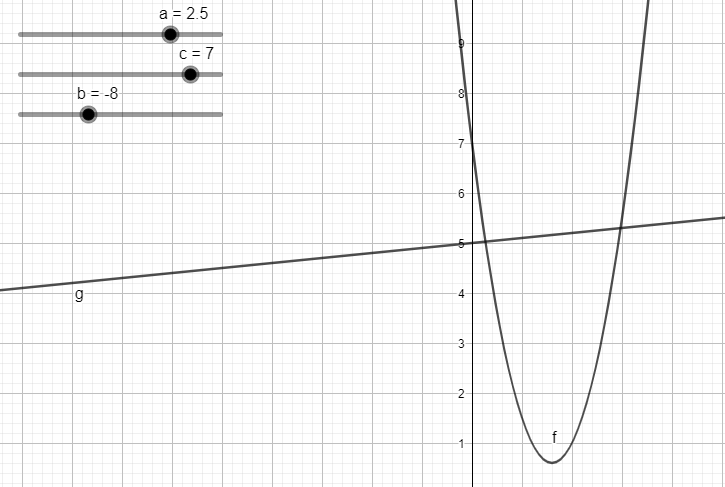
In this tutorial, we will explain how to add a model to the library.  
We will consider the following problem :



We consider the function f(x) = ax^2 +bx +c with a=2.5 b=-8 and c=7 and g(x)=alpha x + beta with alpha =0.1 and beta =5

We want the minimum of f over x for f(x)<=g(x) (which is trivial but will serve as example).

The studied space for x is [0;5] and the result will be the Trivial file under /Modèles

The name of your model file must represent the physical phenomenon depicted and contain one directory per optimization tool usable. The following model must respectively be in the Ibex, Ibex and Nomad directories.

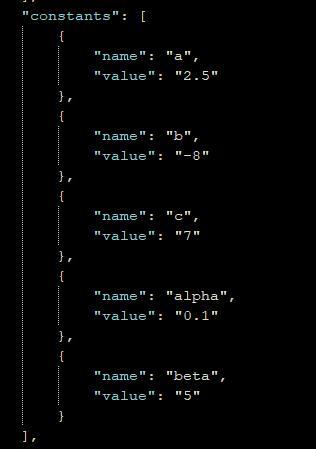
* Json file :

To fill the json file, first complete the variables, there are only on in this case, x :

Une image contenant texte

Description générée automatiquement

Then, you’ll need to fill the constants, there are five, a,b,c,alpha and beta



Next step is the constraints. Here we will reproduce the constraint that f(x)<=g(x) ⬄ f(x)-g(x)<=0 which is the form we want this equation to be, the expression is the quantity that must be compared to 0 and type is the way of comparing to 0 (< or =). Finally if we want to allow this constraint to be relaxed, we must attest “relaxable”: true :

Une image contenant texte

Description générée automatiquement

The last step is to give the quantity that must be minimized here f(x)



Please note that as much as possible, no intermediate variable must be introduced, for example, we will not minimize f with f=ax^2+bx+c.

* Minibex file :

The minibex file is quite similar to the json file as the former is inspired by the first. There is a Constant section, a Variables one, a Minimize that is followed by the objective function and a Constraint section :

Une image contenant texte

Description générée automatiquement

* Nomad file :

The Nomad version of this model requires three files, the blackbox and the param.txt file.

As we are code a whitebox model, we’ll have to code a cpp file taking in input the values of the variables (here x) and output the values of the objective function and constraints.

Une image contenant texte

Description générée automatiquement

In this bloc, only the “else” and the end of loop criteria (I < X) must be adapted. You’ll have to keep note of the order of the variables, it must be the same in the param.txt file.

See the final state of the cpp file :

Une image contenant texte

Description générée automatiquement

We declared the constants, the returned quantities and the variable vector before the loop. At the end of the loop, give its value to the objective function and intermediate variables (if needed) and then, if no error is detected affect the value to constraints.

Once the file is written, compile it with “g++ -o bb.exe bb.cpp” in a shell.

However, if you want to use Nomad, you should have an external executable that functions the same way as the exposed bb.cpp file.In this case, adapt the name to your executable in the param.txt file.

Then, we’ll have to write the param.txt file that will be used by nomad.

Une image contenant texte

Description générée automatiquement

This is a text file containing all the information required by Nomad to run the optimization as to know, which executable to use, the bounds of the variables in order as well as the starting value for the variables. You’ll also need to precise the number of variables, the number of evaluations and finally the structure of the output, beginning with the OBJ (the objective function value) and the constraints precising if they are relaxable (PB) or non-relaxable (EB). This is just a quick overviex, for more information, please consult <https://nomad-4-user-guide.readthedocs.io/en/latest/Appendix.html?highlight=param>

This concludes our tutorial, please note that as well as the Trivial model, a Template file is given to facilitate the implementation of new models.