh to find a better solution than $x_k$
  **end if**
  **if** a better solution than $x_k$ was found by either the SEARCH or the POLL **then**
    call it $x_{k+1}$ and coarsen the mesh
  **else**
    set $x_{k+1} = x_k$ and refine the mesh
    Update parameters and set $k \leftarrow k + 1$
  **end if**
**until** Stopping criteria is satisfied

function.

- A direct consequence of the point above is that the efficiency of the algorithm heavily relies on the starting point and changing it can alter the output of the algorithm.
- Due to the method used to find the optimum, that is to say make a grid around the considered point, the algorithm cannot manage equality constraints. It can be visualized as the difficulty (and statistic impossibility) of finding the exact point satisfying the equality constraints.
- Finally, because `NOMAD` only needs the value of the objective function and constraints at the examined point, `NOMAD` can optimize Black Box. It can also optimize White Box but will be less effective than methods using gradients.

To work, `NOMAD` uses a text file containing all the information required to conduct the optimization, at least the name of an executable (that is the Black Box), the ordered variables, their number and their bounds as well as a starting point. Many other parameters (listed here [7]) can be asserted in this file that we will later refer to as `param.txt`. Moreover, in theory, `NOMAD` can handle bi-objective models. However this functionality has not yet been implemented in the version of `NOMAD` we are using here (`NOMAD 4`) but was available in its previous version.

# 2 The Library

Now that we introduced the important elements of the mission, we will dive into the details of the results of the Internship, starting with the "Polytechnique Montréal" colored objective.

## 2.1 Objective

In this part we will talk about the constitution of the Optimisation Benchmark Library. The point of this part was to provide a set of models to test optimization programs. The mentionned boxes can be of two kind, the White Box, more interesting for Centrale and the Black Box more interesting for Polytechnique Montréal. All the models can be found at https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/tree/main All the information required to install the library, add a model to it or how to run it is provided in the gitlab, respectively in the Installation Guide, the Model creation guide and the README.

We will now give a list of the models that have been coded during the Internship beginning with the White Boxes. The models have explanatory sheets under their Docs directory https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary

## 2.2 White Boxes

The White Box we present here were coded mainly for `Ibex` but also have a `NOMAD` version and an explanatory sheet. The explanatory sheet details the variables, the constants, the expression of the objective function and the constraints as well as at least two verification points.

The `Minibex` files we give should be executed in a shell with the command `ibexopt Model.mbx`. Moreover we advice the use of the `-trace` command to have a monitoring of the optimization.

### 2.2.1 Coil

The objective of this model is to find the sizing parameters, as shown Figure 9, of a coil that minimize its mass while keeping the coil's inductance at $10^{-3}$ $H$ and the Joule Power in $[70; 90]$ $W$. The model is composed of five bounded variables, one equality constraint and two bounded outputs among which the objective function.

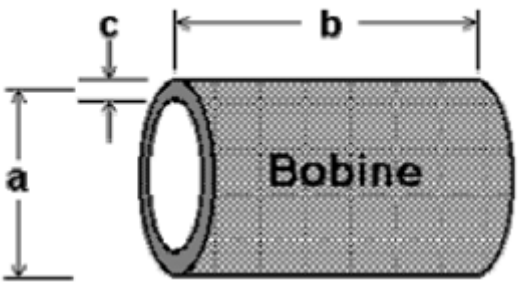



Figure 9: Scheme of a coil showing the main parameters [3]

`Ibex` manages to obtain a result at the relative precision of $10^{-4}$ in less than one second as this result is the lower bound given by the model 5, $20$ $kg$.

### 2.2.2 MAPSE

The objective of this model 5 is to find the form of a motor with permanents magnets (MAPSE stands for Machine à aimant permanents sans encoches in french) that minimizes a custom objective function

while keeping its electromagnetic torque at $10\ N.m$. The main variables are shown Figure 10. The objective function is composed of the Joule power, the volume of magnets and the volume of useful parts each divided by their minimum reachable in the studied space. Those said minimums were obtained via `Ibex` and respectively equal $12.235\ W$, $0.2750 * 10^{-4}\ m^3$ and $2.4991 * 10^{-4}\ m^{-3}$. The model contains eleven bounded variables with one being an integer (the number of pairs of magnets) and one equality constraint.

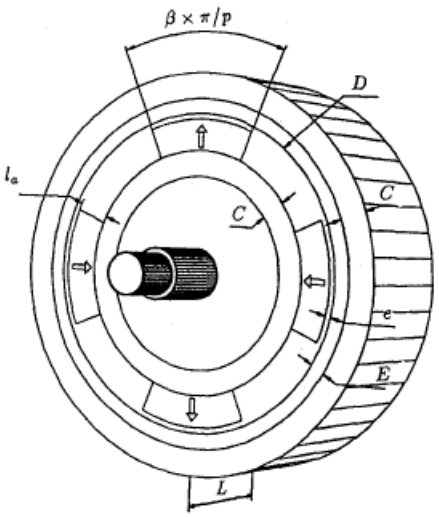



Figure 10: Scheme of the machine with the main parameters [15]

The final optimization gives a value of the objective function of $4.348$ compared to a minimum value of $3$ in several hours with `Ibex`.

### 2.2.3 I-Beam

The I-Beam model 5 aims to find the shape of a steel I-beam that will minimize its weight while keeping the maximal constraint under $\sigma_{max} = 12.87 * 10^6\ kg.m^{-1}.s^{-2}$. The model is quite simple and is composed of four variables, the dimensions shown on Figure 11, one bounded output and an objective function.

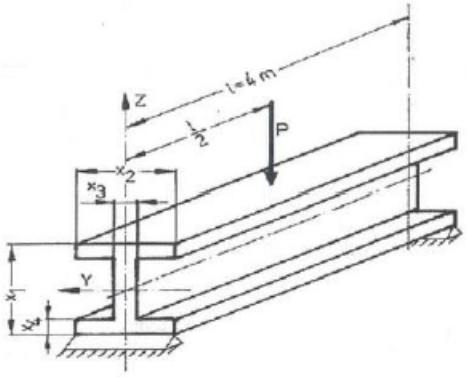



Figure 11: Scheme of the beam with the main parameters [4]

The optimization of the model gives the result of $M = 185.1\ kg$ in three seconds.

### 2.2.4 Transformer 1

The first Transformer model aims to design a security transformer under several hypothesis (detailed in the corresponding document) with the lowest mass possible while keeping the copper temperature $T_{copper}$, the iron temperature $T_{iron}$, the efficiency $\eta$, the tension ratio $\frac{\Delta V_2}{V_2}$, the intensity ratio $\frac{I_{10}}{I_2}$ and the filling factors of the coils $f_1$ and $f_2$ under some values.

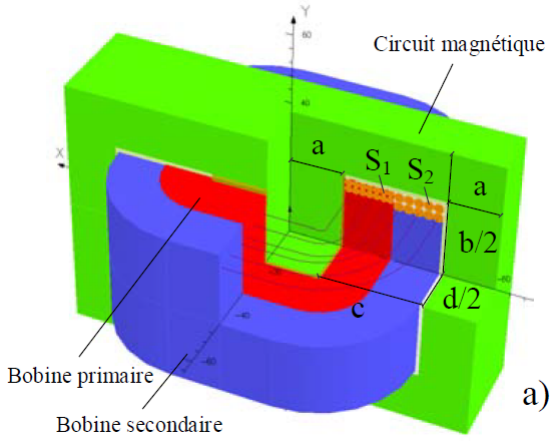



Figure 12: scheme of the transformer with some of the variables displayed [23]

The model [23] is quite heavy, having 9 variables (some of them are represented on Figure 12) with one of them being integers and needing to calculate thirteen additional variables to conclude on the validity of the point. The optimum point gives $M_{tot} = 2.844\ kg$ after several hours of optimization.

It is interesting to note that the model was at the origin described in [8] as discrete but no catalogue was provided for the variables. Creating a fitting catalogue would probably reduce the computation time and give a more realistic depiction of the reality.

### 2.2.5 Transformer 2

The second Transformer model [21] aims to design the same kind of Transformer than the previous model but optimizes the production and functioning cost, having as variables the height (represented by h on Figure 13)and the number of spires in the coil.

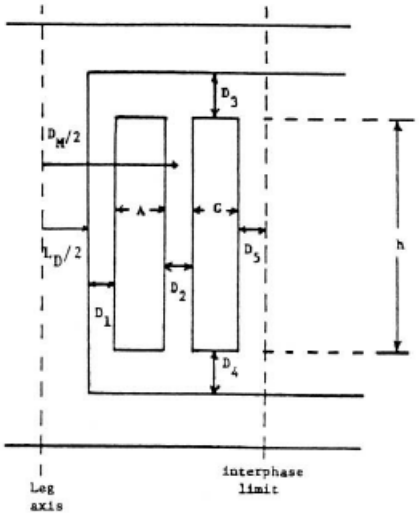



Figure 13: scheme of the transformer with the geometry parameters represented [**Transfo2**]

Like in the previous model, the optimization process is quite long because a lot of intermediate variables have to be calculated. The optimisation gives a final price of $2085\ k\$$ for a $h = 72\ cm$ high transformer.

### 2.3 Black Boxes

The models we will talk about now were coded for `NOMAD` optimization. As they are all modeled as Black Boxes coded in different languages, two `python` executable are provided, one to test the different settings of the Black Box on a chosen point (one iteration) and one to launch the optimization with the desired settings starting on a chosen point.

This design was chosen to answer the need of simplicity for the future users. In the same idea, a README document in each file gives the packages used by the model and the optimization and execution scripts are respectively the ones ending in `opt.py` and `sim.py`.

The `README.md` file explains how to use the library and the `sim.py` files are as guiding as possible for the user, help is provided upon entering an unrecognized command and possible starting points are also provided and should be executed in a `python` environment with the `python sim.py`. The file are designed to guide the user to test the point he wants with the desired parameters. The `opt.py` files are less guided and only ask what value the user wants to give to the settings. They suppose that the user already ran the `sim.py` at least once.

#### 2.3.1 COVID

The model COVID is issued from the article [22] and the associated code. The objective of this model is to determine the true parameters of the spreading of the COVID-19 pandemic in France as they were supposed to be highly underestimated.

The determination of those parameters is done by giving those parameters as variables and creating an objective function that compares the consequent spread to the actual one (with data collected per region in France) and try to minimize it (this is a parameter tuning method).

The model has two different problems depending on whether we use the data of France before or after the lockdown, with four variables in the pre-lockdown instance and only one in the post-lockdown.

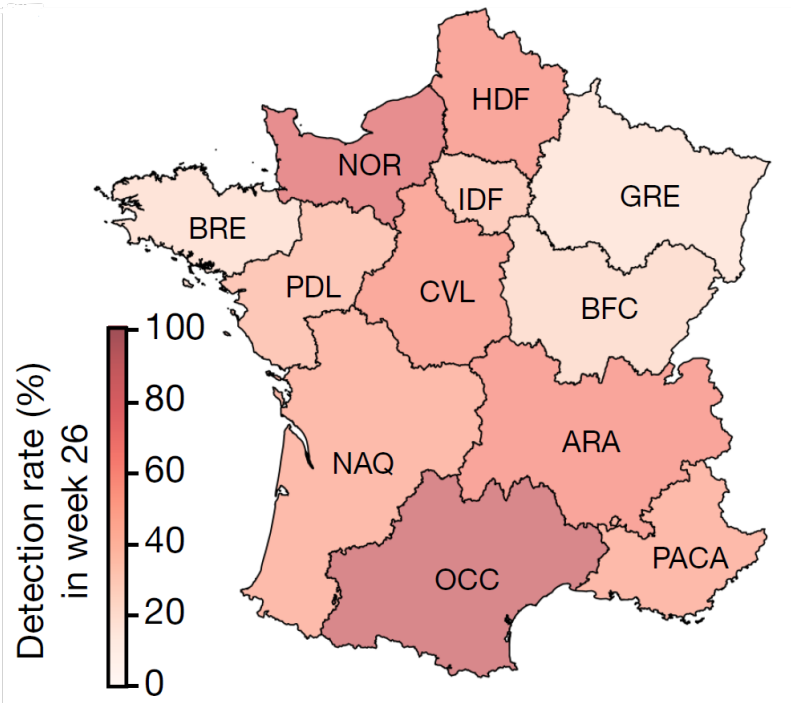


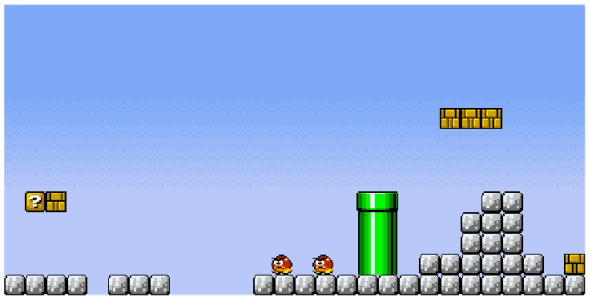

Figure 14: Map of the region considered by the model together with the estimated detection rate [22]

To launch the evaluation of a point, a region of study is required (as the data are different) and each region can be considered as another instance of the model making twelve of them. The region

considered are shown on Figure 14. Moreover, because the model is using random functions, a settable seed is available in the model. Finally, once again because of the random component, a number of stochastic run is demanded and the result is the median of those runs.

### 2.3.2 Mario

The objective of this model is to generate Mario levels, as in Figure 15, though the optimization of objective functions and the original code is taken from [25]. The number of variables is at the discretion of the user as it can be ten, twenty, thirty or forty.



(a) Playable level maximizing jumps

Figure 15: Example of level generated by mario-gan [25]

The variables are used by the Black Box to create a Mario level which is then tested by one of the twenty-eight objective functions given by the model. Those functions aim to establish if the level is a suitable one or not, checking for example the time taken to complete the level. The output of this model can also be set to be bi-objective, recombining the twenty-eight objective functions in ten pairs.

The model also gives the choice between seven instances of the problem. However it is important to note that the functions eleven to twenty-eight are stochastic and that the instance does not stand for a seed as fixing it won't prevent two runs with the same parameters to output different results. Moreover no seed implementation was examined by the authors of the code.

### 2.3.3 TopTrumps

Top Trumps model is issued from the paper [24] and has a objective similar to Mario but focuses on creating a balanced top trumps card game by setting the value of each card like the one depicted by Figure 16 (the only difference is that the card shown has five values instead of four).

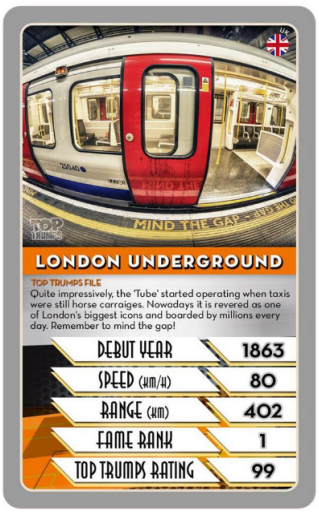



Figure 16: Example of a TopTrumps card with 5 values (the designed game has 4) [24]

Like in the Mario model, five functions are available as objective functions and are used to balance the game. For example, the trick difference at the end of the game is to be minimized.

Those five functions are also paired in three options when the model is set to give bi-objectives outputs. Finally, the model provides fifteen instances of the problem to work with.

### 2.3.4 Rover

The rover model is issued from the article [27] just like the next model we will examine. The objective is to create a suitable trajectory for a rover in a cluttered area like 17.

The variables are a set of sixty floats with values in $[0; 1]$ and are used to create the trajectory between a pre-selected starting and ending point on a pre-created map.

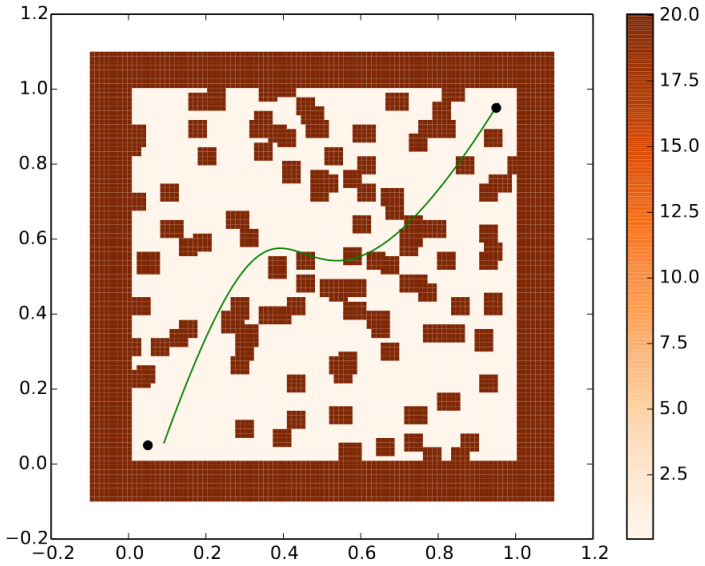



Figure 17: Example of a cluttered map and the trajectory determined via Ensemble Bayesian Optimization in [27]

The optimized function is a reward function penalizing any collision and aiming to have the shortest trajectory possible. Like we can see in the example, this does not mean that no collision will occur but that the rover will avoid them if there is a short way around.

The model offers no alternative instances nor is using any seed that should be fixed.

### 2.3.5 Push

As said previously, the Push model is taken from the same work as the Rover model, the article [27], and intends to simulate two objects being pushed together by two robot hands.

The situation is represented by 18, the blue points are the starting positions of the objects, the red ones are the ending points and the green are the goals

The variables are fourteen parameters specifying the location, rotation, pushing speed direction and time of each hand. The objective function is, as shown in the Figure 18, the lowest when the ending points are close to the goal.

### 2.3.6 Neural

The final Black Box model we provide is slightly different than the previous ones. Indeed, the objective here is to tune the hyper-parameters of a neural network for the MNIST database 19. We based this algorithm on the work of Mr. Edward H-Hannan [12].

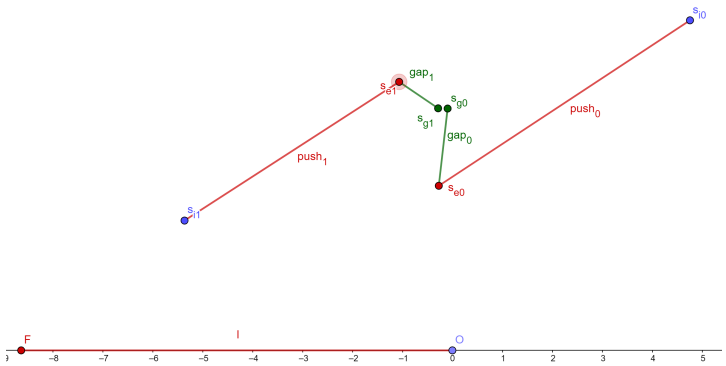



Figure 18: Scheme of the situation done under Geogebra

The example here is easily transportable to another architecture of picture recognition neural network.

Figure 19: Sample of the MNIST data [26]

The model we provide now creates an on-measure neural network by choosing some parameters that will not be taken into account in the optimization (the hyper-parameters, like the number of layers) and will then maximize the percentage of successful sorting after training by changing the variables (like the hidden layer input and output size).

Currently, the model offers the possibility to create convolutional neural networks (CNN) or fully-connected neural networks (FCC).

## 2.4 Conclusion

We will now have at the output of a `Ibex` and a `NOMAD` model, respectively Poutre and Push :

- `Ibex` output :

```
uplo= 184.730837787
uplo= 184.789743404

optimization successful!

f* in  [184.990405736,185.175396142]
       (best bound)

x* in  (<0.1, 0.1000000000000001> ; [0.2567554425341794, 0.2567554425341798] ; <0.01, 0.01000000000000001> ; <0.01, 0.0
1000000000000001> ; [0.005935108850683582, 0.0059351088506836] ; [185.1753961413277, 185.1753961413284] ; [3.90850226121
1414e-05, 3.908502261211416e-05] ; <12792624.04328322, 12792624.04328323> ; [0.001705683205771096, 0.001705683205771098]
)
       (best feasible point)

relative precision on f*:      0.001 [passed]
absolute precision on f*:      0.184990405736
cpu time used:                 2.424904000015
number of cells:               1874

results written in Poutre.cov
(old file saved in Poutre.cov~)
```

Figure 20: Output of the command `ibexopt Poutre.mbx -trace -rigor -t600`

We will now analyse this output. First one could notice the two yellow lines at the top of figure 20. Uplo stands for upper lower bound and is paired with loup, for lower upper bound (not shown in the figure). They both frame the value of the objective function and are supposed to tend toward the minimum of the objective function. Under the `optimization successful!` line, the two blocks represent the box in which the objective function and the variable vector are framed. For the variable vector, the variables are in the order of the .mbx file and a box with "<>" means that a point,withe the precision of `Ibex` is given. Finally, the last bloc gives useful information that, in the future, ought to be used to compare the efficiency of Ibex on this problem to the efficiency of other solvers. The last line gives the name of a .cov file (for covering) that contains the paving of boxes over the search space.

- `NOMAD` output :

```
102  -0.138772
103  -0.132655
104  -9.74937
105  -5.740115
A termination criterion is reached: Maximum number of blackbox evaluations (Eval Global) Success found and opportunistic
 strategy is used 105

Best feasible solution:     #4404 ( 0 -1 2 -10 18 3.1415 0 -5 -10 1 15 3.1415 0 0 )     Evaluation OK    f = -15.3310244
83598415742      h =   0

Best infeasible solution:   Undefined.

Blackbox evaluations:    105
Total model evaluations: 4065
(Nomad) max@LAPTOP-K7IMKV56:/mnt/c/Users/maxim/Projets/GitLab/TFE/optimizationbenchmarklibrary/Modèles/Push/Nomad$
```

Figure 21: Output of the command `python puopt.py` with default parameters.

We will now analyze the output of a `NOMAD` optimization with Figure 21. First we can see the upper bloc shows the number of the Black Box evaluation and the value of the objective function. The line under it statutes about the reason the algorithm stopped. In the current case, the budget of evaluation was trespassed, however, as we can see, there were 105 Black Box evaluation and not 100, this is due to the fact that the opportunistic method is used and, after the $99^{th}$ evaluation, it took 6 evaluations to find a better point. The bloc under that gives the best point found, the objective function value, the h-value and the number of the generated point (not evaluated point, that is why it can go so high). This is good to note that the setting DISPLAY_DEGREE corresponding to this output is 2 and can go up to 3 giving a lot more information about the steps of optimization. Once again those outputs might be used to compare algorithms over a problem in later versions of the library.

We have presented the eleven models that compose the library we propose and we can now conclude about the variety of the optimization problems we propose. The number of variables goes from only one (cosim post lockdown) to 208 (last instance of topsim), some have integers variables and some only floats (respectively the MAPSE engine for example and rosim) and some are non stochastic, stochastic with a settable seed or stochastic without a settable seed (like pusim, cosim and marsim respectively).

We have a large panel of examples available to test future optimization tools plus a full guide in the GitLab https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/tree/main for the one who want to add a model. But what is the particularity to the one we provide why should we choose it instead of the coco library for example ? The main advantage of our library is that it is coded in `python` and only needs a few `python` packages to work depending on the model intended to be used. Moreover, the models provided can acquire their points through text files (an easy way to give one) and print the output to be easily recoverable. In order to use the COCO library, the user has to dig through and understand the concepts of suite (a collection of one to five thousand problems) and observers (the interpret of the quality of the optimizer using the results over the suite). Besides, one can easily understand that a suite containing thousands of problems cannot have too complex models like the

ones we provide. To summarize, the COCO framework is the perfect tool to test an optimizer over a huge amount of models but less adapted to test an optimizer on a complex Black Box problem. Hence the work provided here.

Despite not having the main purpose as COCO, it would be interesting to take inspiration of the post working part of it, as we mentioned earlier,, that is to say run optimization several times, with different optimizers of with one optimizer on different models. However this would mean having the same information after an `Ibex` and a `NOMAD` run. Indeed, for example, `Ibex` cannot count easily the number of evaluation of the model while this is the main stopping criterion of `NOMAD`.

One way to solve this problem, at least for White Box, would be to be able to evaluate `Minibex` models with `NOMAD`. It might thus be interesting to create a way to use and optimize `Minibex` models in `python` if we do not want to code them from `Minibex` to `python` (something we can always do because they are White Boxes). But we could even go further and provide a way to use `Ibex`'s equations solving system as a Black Box for `NOMAD` and combine the mesh adaptive search and the interval calculus.

That is the objective of the next part, the Bridge.

# 3 The Bridge

As we said previously, the objective of what we call the bridge is to allow `Ibex` and `NOMAD` to cooperate for optimization. In order to accomplish that, we provide 3 files, `Lecteur.py`, `MiNomad.py` and `passerelle.cpp`. We will detail later the use of each file.

## 3.1 Objective

Before diving into details, let explain what exactly we intend to do with this bridge.

- We aim to provide a generalization of a model and aim to make it compatible with the provided tools. This generalization will allow anyone with a White Box model to easily add it to the library without having to understand the syntax of `Minibex`.
- As `NOMAD` considers everything as a Black Box, this bridge will allow to use any `Minibex` model as a `NOMAD` Black Box, giving the possibility to compare the two algorithm over the exact same problem.
- A consequence of this is that a new equation system resolution will be available for `NOMAD` in the case of a White Box (`NOMAD` will treat everything as a Black Box but by that we mean a model with explicit equations coded for `NOMAD`).

We will now expose how those objectives where answered and conclude on the success or not of those objectives.

## 3.2 Generalisation of a model

First of all, as it is at the base of the rest of the work provided, we will detail what a model is and what kind of structure we propose to generalise it and make it usable in our programs.

### 3.2.1 What is a model ?

We will first distinguish the notion of optimization model and simulation model.

- We call a simulation model a set of physical measures linked by equations aiming to transcribe the real behaviour of a machine or a physical phenomenon.
- What we call a model or optimization model is a by-product of a simulation model. We gave a value to some physical measures (typically choosing a material fixes density or conductivity) and chose others to be variable. We will try to find the values of the variables minimizing a quantity while respecting some constraints. The said quantity and the constraints can be taken from the original simulation model or completely ad-hoc.

We can thus construct the class diagram (see Figure 22) to preview what a generalised optimization model should be composed of :

- The model must have at least one variables with a name, bounds and if possible an initial guess as which value should be given to the variable in first approximation.
- An objective function $f_{obj}$.
- The expression of constraints. We will take as a convention that all constraints will be expressed as a quantity (one for each constraint) $Q = 0$ or $Q \leq 0$
- A box that can be Black or White that calculates the objective function $f_{obj}$ and the constraints quantities $Q$

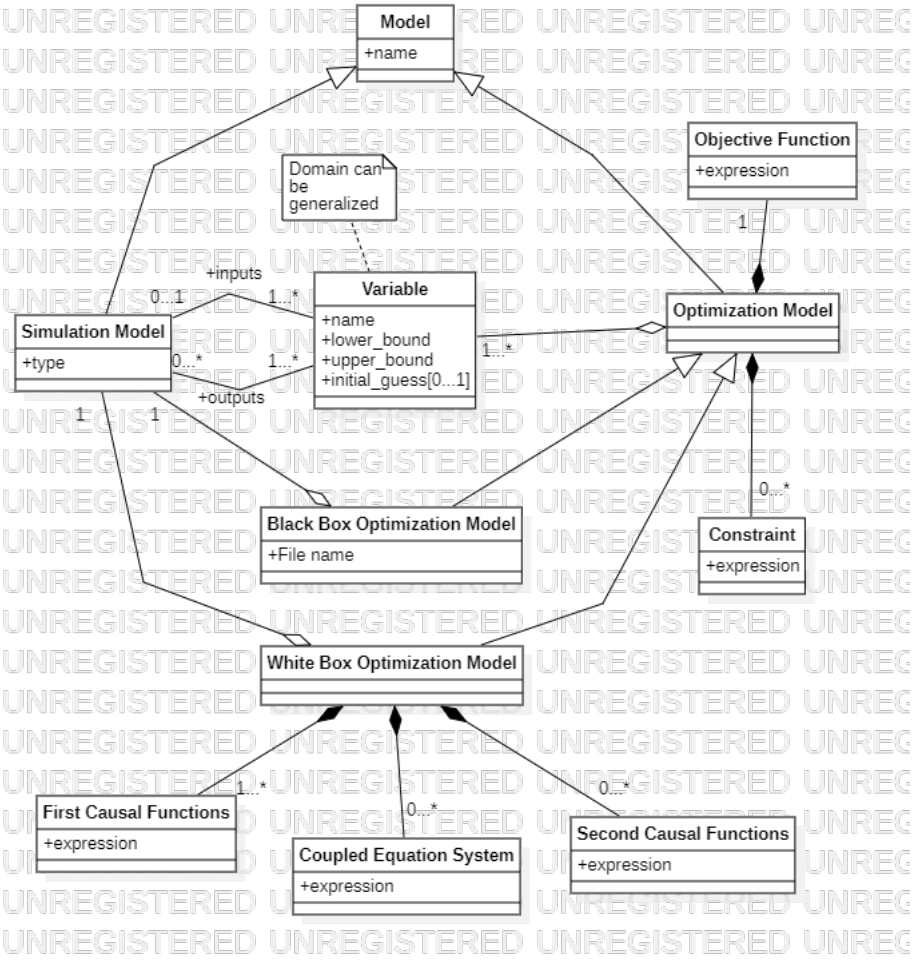



Figure 22: Class diagram of a model, realized under starUML

Now that we have established our structure, we will present it under the form of a `Json` file and compare it to what can resemble it.

### 3.2.2 The Json Structure

The `Json`, for JavaScript Object Notation, is a format of data designed to be easily readable and writable. This format was chosen (instead of `XML` or `YAML` for example) for their use is similar and it was prefered by Ludovic who was my main help during the Internship. First of all, let's show an example of the Json structure we provide :

```
{
   "name": "Model",
   "variables": [
      {
         "name": "Var1",
         "integer": false,
         "lower_bound": "0.1",
         "upper_bound": "1",
         "initial_guess": "0.4"
      },
      {
         "name": "Var2",
```

```
        "integer": true,
        "lower_bound": "1",
        "upper_bound": "10",
        "initial_guess": "7"
      }
   ],
   "constants": [
      {
        "name": "Const1",
        "value": "0.3"
      },
      {
        "name": "Const2",
        "value": "1e2"
      }
   ],
   "constraints": [
      {
        "expression": "Const1-Var1",
        "type": "<=",
        "relaxable": false
      },
      {
        "expression": "Var2^Var1-Const2",
        "type": "=",
        "relaxable": true
      }
   ],
   "objective_function": "Var1*Const1+Var2*Const2"
}
```

We can see in the code above that, in practice, the organisation of the Json file is slightly different than what we defined earlier to be an optimization model. For instance, in the `Json` file we define some constants that did not appeared in 22. Moreover we define no equations in the Json model. This is due to the particularity of `Ibex` which does not recognize the format of equations, however, well chosen constraints and constants can stand for equations, hence the difference of architecture in this particular case. As a consequence of the structure of the constraints and the lack of equations properly said, every quantity used in a model should be declared as a variable and one cannot use intermediate quantity.

However it is good to note that this architecture prevents from using some functionalities of `Ibex` and does not allow a model using them to be optimized. Indeed, in this language, intermediate variables do not have to be explicitly declared but need to be implicitly declared as a $x_{temp} = f(x)$ constraint, which is incompatible with the convention we established earlier to only write $Q = 0$ or $Q \leq 0$. Moreover, intermediate functions like the one usable in `Ibex` are currently not supported as the use of it is not standard and would require to develop a specific parser, a work which was not the objective.

Now that we exposed the `Json` structure we propose for the generalisation of models, we will compare it with what it currently available, the Functional Mock-up Interface (FMI and FMU for Functional Mock-up Unit).

First of all, as established in [10], "The Functional Mock-up Interface (FMI) is a free standard that

defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file." We can already note a difference as FMI is using a XML format while we use Json. This choice was only motivated by the syntax of Json that was clearer than XML in our opinion.

But the main difference is that, while we provide only four classes of objects (constants, variables, constraints and objective function), FMI defines a way more complete description 23:


Figure 23: fmimodelDescription element taken from [11]

This definition is way more complex than our library but the purpose is not the same. Indeed, FMI intends to provide complete physical models for engineering purpose, not to test optimization model, and are thus heavy, complex to apprehend and to code. That is why FMI is not adapted to our needs here for we aim, as said before, the simplicity of use and installation.

One could also want to compare our framework to the Optimization Services Project [19] which is, as stated in the user manual [20] "to provide a general framework consisting of a set of standards for representing optimization instances, results, solver options, and communication between clients and solver". Once again, the objective here is a bit different than what we are searching for. Indeed, this project provides a framework for models designed to be solved online by the OSP solver while our objective is to provide a way to test solvers. Thus we might as well provide our own framework more fitting for our needs.

To summarize our proposition of model architecture, it provides way less options, does not allow the complexity of FMI but is more than enough to code a benchmark of White Box models and manages to be simple enough to allow anyone to understand it easily and to add his own model.

### 3.3 Developed tools

Let us recall that the point here is to allow NOMAD to use Minibex or Json model in its optimization algorithms.

Having now detailed our architecture of model, we can present the tools we developed to answer the said objective.

To schematize, the three files we provide fill the role of a wrapper, linker and a core, respectively `MiNomad.py`, `Lecteur.py` and `Passerelle.exe` that cooperate according to Figure 24.

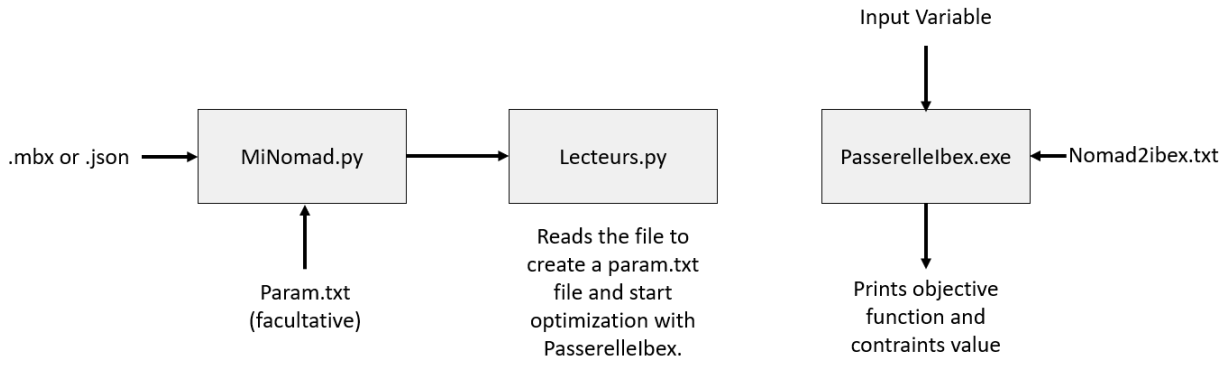



Figure 24: Representation of the interactions between the files of the Link

### 3.3.1 MiNomad.py

`MiNomad.py` plays the role of the wrapper. It is based on the same template as the Black Boxes. Its aim is to help the user doing what he wants with the core it is wrapped around.

It is this program that (via a `python` parser) asks the user what file he wants to use, the number of `NOMAD` runs and if there are any additional files to take into account.

The file itself takes as an input a .mbx (`Minibex`) or `.json` file to which can be added a `param.txt` file as used in `NOMAD`. If a `param.txt` file is provided it will be the one used, If not it will create a suitable param.txt file from the `.mbx` or `.json` file provided using `Lecteur.py`.

### 3.3.2 Lecteur.py

The file Lecteur.py is a function file for `MiNomad.py`. It contains the functions that will create a suitable for optimization `param.txt` file from the `.mbx` of `.json`.

The choice of making a function file more than to add the corresponding functionalities to the `MiNomad.py` file was to allow anyone to use the functions separately.

This also allows anyone to take inspiration from the functions and to create his own Lecteur function and use it in `MiNomad`.

In practice, there are four functions in the file. The functions `LecteurJson` and `LecteurMbx` create an array containing all the information to create a `param.txt` by reading the corresponding type of file. The `LecteurMbx` function will also create a corresponding `.mbx` file for the optimization.

The information present in each file is different, hence the separation, and `LecteurMbx` will use the function `ReadSection` (specific to mbx files).

The last operation made by the two `Lecteur` functions is to write a text file containing the name of the .mbx file used for the optimization. We will explain why in the following part.

Finally, the WriteInfo function will take in argument an array to create a `param.txt` file (retrieved by `MiNomad`) to conduct the optimization.

### 3.3.3 Passerelle.cpp

The final program we will look at is the `Passerelle.cpp`, the core of the three programs. Its role is to take a point sent by the `NOMAD` algorithm and to calculate the value of the objective function and the constraints using `Ibex`.

To do so, Ibex needs a box not a point. We chose to make it so that we do not create a degenerated interval (that is to say an interval where the upper and lower bounds are the same) but a small box around the point, precise at 0.1% of the value of the variable.

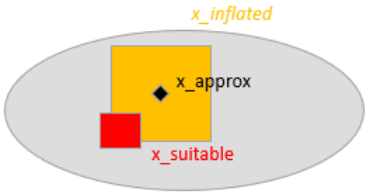



Figure 25: Representation of the problem of equality constraints

This little yellow box in 25 allows us to deal with some difficulties inherent to equality constraints, indeed the points given by NOMAD are indeed truncated and might not respect the said constraints. This is a common way to deal with the constraints imposed by the fact computers are using floats and not real numbers.

The precision can be modified in the file by changing the value of the "prec" parameter.

The Passerelle will finally calculate the value of the constraints and objective function and print the results separated by spaces as NOMAD expects it.

To avoid having to recompile the file each time a new model is used, Passerelle searches for the name of the file to use in a nomad2ibex.txt file that is usually created by Lecteur.py but can be handwritten.

### 3.4 Conclusion

We have presented the three programs that compose our link between Ibex and NOMAD. It is now time to criticize the final product and give some improvement directions.

First of all, while the link works perfectly well in theory, the hypothesis to know a feasible point is quite strong, all the more if there are some equality constraints. Not respecting those conditions can prevent the optimization from starting with NOMAD. Moreover, while there is no way to handle temporary variables, the user also needs to know the approximated value of each of them for his starting point.

Those problems prevented any test on complicated models to be realised easily while the easy problems were to simple to show any meaningful differences. Indeed, using anything but the trivial optimized point ended up in a failed first iteration. What we mean by complicated model is one with intermediate functions, with equality constraints or Jsons with temporary variables because this currently implies the presence of equality constraints during the writing of the .mbx file.

Moreover, if a .mbx file is used, the starting point is defined by the middle of the bounds which is quite unlikely to satisfy the constraints. An easier way to give a starting point in this case than modifying a param.txt file would come quite handy.

To summarize, three improvement axes are quite clear :

- A way to handle the temporary variable when a Json file is used to not add variables in the Minibex file.
- A way to handle intermediate function while reading a Minibex file.
- An easy way to give a custom starting point when using a Minibex file.

Those improvement axis are not especially complex but the time was running short to solve them during the Internship and some other choices (such as the coding of Neural Black Box) were preferred.

# 4 General Conclusion

Now we presented the Library and the link between NOMAD and Ibex. Both the Black Boxes and the White Boxes are available at https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary.git. White Boxes have a .pdf documents giving points to check any other implementations and some Black Boxes have test functions with hard coded points to check that the programs work as expected on another machine (Mario and Neural do not have those because they are stochastic and unseeded).

While being quite useful, this Library is not the first of its kind as the COCO framework [6] and the Optimization Services Project [19] but we already positioned our work toward those. Now a question one could ask himself by then is "why should I chose NOMAD instead of Ibex ?" and vice-versa. The main answer would be "well it depends on your model" but not for efficiency reasons. Let's dive into details.

First of all NOMAD and Ibex have quite different termination criteria, indeed, as we said earlier, Ibex will stop its algorithm once its boxes around the solution are small enough, while NOMAD will stop once "the ball has stopped", that is to say when the mesh gets infinitely fine (even if it can be stopped before). This means that the precision we apply to each algorithm does not means the same thing. This means that if we want to compare the two algorithm, we have to measure the time of execution but even this way is not optimal for it will depend on the computer and the way the code is written.

However, comparing them has in my opinion little interest because they will be used in completely different problems. Indeed NOMAD's main asset is the ability to handle Black Boxes function, something Ibex cannot do at all. This comes with being less effective when it comes to White Boxes because the method stays the same and do not resort to useful functionalities like the one Ibex has and we discussed previously. Moreover, as said before, NOMAD struggles with equality constraints while Ibex is fine with it.

This concludes our work on the establishment of a Optimization problem Library and the work on making a link between Ibex and NOMAD.

## References


[1] IMT Atlantique. *Documentation on Contractors, Ibex*. URL: http://www.ibex-lib.org/doc/solver.html#getting-started.

[2] IMT Atlantique. *Documentation on Contractors, Ibex*. URL: http://www.ibex-lib.org/doc/contractor.html?highlight=contractors#hc4.

[3] Jean Bigeon. *Dimensionnement d'une bobine pas à pas*. URL: %5Curl%7Bhttps://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/blob/main/Mod%C3%A8les/Bobine/Refs/docs/exo%20dimension%20bobine%202021-fr.docx%7D.

[4] Jean Bigeon and Kaveh Babanezhad. *Engineering Model Description | I-Beam*. URL: %5Curl%7Bhttps://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/tree/main/Mod%C3%A8les/Poutre/Refs%7D.

[5] Gilles Chabert. *IbexOpt, Global Optimization Plugin*. URL: http://www.ibex-lib.org/ibexdays2019/ibexdays2019-gilles-chabert2.pdf.

[6] *Comparing Continuous Optimizers*. URL: http://numbbo.github.io/coco-doc/.

[7] *Complete list of parameters*. URL: https://nomad-4-user-guide.readthedocs.io/en/latest/Appendix.html#appendix-parameters.

[8] Sephora Diampovesa. "Modèles de synthèse pour la conception en génie électrique. Application à l'électrification des véhicules." In: (June 2021), pp. 86–91.

[9] Sébastien Le Digabel et al. *NOMAD User Guide*. URL: %5Curl%7Bhttps://www.gerad.ca/software/nomad/Downloads/user_guide.pdf%7D.

[10] *functional mock-up interface*. URL: https://fmi-standard.org/.

[11] *functional mock-up interface - model description*. URL: https://fmi-standard.org/docs/3.0-dev/#modelDescription.xml.

[12] Edward H-Hannan. *pytorch-f-mnist*. URL: https://github.com/edhhan/pytorch-f-mnist.

[13] Charles Audet Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Opreations Research and Financial Engineering. Springer, 2017. ISBN: 978-3-319-68912-8.

[14] *Introducing JSON*. URL: https://www.json.org/json-en.html.

[15] Frédéric Messine, Bertrand Nogarede, and Jean-Louis Lagouanelle. "Optimal Design of Electromechanical Actuators: A New Method Based on Global Optimization". In: *Transactions on magnetics* 34 (1998), pp. 299–308. DOI: https://ieeexplore.ieee.org/document/650361.

[16] David R. Morrison et al. *Branch-and-bound algorithms : a survey of recent advances in searching, branching and pruning*. Vol. 19. 2016, pp. 79–102. DOI: https://doi.org/10.1016/j.disopt.2016.01.005.

[17] *Nomad 4 User Guide*. URL: https://nomad-4-user-guide.readthedocs.io/en/latest/.

[18] *Nomad Usage*. URL: %5Curldef%7B%5Ctechnical%7D%5Curl%7Bhttps://nomad-4-user-guide.readthedocs.io/en/latest/HowToUseNomad.html?highlight=extreme%20barrier#bb-output-type%7D.

[19] *Optimization Services Project*. URL: http://www.optimizationservices.org.

[20] *Optimization Services User Manual*. URL: https://projects.coin-or.org/svn/OS/trunk/OS/doc/osUsersManual.pdf.

[21] Michel Poloujadoff and Raymond D. Findlay. *A Procedure for illustrating the effect of variation of parameters on optimal transformer design*. Nov. 1986. URL: %5Curl%7Bhttps://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/blob/main/Mod%C3%A8les/Transfo2/Refs/Poloujadoff.pdf%7D.

[22] Giulia Pullano, Laura Di Domenico, and Chiara E. Sabbatini. "Underdetection of cases of COVID-19 in France threatens epidemic control". In: *Nature* 590 (2021), pp. 134–139. DOI: https://doi.org/10.1038/s41586-020-03095-6.

[23] Tuan Vu Tran. *Problèmes combinatoires et modèles multi-niveaux pour la conception optimale des machines électriques*. Mar. 2011. URL: %5Curl%7Bhttps://tel.archives-ouvertes.fr/tel-00425590v3%7D.

[24] Vanessa Volz, Günter Rudolph, and Boris Naujoks. *Demonstrating the Feasibility of Automatic Game Balancing*. URL: https://arxiv.org/pdf/1603.03795.pdf.

[25] Vanessa Volz et al. *Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network*. URL: https://arxiv.org/pdf/1805.00728.pdf.

[26] Jiheng Wang. *A random sample of the original hand-written digit images for MNIST data*. URL: https://www.researchgate.net/figure/A-random-sample-of-the-original-hand-written-digit-images-from-MINST-data_fig2_252343403.

[27] Zi Wang et al. *Batched Large-scale Bayesian Optimization in High-dimensional Spaces*. URL: https://arxiv.org/abs/1706.01445.

# 5 Annexes

# Dimensionnement Bobine

## Description du modèle :

Le modèle décrit ci-après est tiré de [1] par M. Jean Bigeon, ORCID : 0000-0002-6112-6913

**N**omenclature :

— $a$ le diamètre moyen de la bobine en $m$
— $b$ la longueur de la bobine en $m$
— $c$ l'épaisseur de la bobine en $m$
— $I$ l'intensité efficace du courant dans la bobine en $A$
— $L$ l'inductance de la bobine en $H$
— $M$ la masse de la bobine en $kg$
— $mv_{cu}$ la masse volumique du cuivre en $kg.m^{-3}$
— $n$ le nombre de spires de la bobine (sans unité)
— $P_J$ les pertes Joule en $W$
— $W_{mag}$ l'énergie magnétique de la bobine en $J$

— $\delta$ la densité efficace de courant dans la bobine en $A.m^{-2}$
— $\rho_{cu}$ la résistivité électrique du cuivre en $\Omega.m$

**E**quations :

— $L = \frac{50.8*10^{-9}*a^2*n^2}{3*a+9*b+10*c}$
— $P_J = \rho_{cu} * \delta^2 * \pi * a * b * c$
— $I = \delta * \frac{b*c}{n}$
— $W_{mag} = \frac{1}{2} * L * I^2$
— $M = mv_{cu} * \pi * a * b * c$

## Cahier des Charges :

Dans cet exemple, on fixe $L$ à $10^{-3}$ ainsi que $mv_{cu}$ et $rho_{cu}$ (en fait fixés par le choix du cuivre comme matériaux) et on cherche à minimiser $M$ en respectant les contraintes détaillées ci-après.

| Variables de Décision | | | | |
|---|---|---|---|---|
| Paramètre | Valeur min | Valeur max | Valeur initiale | Unité |
| $a$ | 0.1 | 0.5 | 0.3 | $m$ |
| $b$ | 0.1 | 1 | 0.5 | $m$ |
| $c$ | 0.005 | 0.1 | 0.01 | $m$ |
| $mv_{cu}$ | 8800 | | | $kg.m^{-3}$ |
| $n$ | 10 | 5000 | 1000 | / |
| $\delta$ | $10^5$ | $10^7$ | $10^6$ | $A.m^{-2}$ |
| $\rho_{cu}$ | $1.724 * 10^{-8}$ | | | $\Omega.m$ |

| Sorties | | | |
|---|---|---|---|
| Paramètre | Type | Valeur | Unité |
| $I$ | $Libre$ | $-$ | $A$ |
| $L$ | $Fixe$ | $10^{-3}$ | $H$ |
| $M$ | $Objectif$ | $[20; 100]$ | $kg$ |
| $P_J$ | $Contraintparintervalle$ | $[70; 90]$ | $W$ |
| $Wmag$ | $Libre$ | $-$ | $J$ |

**F**onction Objectif :

$$f_{obj}(V) = M(V) = mv_{cu} * \pi * a * b * c$$

## Test de Fiabilité :

Afin de vérifier la validité du modèle proposé, il convient de tester ce dernier avec plusieurs sets de valeurs. Vous trouverez ci-après un ensemble de valeurs d'entrée et les résultats attendus sur la base des valeurs de [1].

| Numéro du set | Set 1 | Set 2 |
|---|---|---|
| $a$ | 0.1000 | 0.2999 |
| $b$ | 0.9997 | 0.5001 |
| $c$ | 0.007233 | 0.0999 |
| $n$ | 4294 | 1000 |
| $\delta$ | $1428 * 10^3$ | $1e6$ |
| $I$ | 2.405 | 4.996 |
| $L$ | $1e-3$ | 0.0008306 |
| $M$ | 20 | 41.42 |
| $Pj$ | 79.91 | 81.15 |
| $W_{mag}$ | 0.002892 | 0.01037 |

# Références

[1] J. Bigeon and C. Espanet, "Dimensionnement d'une bobine pas à pas." https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/tree/main/Mod%C3%A8les/Bobine/Refs/docs.

# Dimensionnement de la MAPSE

## Description du modèle :

On considère ici une MAPSE pour Machine à Aimants Permanents Sans Encoches. Le modèle décrit ci-après est tiré de [1] et [2].

**N**omenclature :

— $B_e$ induction à vide dans l'entrefer en $T$
— $B_{iron}$ l'induction du fer en $T$
— $C$ l'épaisseur des culasses en $m$
— $D$ le diamètre d'alésage en $m$
— $e$ l'entrefer mécanique en $m$
— $E$ l'épaisseur d'entrefers en $m$
— $E_{ch}$ l'échauffement de la machine en $A^2.m^{-3}$
— $J_{cu}$ densité de courant dans le cuivre en $A.m^{-2}$
— $k_r$ le coefficient de remplissage du bobinage (sans unité)
— $K_f$ le coefficient de fuites inter-polaires (sans unité)
— $l_a$ l'épaisseur des aimants en $m$
— $L$ la longueur du fer en $m$
— $p$ le nombre de paires de pôles de la machine (sans unité)
— $P$ l'aimantation en $T$
— $P_j$ les pertes par effet Joules en $W$
— $V_a$ le volume des aimants en $m^3$
— $V_u$ le volume des parties utiles en $m^3$

— $\beta$ le coefficient d'arc polaire (sans unité)
— $\Gamma_{em}$ le couple électromagnétique en $N.m$
— $\Delta_p$ le double pas polaire en $m$
— $\lambda$ le facteur de forme de la machine (sans unité)
— $\rho_{cu}$ la résistivité du cuivre en $\Omega.m$

**E**quations :

— $\Gamma_{em} = \frac{\pi}{2\lambda}(1 - K_f)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e$
— $\lambda = D/L$
— $E_{ch} = k_r E J_{cu}^2$
— $K_f = 1.5p\beta \, \frac{e+E}{D}$
— $B_e = \frac{2 l_a P}{D ln[\frac{D+2E}{D-2(l_a+e)}]}$

— $C = \dfrac{\pi \beta B_e}{4 p B_{iron}}\ D$
— $p = \dfrac{\pi D}{\Delta_p}$
— $V_u = \pi \dfrac{D}{\lambda}\ (D + E - e - l_a)(2C + E + e + l_a)$
— $V_a = \pi \beta l_a \dfrac{D}{\lambda}\ (D - 2e - l_a)$
— $P_j = \pi \rho_{cu} \dfrac{D}{\lambda}\ (D + E) E_{ch}$

## Cahier des Charges :

Dans cet exemple, on fixe le couple de la MAPSE ainsi qu'un certain nombre de grandeurs désignées comme paramètres et on cherche les variables de décision qui minimisent le volumes des parties utiles, le volume des aimants et la puissance joules tout en respectant les contraintes.

| Variables de Décision | | | | |
|---|---|---|---|---|
| Paramètre | Valeur min | Valeur max | Valeur initiale | Unité |
| $B_e$ | 0.1 | 1.0 | 0.6284 | $T$ |
| $B_{iron}$ | 1.50 | | | $T$ |
| $C$ | 0.001 | 0.05 | 0.009426 | $m$ |
| $D$ | 0.01 | 0.5 | 0.3183 | $m$ |
| $e$ | 0.0010 | 0.0050 | 0.0025 | $m$ |
| $E$ | 0.001 | 0.05 | 0.00571 | $m$ |
| $E_{ch}$ | $10^{+11}$ | | | $A^2.m^{-3}$ |
| $J_{cu}$ | $1.0 * 10^5$ | $1.0 * 10^7$ | $5.0 * 10^6$ | $A.m^{-2}$ |
| $k_r$ | 0.7 | | | $(/)$ |
| $K_f$ | 0.01 | 0.3 | 0.348 | $(/)$ |
| $l_a$ | 0.001 | 0.05 | 0.025 | $m$ |
| $L$ | 0.004 | 0.5 | 0.25 | $m$ |
| $p$ | 1 | 10 | 8 | $(/)$ |
| $P$ | 0.90 | | | $T$ |
| $\beta$ | 0.8 | 1 | 0.9 | $(/)$ |
| $\Delta_p$ | 0.100 | | | $m$ |
| $\rho_{cu}$ | $0.018 * 10^{-6}$ | | | $\Omega.m$ |

| Sorties | | | |
|---|---|---|---|
| Paramètre | Type | Valeur | Unité |
| $\Gamma_{em}$ | $Fixe$ | 10 | $N.m$ |
| $\lambda$ | $Contraint\ par\ intervalle$ | $[0.02; 125]$ | $(/)$ |
| $V_u$ | $Objectif$ | $-$ | $m^3$ |
| $V_a$ | $Objectif$ | $-$ | $m^3$ |
| $P_j$ | $Objectif$ | $-$ | $W$ |

**F**onction Objectif :

$$f_{obj}(V) = \frac{V_u(V)}{min(V_u)} + \frac{V_a(V)}{min(V_a)} + \frac{P_j(V)}{min(P_j)}$$

## Test de Fiabilité :

Afin de vérifier la validité du modèle proposé, il convient de tester ce dernier avec plusieurs sets de valeurs. Vous trouverez ci-après un ensemble de valeurs d'entrée et les résultats attendus sur la base des valeurs de [1] en ayant corrigé les quelques coquilles qui s'y trouvaient. Ces 4 sets correspondent à la minimisation individuelle des paramètres $V_u$, $V_a$ et $P_j$ pour enfin minimiser la somme normalisée (divisés par les minimums trouvés) de ces trois grandeurs.

| Quantité minimisée | $V_u$ | $V_a$ | $P_j$ | $\frac{V_u}{V_{um}} + \frac{V_a}{V_{am}} + \frac{P_j}{P_{jm}}$ |
|---|---|---|---|---|
| $B_e$ | 0.4536 | 0.1439 | 0.6889 | 0.4435 |
| $C$ | 0.0060 | 0.0019 | 0.0115 | 0.005914 |
| $D$ | 0.3183 | 0.3183 | 0.3183 | 0.3183 |
| $e$ | 0.0010 | 0.0010 | 0.0010 | 0.0010 |
| $E$ | 0.0043 | 0.0043 | 0.0050 | 0.0043 |
| $J_{cu}$ | $5.764 * 10^6$ | $5.764 * 10^6$ | $5.345 * 10^6$ | $5.764 * 10^6$ |
| $K_f$ | 0.1998 | 0.1998 | 0.2827 | 0.1998 |
| $l_a$ | 0.0055 | 0.0010 | 0.0361 | 0.005216 |
| $L$ | 0.0110 | 0.0347 | 0.0067 | 0.011256 |
| $p$ | 10 | 10 | 10 | 10 |
| $\beta$ | 0.8 | 0.8 | 1 | 0.8 |
| $\lambda$ | 28.93 | 9.173 | 47.51 | 14.34 |
| $V_u$ | $2.4991 * 10^{-4}$ | $3.530 * 10^{-4}$ | $3.922 * 10^{-4}$ | $2.4998 * 10^{-4}$ |
| $V_a$ | $0.4726 * 10^{-4}$ | $0.2750 * 10^{-4}$ | $2.1291 * 10^{-4}$ | $0.4594 * 10^{-4}$ |
| $P_j$ | 20.067 | 63.302 | 12.235 | 20.517 |
| $\frac{V_u}{V_{um}} + \frac{V_a}{V_{am}} + \frac{P_j}{P_{jm}}$ | 4.359 | 7.586 | 10.312 | 4.348 |

# Références


[1] A. D. Kane, B. Nogarede, and M. L. Mazenc, "Le dimensionnement des actionneurs é1ectriques : un problème de programmation non linéaire," *Journal de la Physique III*, pp. 293–299, feb 1993.

[2] F. Messine, B. Nogarede, and J.-L. Lagouanelle, "Optimal design of electromechanical actuators : A new method based on global optimization," *IEEE TRANSACTIONS ON MAGNETICS*, pp. 303–307, jan 1998.

# Dimensionnement d'une poutre

## Description du modèle :

Le modèle décrit ci-après est tiré de [1] par M. Jean Bigeon, ORCID : 0000-0002-6112-6913.

Nomenclature :
— $E$ le module de Young en $Pa = kg.m^{-1}.s^{-2}$
— $I$ le moment d'inertie en $m^4$
— $I_x$ le moment d'inertie selon x en $m^4$
— $I_y$ le moment d'inertie selon y en $m^4$
— $L$ la longueur de la poutre en $m$
— $M$ la masse de la poutre en $kg$
— $P$ la force appliquée au centre de masse de la Poutre en $N = kg.m.s^{-2}$
— $S$ la surface de la section de la poutre en $m^2$
— $w_{max}$ la déformation maximum de la poutre en $m$
— $x_1$ la hauteur de la poutre en $m$
— $x_2$ la largeur de la poutre en $m$
— $x_3$ la largeur de la barre centrale de la poutre en $m$
— $x_4$ l'épaisseur des faces de la poutre en $m$
— $\rho_{acier}$ la masse volumique de l'acier en $kg.m^{-3}$
— $\sigma_{max}$ la contrainte maximale appliquée à la poutre en $kg.m^{-1}.s^{-2}$

Equations :
— $S = x_2 * x_1 - (x_1 - 2 * x_4)(x_2 - x_3)$
— $M = L * S * \rho_{acier}$
— $I_x = \frac{(x_1-2*x_4)^3*x_3+2*x_4^3*x_2+6*x_4*x_2*(x_1-x_4)^2}{12}$
— $I_y = \frac{2*x_4*x_2^3+(x_1-2*x_4)*x_3^3}{12}$
— $I = I_x + I_y$
— $\sigma_{max} = \frac{P*L*\frac{x_1}{2}}{4*I_x}$
— $w_{max} = \frac{P*L^3}{48*E*I_x}$

## Cahier des Charges :

Dans cet exemple, on fixe $L$, $E$, $P$ et $\rho_{acier}$ et on cherche à minimiser $M$

en respectant des contraintes sur $\sigma_{max}$ et $w_{max}$

| Variables de Décision | | | | |
|---|---|---|---|---|
| Paramètre | Valeur min | Valeur max | Valeur initiale | Unité |
| $E$ | $2 * 10^{11}$ | | | $kg.m^{-1}.s^{-2}$ |
| $L$ | 4 | | | $m$ |
| $P$ | $10^4$ | | | $kg.m^{-1}.s^{-2}$ |
| $x_1$ | 0.1 | 0.8 | 0.45 | $m$ |
| $x_2$ | 0.1 | 0.6 | 0.35 | $m$ |
| $x_3$ | 0.01 | 0.05 | 0.03 | $m$ |
| $x_4$ | 0.01 | 0.05 | 0.03 | $m$ |
| $\rho_{acier}$ | 7800 | | | $kg.m^{-3}$ |

| Sorties | | | |
|---|---|---|---|
| Paramètre | Type | Valeur | Unité |
| $\sigma_{max}$ | $Intervalle$ | $[0; 12.8 * 10^6[$ | $kg.m^{-1}.s^{-2}$ |
| $w_{max}$ | $Libre$ | $-$ | $m$ |

**F**onction Objectif :

$$f_{obj}(V) = M = L * S * \rho_{acier}$$

## Test de Fiabilité :

Afin de vérifier la validité du modèle proposé, il convient de tester ce dernier avec plusieurs sets de valeurs. Vous trouverez ci-après un ensemble de valeurs d'entrée et les résultats attendus sur la base des valeurs de [1].

| Numéro du set | Set 1 | Set 2 |
|---|---|---|
| $x_1$ | 0.1 | 0.1 |
| $x_2$ | 0.2566 | 0.6 |
| $x_3$ | 0.01 | 0.01 |
| $x_4$ | 0.01 | 0.01273 |
| $S$ | 0.005938 | 0.0160 |
| $M$ | 185.26 | 500 |
| $I_x$ | $3.906 * 10^{-5}$ | $4.879 * 10^{-4}$ |
| $\sigma_{max}$ | $12.79994 * 10^6$ | $1.0247 * 10^6$ |
| $w_{max}$ | $1.706 * 10^{-3}$ | $1.366 * 10^{-4}$ |

# Références

[1] J. Bigeon and K. Babanezhad, "Engineering model description | i-beam." https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/tree/main/Mod%C3%A8les/Poutre/Refs.

# Transformateur de Sécurité

## Description du modèle :

Le modèle décrit ci-après est tiré de la thèse de M. TRAN [1], [2] ansi que de celle de Mme. Diampovesa [3] et fait les suppositions suivantes :
- l'isolant entre le noyau central et les bobines est en parfait contact avec le cuivre et le noyau central.
- le contact entre le secondaire et le circuit magnétique est nul et la couche d'air prisonnière entre les deux est supposée parfaitement isolante
- le contact entre les enroulements et le circuit magnétique (autre que noyau central) est nul ou de mauvaise conduction
- toutes les surfaces verticales ont le même coefficient de convection

Nomenclature :
— $a$ la demie largeur du bras médian du circuit magnétique en $m$
— $b$ la hauteur de la cavité des bobines en $m$
— $B_m$ l'induction maximale en $kg.A^{-1}.s^{-2} = T$
— $c$ la largeur de la cavité des bobines en $m$
— $d$ l'épaisseur du circuit magnétique en $m$
— $e_{isol}$ l'épaisseur de l'isolant entre le fer et le cuivre en $m$
— $f$ la fréquence de fonctionnement du circuit en $s^{-1} = Hz$
— $fp$ le facteur de puissance (sans unité)
— $f_1$ le facteur de remplissage de la bobine primaire en %
— $f_2$ le facteur de remplissage de la bobine secondaire en %
— $h$ le coefficient de convection thermique en $kg.s^{-3}.K^{-1} = W.m^{-2}.K^{-1}$
— $I_1$ le courant dans le circuit primaire en $A$
— $I_2$ le courant dans le circuit secondaire en $A$
— $I_{10}$ le courant de magnétisation en $A$
— $l_{1spire}$ la longueur d'une spire de la bobine primaire en $m$
— $l_{2spire}$ la longueur d'une spire de la bobine secondaire en $m$
— $L_\mu$ l'inductance magnétisante en $kg.m^2.s^{-2}.A^{-2} = H$
— $mv_{copper}$ la masse volumique du cuivre en $kg.m^{-3}$
— $mv_{iron}$ la masse volumique du fer en $kg.m^{-3}$
— $M_{copper}$ la masse de cuivre en $kg$
— $M_{iron}$ la masse de fer en $kg$
— $M_{tot}$ la masse totale du transformateur en $kg$
— $n_1$ le nombre de spires primaires (sans unités)
— $n_2$ le nombre de spires secondaires (sans unités)

— $P_{iron}$ la perte de puissance due au fer en $kg.m^2.s^{-3} = W$
— $P_j$ la puissance Joules en $kg.m^2.s^{-3} = W$
— $P_1$ la puissance du circuit en $kg.m^2.s^{-3} = W$
— $q$ les pertes spécifiques de la tôle en $W.kg^{-1} = m^2.s^{-3}$
— $Q_1$ une puissance en $kg.m^2.s^{-3} = W$
— $r_1$ la résistance de la bobine primaire en $m^2s^{-3}A^{-2}kg = \Omega$
— $r_2$ la résistance de la bobine secondaire en $m^2s^{-3}A^{-2}kg = \Omega$
— $R_{iron-air}$ résistance thermique de convection entre le fer et l'air en $K.kg^{-1}.m^{-2}.s^2 = K.W^{-1}$
— $R_{cond}$ résistance de conduction entre le fer et le cuivre en $K.kg^{-1}.m^{-2}.s^2 = K.W^{-1}$
— $R_{copp-air}$ résistance thermique de convection entre le cuivre et l'air en $K.kg^{-1}.m^{-2}.s^2 = K.W^{-1}$
— $R_2$ la résistance totale ramenée au secondaire en $m^2s^{-3}A^{-2}kg = \Omega$
— $S_{iron-air}$ la surface de contact entre l'air et le fer en $m^2$
— $S_{copp-air}$ la surface de contact entre l'air et le cuivre en $m^2$
— $S_1$ la section du fil de la bobine primaire en $m^2$
— $S_2$ la section du fil de la bobine secondaire en $m^2$
— $T_{ext}$ la Température extérieure en $K$
— $T_{iron}$ la température du fer en régime permanent en $C$
— $T_{copper}$ la température du cuivre en régime permanent en $C$
— $V_1$ la tension dans la bobine primaire en $kg.m^2.s^{-3}.A^{-1} = V$
— $V_2$ la tension dans la bobine secondaire en $kg.m^2.s^{-3}.A^{-1} = V$
— $X_2$ une impédance en $m^2s^{-3}A^{-2}kg = \Omega$
— $\alpha_{cop}$ le coefficient de température du cuivre en $K^{-1}$
— $\Delta V_2$ la chute de tension en $kg.m^2.s^{-3}.A^{-1} = V$
— $\eta$ le rendement en %
— $\lambda$ la conductivité thermique de l'isolant en $kg.m.s{-}3.K^{-1} = W.m^{-1}.K^{-1}$
— $\mu_0$ la perméabilité du vide en $kg.m.s^{-2}.A^{-2} = T.m.A^{-1}$
— $\mu r(\frac{B_m}{1T})$ perméabilité relative des tôles (sans unité)
— $\rho_{cop}$ la résistivité du cuivre en $m^3s^{-3}A^{-2}kg = \Omega.m$

**E**quations :

— $B_m = \frac{V_1 * \sqrt{2}}{4 * \pi * n_1 * a * d * f}$
— $l_{1spire} = 2 * (d + 2 * a) + \pi * \frac{c}{2}$
— $l_{2spire} = 2 * (d + 2 * a) + \pi * c * \frac{3}{2}$
— $\mu r(x) = \dfrac{1}{2.12*10^{-4} + \frac{(1-2.12*10^{-4})*x^{2*7.358}}{x^{2*7.358} + 1.18*10^6}}$
— $L_\mu = \mu_0 * \mu r(\frac{B_m}{1T}) * n_1^2 * \frac{a*d}{2a+b+c}$
— $M_{iron} = mv_{iron} * 4 * a * d * (2 * a + b + c)$

— $P_{iron} = q * M_{iron} * \frac{f}{50Hz} * \frac{B_m}{1T}^2$
— $R_{cond} = \frac{e_{isol}}{\lambda * b * (4*a+2*d)}$
— $S_{copp-air} = b * (4 * a + 2 * \pi * c)$
— $S_{iron-air} = 4 * a * (b + 4 * a + 2 * c) + 2 * d * (6 * a + 2 * c + b)$
— $R_{copp-air} = \frac{1}{h * S_{copp-air}}$
— $R_{iron-air} = \frac{1}{h * S_{iron-air}}$

*Les equations suivantes forment un système non causal.*

— $n_2 = n_1 * \frac{(V_2 + \Delta V_2)}{V_1}$
— $P_j = R_2 * I_2^2$
— $Q_1 = \frac{V_1^2}{L_\mu * 2 * \pi * f} + X_2 * I_2^2 + V_2 * I_2 * sin(acos(fp))$
— $r_1 = \rho_{cop} * (1 + \alpha_{cop} * T_{copper}) * \frac{n_1 * l_{1spire}}{S_1}$
— $r_2 = \rho_{cop} * (1 + \alpha_{cop} * T_{copper}) * \frac{n_2 * l_{2spire}}{S_2}$
— $R_2 = r_2 + (\frac{n_2}{n_1})^2 * r_1$
— $T_{copper} = T_{ext} + R_{copp-air} * \frac{R_{cond} * P_j + R_{iron-air} * (P_j + P_{iron})}{R_{copp-air} + R_{iron-air} + R_{cond}}$
— $T_{iron} = T_{ext} + R_{iron-air} * \frac{R_{copp-air} * P_j + (R_{copp-air} + R_{cond}) * P_{iron}}{R_{copp-air} + R_{iron-air} + R_{cond}}$
— $X_2 = \mu_0 * n_2^2 * c * (4 * a + 2 * d + \pi * c) * \frac{2 * \pi * f}{3 * b}$
— $\Delta V_2 = I_2 * (R_2 * fp + X_2 * sin(acos(fp)))$

*Les equations suivantes découlent su système précédent.*

— $f_1 = \frac{2 * n_1 * s1}{b * c}$
— $f_2 = \frac{2 * n_2 * s2}{b * c}$
— $M_{copper} = mv_{copper} * (n_1 * s_1 * l_{1spire} + n_2 * s_2 * l_{2spire})$
— $M_{tot} = M_{iron} + M_{copper}$
— $P_1 = P_{iron} + P_j + V_2 * I_2 * fp$
— $I_1 = \frac{\sqrt{P_1^2 + Q_1^2}}{V_1}$
— $I_{10} = \sqrt{(\frac{P_{iron}}{V_1})^2 + (\frac{V_1}{L_\mu * 2 * \pi * f})^2}$
— $\eta = \frac{V_2 * I_2 * fp}{V_2 * I_2 * fp + P_{iron} + P_j}$

## Cahier des Charges :

Dans cet exemple, on cherche les valeurs de $a$, $b$, $c$, $d$, $s_1$, $s_2$, $n_1$, $n_2$, $r_1$, $r_2$ et $i_2$ qui minimisent $M_{tot}$ en respectant les contraintes détaillées ci-dessous et les équations présentées précédemment. On notera que les variables géomé-

triques (section, dimensionnement du fer) sont discrètes car les pièces sont commandées dans un catalogue et pas fabriquées sur mesures.

| Variables de Décision | | | | |
|---|---|---|---|---|
| Paramètre | Valeur min | Valeur max | Valeur initiale | Unité |
| $a$ | 0.002 | 1.0 | 0.0225 | $m$ |
| $b$ | 0.006 | 1.0 | 0.095 | $m$ |
| $c$ | 0.0035 | 1.0 | 0.04 | $m$ |
| $d$ | 0.0052 | 1.0 | 0.465 | $m$ |
| $e_{isol}$ | $10^{-3}$ | | | $m$ |
| $f$ | 50 | | | $Hz$ |
| $fp$ | 0.8 | | | $(/)$ |
| $h$ | 10 | | | $W.m^{-2}.K^{-1}$ |
| $I_2$ | 8.0 | 1.0 | $+\infty$ | $A$ |
| $mv_{copper}$ | 8800 | | | $kg.m^{-3}$ |
| $mv_{iron}$ | 7800 | | | $kg.m^{-3}$ |
| $n_1$ | 200 | 1.0 | 1200 | $(/)$ |
| $n_2$ | 1 | 1.0 | 10000 | $(/)$ |
| $q$ | 1.0 | | | $m^2.A^2.kg^{-2}$ |
| $r_1$ | 0 | 1.0 | $+\infty$ | $\Omega$ |
| $r_2$ | 0 | 1.0 | $+\infty$ | $\Omega$ |
| $s_1$ | $0.05515 * 10^{-6}$ | 1.0 | $19.635 * 10^{-6}$ | $m^2$ |
| $s_2$ | $0.05515 * 10^{-6}$ | 1.0 | $19.635 * 10^{-6}$ | $m^2$ |
| $T_{ext}$ | 40 | | | $C$ |
| $V_1$ | 230 | | | $V$ |
| $V_2$ | 24 | | | $V$ |
| $\alpha_{cop}$ | $3.8 * 10^{-3}$ | | | $K^{-1}$ |
| $\lambda$ | 0.15 | | | $W.m^{-1}.K^{-1}$ |
| $\mu_0$ | $4\pi * 10^{-7}$ | | | $T.m.A^{-1}$ |
| $\rho_{cop}$ | $1.72 * 10^{-8}$ | | | $\Omega.m$ |

| Sorties | | | |
|---|---|---|---|
| Paramètre | Type | Valeur | Unité |
| $T_{copper}$ | $Intervalle$ | $[0; 120]$ | $C$ |
| $T_{iron}$ | $Intervalle$ | $[0; 100]$ | $C$ |
| $\eta$ | $Intervalle$ | $[0.8; 1]$ | $(\%)$ |
| $\frac{\Delta V_2}{V_2}$ | $Intervalle$ | $[0; 0.1]$ | $(/)$ |
| $\frac{I_{10}}{I_1}$ | $Intervalle$ | $[0; 0.1]$ | $(/)$ |
| $M_{tot}$ | $Intervalle$ | $[0; 2.6]$ | $kg$ |
| $f_1$ | $Intervalle$ | $[0; 0.5]$ | $(\%)$ |
| $f_2$ | $Intervalle$ | $[0; 0.5]$ | $(\%)$ |

Le reste des sorties n'est pas contraint et n'est pas dans le tableau pour ne pas l'encombrer inutilement.

**F**onction Objectif :

$$f_{obj}(V) = M_{tot} = M_{iron} + M_{copper}$$

## Test de Fiabilité :

Afin de vérifier la validité du modèle proposé, il convient de tester ce dernier avec plusieurs sets de valeurs. Vous trouverez ci-après un ensemble de valeurs d'entrée et ainsi que les résultats attendus pour les sorties d'intérêt sur la base des valeurs de [1].

| Grandeur | Set 1 | Set 2 |
|---|---|---|
| $a$ | $18 * 10^{-3}$ | $6.165 * 10^{-3}$ |
| $b$ | $54 * 10^{-3}$ | $7.006 * 10^{-2}$ |
| $c$ | $18 * 10^{-3}$ | $7.731 * 10^{-3}$ |
| $d$ | $33.5 * 10^{-3}$ | 0.1726 |
| $I_2$ | 8.288 | 8.165 |
| $n_1$ | 722 | 366 |
| $n_2$ | 82 | 42 |
| $s_1$ | $0.3318 * 10^{-6}$ | 0.2121 |
| $s_2$ | $2.835 * 10^{-6}$ | 2.703 |
| $B_m$ | 1.189 | 1.330 |
| $M_{copper}$ | 0.811 | 103.0 |
| $M_{iron}$ | 2.032 | 97.72 |
| $M_{tot}$ | 2.844 | 3.658 |
| $P_i$ | 2.873 | 5.288 |
| $P_j$ | 16.999 | 23.92 |
| $L_{mu}$ | 16.41 | 7.413 |
| $R_2$ | 36.13 | 0.3589 |
| $T_{copper}$ | 108.98 | 103.0 |
| $T_{iron}$ | 98.55 | 97.72 |
| $\eta$ | 0.88 | 0.8430 |
| $\frac{I_{10}}{I_1}$ | 0.047 | 0.09994 |
| $\frac{\Delta V_2}{V_2}$ | 0.087 | 0.09973 |
| $f_1$ | 0.493 | 0.2866 |
| $f_2$ | 0.478 | 0.4191 |

# Références


[1] T. V. Tran, "Problèmes combinatoires et modèles multi-niveaux pour la conception optimale des machines électriques." `https://tel.archives-ouvertes.fr/tel-00425590v3`, March 2011.

[2] T. V. Tran, "Design of a safety isolating transformer." `http://optimisation.l2ep.ec-lille.fr/benchmarks/safety_transformer/files/safety_transformer_equations.pdf`.

[3] S. Diampovesa, "Modèles de synthèse pour la conception en génie électrique. application à l'électrification des véhicules.," pp. 86–91, June 2021.

# Dimensionnement Transformateur 2

## Description du modèle :

Le modèle décrit ci-après est tiré de [1] par M. Jean Bigeon, ORCID : 0000-0002-6112-6913 et de [2].

Nomenclature :

- $A$ l'épaisseur de la bobine primaire en $m$
- $A_L$ l'aire de la section des jambes du transformateur en $m^2$
- $B_T$ la densité de flux en $T = kg.A^{-1}.s^{-2}$
- $D_1$ l'espace entre la bobine primaire et la jambe du bâti en $m$
- $D_2$ l'espace entre la bobine primaire et la bobine secondaire en $m$
- $D_3$ l'espace entre le sommet des bobines et le haut du bâti en $m$
- $D_4$ l'espace entre le bas des bobines et la base du bâti en $m$
- $D_5$ l'espace entre la bobine secondaire et la limite d'inter-phase $m$
- $D_M$ le diamètre moyen du transformateur en $m$
- $f$ la fréquence de fonctionnement du système en $Hz = s^{-1}$
- $F_I$ le facteur de remplissage du fer sans unité.
- $F_F$ le facteur de forme sans unité.
- $F_1$ le facteur de remplissage de la bobine primaire sans unité
- $F_2$ le facteur de remplissage de la bobine secondaire sans unité
- $G$ l'épaisseur de la bobine secondaire en $m$
- $h$ la hauteur de la bobine en $m$
- $J$ la densité de courant volumique en $A.m^{-2}$
- $L_D$ le diamètre de la jambe en $m$
- $N_1$ Nombre de tours dans la bobine primaire sans unité.
- $P_c$ le prix massique du cuivre en $\$.kg^{-1} = kg^{-1}$
- $P_i$ le prix massique du fer en $\$.kg^{-1} = kg^{-1}$
- $P_C$ le prix total de la masse de cuivre en \$
- $P_I$ le prix total de la masse de fer en \$
- $PC_C$ les pertes énergétiques dues au cuivre en $W$
- $PC_I$ les pertes énergétiques dues au fer en $W$
- $PSPC$ le coût des pertes dues au cuivre en $\$.W^{-1}$
- $PSPI$ le coût des pertes dues au fer en $\$.W^{-1}$
- $S$ la puissance apparente par jambe en $W = kg.m^2.s^{-3}$
- $S_T$ la puissance totale apparente en $W = kg.m^2.s^{-3}$
- $T_C$ le coût total dû au cuivre en \$
- $T_I$ le coût total dû au fer en \$
- $U_1$ la tension du système en $V = kg.m^2.s^{-3}.A^{-1}$

— $V_1$ la tension efficace du système en $V = kg.m^2.s^{-3}.A^{-1}$
— $V_C$ le volume de cuivre en $m^3$
— $V_I$ le volume de cuivre en $m^3$
— $X$ la réactance en %
— $X_2$ la réactance en $\Omega$
— $\rho$ la résistivité électrique du cuivre en $\Omega.m$
— $\rho_I$ la masse volumique du fer en $kg.m^{-3}$
— $\rho_C$ la masse volumique du cuivre en $kg.m^{-3}$
— $\mu_0$ la perméabilité du vide en $kg.m.A^{-2}.s^{-2}$

**E**quations :
— $S = \frac{S_T}{3}$
— $V_1 = \frac{U_1}{\sqrt{3}}$
— $A = \frac{N_1 S}{V_1 h F_1 J}$
— $G = \frac{N_1 S}{V_1 h F_2 J}$
— $F_F = \frac{D_2 + \frac{A+G}{3}}{h}$
— $L_D = \sqrt{\frac{2\sqrt{2} V_1}{\pi^2 f B_T N_1 F_1}}$
— $D_M = L_D + 2D_1 + 2A + D_2$
— $X_2 = \mu_0 * \pi * D_M * N_I^2 * (2 * \pi * f) * F_F$
— $X = \frac{X_2 * S}{V_1^2}$
— $A_L = \frac{\pi * L_D^2}{4}$
— $V_C = 3 * \pi * D_M * h * (A * F_1 + G * F_2)$
— $V_I = A_L * F_I * (8 * (D_1 + A + D_2 + G + D_5) + 6 * L_D + 3 * (h + D_4 + D_3))$
— $P_C = P_c * \rho_C * V_C$
— $P_I = P_i * \rho_I * V_I$
— $PC_C = \rho * V_C * J^2$
— $PC_I = \rho_I * V_I * (1.996 - 8.125 * B_T + 12.277 * B_T^2 - 7.502 * B_T^3 + 1.702 * B_T^4)$
— $T_C = PSPC * PC_C$
— $T_I = PSPI * PC_I$

## Cahier des Charges :

Dans cet exemple, on choisit comme variables $h$ et $N_1$ en fixant le reste des paramètres pour minimiser $P_I + T_I + P_C + T_C$.

| Variables de Décision | | | | |
|---|---|---|---|---|
| Paramètre | Valeur min | Valeur max | Valeur initiale | Unité |
| $B_T$ | 1.7 | | | $T$ |
| $D_1$ | 0.05 | | | $m$ |
| $D_2$ | 0.05 | | | $m$ |
| $D_3$ | 0.05 | | | $m$ |
| $D_4$ | 0.05 | | | $m$ |
| $D_5$ | 0.05 | | | $m$ |
| $f$ | 50 | | | $Hz$ |
| $F_I$ | 0.8 | | | $(/)$ |
| $F_1$ | 0.7 | | | $(/)$ |
| $F_2$ | 0.7 | | | $(/)$ |
| $h$ | 0.4 | 50.2 | 100 | $m$ |
| $J$ | $4.5 * 10^6$ | | | $A.m^{-2}$ |
| $N_1$ | 100 | 350 | 600 | $(/)$ |
| $P_c$ | 25 | | | $\$.kg^{-1}$ |
| $P_i$ | 12 | | | $\$.kg^{-1}$ |
| $PSPC$ | 5 | | | $\$.W$ |
| $PSPI$ | 25 | | | $\$.W$ |
| $S_T$ | $4 * 10^7$ | | | $W$ |
| $U_1$ | $6 * 10^4$ | | | $V$ |
| $\rho$ | $2.6 * 10^{-8}$ | | | $\Omega.m$ |
| $\rho_C$ | 8900 | | | $kg.m^{-3}$ |
| $\rho_I$ | 7800 | | | $kg.m^{-3}$ |
| $\mu_0$ | $4\pi * 10^{-7}$ | | | $kg.m.A^{-2}.s^{-2}$ |

| Sorties | | | |
|---|---|---|---|
| Paramètre | Type | Valeur | Unité |
| $A$ | *Libre* | – | $m$ |
| $A_L$ | *Libre* | – | $m^2$ |
| $D_M$ | *Libre* | – | $m$ |
| $F_F$ | *Libre* | – | $(/)$ |
| $G$ | *Libre* | – | $m$ |
| $L_D$ | *Libre* | – | $m$ |
| $P_C$ | *Libre* | – | \$ |
| $P_I$ | *Libre* | – | \$ |
| $PC_C$ | *Libre* | – | $W$ |
| $PC_I$ | *Libre* | – | $W$ |
| $PSPC$ | *Libre* | – | $\$.W^{-1}$ |
| $PSPI$ | *Libre* | – | $\$.W^{-1}$ |
| $T_C$ | *Libre* | – | \$ |
| $T_I$ | *Libre* | – | \$ |
| $V_C$ | *Libre* | – | $m^3$ |
| $V_I$ | *Libre* | – | $m^3$ |
| $X$ | *Libre* | – | % |
| $X_2$ | *Libre* | – | $\Omega$ |

**F**onction Objectif :

$$f_{obj}(V) = P_I + T_I + P_C + T_C$$

## Test de Fiabilité :

Afin de vérifier la validité du modèle proposé, il convient de tester ce dernier avec plusieurs sets de valeurs. Vous trouverez ci-après un ensemble de valeurs d'entrée et les résultats attendus sur la base des valeurs de [2].

| Numéro du set | Set 1 | Set 2 | Set 3 | Set 4 |
|---|---|---|---|---|
| $h$ | 0.727 | 0.4 | 0.4 | 100 |
| $N_1$ | 290 | 100 | 600 | 600 |
| $S$ | $\frac{4*10^7}{3}$ | $\frac{4*10^7}{3}$ | $\frac{4*10^7}{3}$ | $\frac{4*10^7}{3}$ |
| $V_1$ | $3.464 * 10^4$ | $3.464 * 10^4$ | $3.464 * 10^4$ | $3.464 * 10^4$ |
| $A$ | 0.04876 | 0.3055 | 0.1833 | 0.0007331 |
| $G$ | 0.04876 | 0.3055 | 0.1833 | 0.0007331 |
| $F_F$ | 0.1135 | 0.1759 | 0.4305 | 0.0005049 |
| $L_D$ | 0.7095 | 1.208 | 0.4933 | 0.4933 |
| $D_M$ | 0.9570 | 1.419 | 1.010 | 0.6447 |
| $X_2$ | 11.33 | 3.097 | 194.1 | 0.1453 |
| $X$ | 0.1259 | 0.03441 | 2.157 | 0.001615 |
| $A_L$ | 0.3954 | 1.147 | 0.1911 | 0.1911 |
| $V_C$ | 0.4475 | 0.2288 | 0.9769 | 0.6237 |
| $V_I$ | 2.757 | 9.575 | 1.314 | 46.55 |
| $P_C$ | $9.956 * 10^4$ | $5.092 * 10^4$ | $2.174 * 10^5$ | $1.388 * 10^5$ |
| $P_I$ | $2.581 * 10^5$ | $8.962 * 10^5$ | $1.230 * 10^5$ | $4.357 * 10^6$ |
| $PC_C$ | $2.356 * 10^5$ | $1.205 * 10^5$ | $5.143 * 10^5$ | $3.284 * 10^5$ |
| $PC_I$ | $2.198 * 10^4$ | $7.632 * 10^4$ | $1.047 * 10^4$ | $3.711 * 10^5$ |
| $T_C$ | $1.178 * 10^6$ | $6.024 * 10^5$ | $2.572 * 10^6$ | $1.642 * 10^6$ |
| $T_I$ | $5.495 * 10^5$ | $1.908 * 10^6$ | $2.618 * 10^5$ | $9.276 * 10^6$ |
| $f_{obj}$ | $2.085 * 10^6$ | $3.458 * 10^6$ | $3.174 * 10^6$ | $15.41 * 10^6$ |

# Références


[1] J. Bigeon, "Modèle du transformateur." `https://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/blob/main/Mod%C3%A8les/Transfo2/Refs/Mod%C3%A8le%20du%20transformateur.docx`.

[2] M. Poloujadoff and R. D. Findlay, "A procedure for illustrating the effect of variation of parameters on optimal transformer design." `ttps://gitlab.univ-nantes.fr/chenouard-r/optimizationbenchmarklibrary/-/blob/main/ModÃÎles/Transfo2/Refs/Poloujadoff.pdf`, November 1986.