Week 5

Monday meeting:

Graphical interface for user (Graph in the paper)
Summarize, Improve, Non-dominated

Function treats the P front removing dominant solutions. (not perfect fobj)
Convergence is missing -> need to improve the P strategy
=> Alternatives
All the points without the curve - > hole
Bad example

Post processing aspects

- Evolution of x (plot the curve min/max)

Be able to click -> GUI

Post processing – change the specification, obtain result from the previous optimization
Load optimization results

Major work:
1/ pareto front - change strategy (alternatives) ? -> Algorithm problem, too difficult for me right now!
2/ post process -> save result (I/O each iteration) in some file (done?)
-> load result (stop and start from the saved point) (Load result to run the next session)
-> GUI (clicking -> comparing 2 windows) (tkinter-gui) (moving block around!)

NoLoadj project - Phan Manh Tung - Week 4 (Report 4 - 27-28/03/2023)

I. NoLoad-jax source code - A closer look

I will arrange the coding files/classes starting from highest level of abstraction.

+/ optimProblem.py : the 'OptimProblem' class defines the major input from the user (mono-objective or multiobjective problem, the actual objective functions, specification, method to run)

- method2d=='epsconstr' (normal epsilon-constraint strategy) / 'ponderation'- weighting (maybe this is the 'advanced' epsilon-constraint strategy)
- Functions: SLSQP, IPOPT, LSSQ (optional optimization methods, prebuilt in Scipy)

+/ multiobjective.py : the EpsilonConstraint class is the core code defining **our strategy** to solve a multi-objective problem

- For 'Ponderation': optim2D_weighting, optim_with_param
- For 'Epsconstr': optim2D
- modifySpec_Optim: remove a chosen objective function and append it as a constraint. Afterward, perform an optimization with a chosen model (SLSQP, IPOPT, LSSQ)

+/ specifications.py : the class Spec defines all specifications including objectives and constraints, and essential methods to modify the specification such as insert, delete ..

- removeObjective: Removes a function from the objectives of the model
- insertObjective: Adds a function to the objectives of the model.
- appendConstraint: Adds an equality constraint.
- removeLastEqConstraint: Removes the last equality constraint from the model.

+/ paretoTools.py : a few calculation between coordinates (x, y) in 2D space.

- getXorYmid: Finds the middle point between x and y. With a specific way.
- maxDist:
- max_dist_pair:

II. Pseudocode for method2d='epsconstr' / 'ponderation'

1/ ‘epsconstr’: In the next step, I try to have a closer look at the ‘optim2D’ function (multiobjective.py) to have a pseudocode for our normal epsilon-constraint strategy

```
#2 optim sans contraintes :
#optim du premier objectif, suppression du 2ème objectif (objectif 1)
self.modifySpec_Optim(x0, options, 1)
xm = self.pareto.pts[0][0]
yM = self.pareto.pts[0][1]

#optim du 2ème objectif, suppression du 1er objectif (objectif 0)
self.modifySpec_Optim(x0, options, 0)
xM = self.pareto.pts[1][0]
ym = self.pareto.pts[1][1]

normX = xM-xm
normY = yM-ym

#optimisation avec l'autre objectif contraint
x0 = self.pareto.vars[0]    # on part de la solution optimale pour
# fobX et on contraint sa valeur en augmentant
x0 = self.modifySpec_Optim(x0, options, 0, (2*xm+xM)/3)   #calcul un
# peu à gauche du milieu (mieux que le milieu pour une courbe)

#on reinitialise x0 à la valeur d'extremité du pareto trouvée au début.
for i in range(1,optimNb-2):
    obj, point = getXorYmid(self.pareto.pts, normX, normY)
    x0 = self.modifySpec_Optim(x0, options, obj, point)
    #print("add a point :"+ str(obj) +" / " + str(point))

#à la fin
self.w.resultsHandler = self.resultsHandler
return self.pareto
```

According to the code, I could formulize an pseudo code as follows:

1. x_max, y_min <- the result from the optimization according to the first objective function, with the second objective function added to the constraint.

2. x_min, y_max <- the result from the optimization according to the second objective function, with the first objective function added to the constraint.
3. norm_x <- x_max - x_min, norm_y <- y_max - y_min
   optimNb <- expected number of points in the Pareto front
4. Find optimNb - 2 middle points with function method getXorYmid(xy, normX, normY) to perform optimization steps, save the calculated results.

Question:
The strategy for finding the middle points with the method getXorYmid(xy, normX, normY)

```
x0 = self.modifySpec_Optim(x0, options, 0, (2*xm+xM)/3)   #calcul un
   # peu à gauche du milieu (mieux que le milieu pour une courbe) ?
```

getXorYmid ?

Answer from Benoit: Il s'agit de commencer non pas au milieu entre min et max, mais de manière un peu asymétrique d'où cette pondération de 2 sur le xmin pour le calcul de moyenne.

justification : J'ai constaté par expérience qu'une répartition des points non symétrique était préférable. Typiquement méthode du nombre d'or plutôt que méthode de dichotomie (=couper en 2)...

getXorYmid vise à trouver le lieu du nouveau point là où la distance verticale ou horizontal entre 2 points successifs du front de Pareto est la plus grande.

2/ 'ponderation': this is not the advanced option but the “weighted-sum method”.

```
self.optim_with_param(0.0,x0,options) #left
[xM,ym]=self.pareto.pts[0] # normalisation
xM=(xM-self.fobjs_val[0][0])/(self.fobjs_val[0][1]-self.fobjs_val[0][0])
ym=(ym-self.fobjs_val[1][0])/(self.fobjs_val[1][1]-self.fobjs_val[1][0])

self.optim_with_param(1.0,x0,options) #right
[xm,yM]=self.pareto.pts[1] # normalisation
xm=(xm-self.fobjs_val[0][0])/(self.fobjs_val[0][1]-self.fobjs_val[0][0])
yM=(yM-self.fobjs_val[1][0])/(self.fobjs_val[1][1]-self.fobjs_val[1][0])

otherspts=optimNb-2
i=0 # initialisation de l'algorithme de dichotomie
left,right=0,1

while i<otherspts: #dichotomie
    param=(left+right)/2
    self.optim_with_param(param,x0,options
    [xmid,ymid]=self.pareto.pts[i+2]
    normidleft=np.sqrt((xM-xmid)**2+(ym-ym
    normidright=np.sqrt((xm-xmid)**2+(yM-y
    if normidright<normidleft:
        xm,yM=xmid,ymid
        right=param
    else:
        xM,ym=xmid,ymid
        left=param
    i+=1
    x0=self.pareto.vars[-1] # dernier point calculé

# à la fin
self.w.model=model
```

1. Write a function that solve single optimization problem with the form: fobj=w*f1+(1-w)*f2 (optim_with_param).
2. w<-0 : find the left point for the Pareto front.
3. w<-1: find the right point for the Pareto front.
4. Systematically choose the rest n-2 points for the Pareto front.

Q's: clarify the system of choosing the n-2 points for both weighted-sum and epsilon-constraints (most important part of the algorithm)
-> Discuss in the meeting 4.3.2023

What is JAX package?
Lightweight NumPy-like API for array-based computing.
Composable function transformations.
(autodif, JIT compilation, vectorization, parallelization).
-> powerful system for building model from Scratch.

How to use: “import jax.numpy as np” and the API is the same with normal numpy. Additionally we have the compasable-transformation aspects:

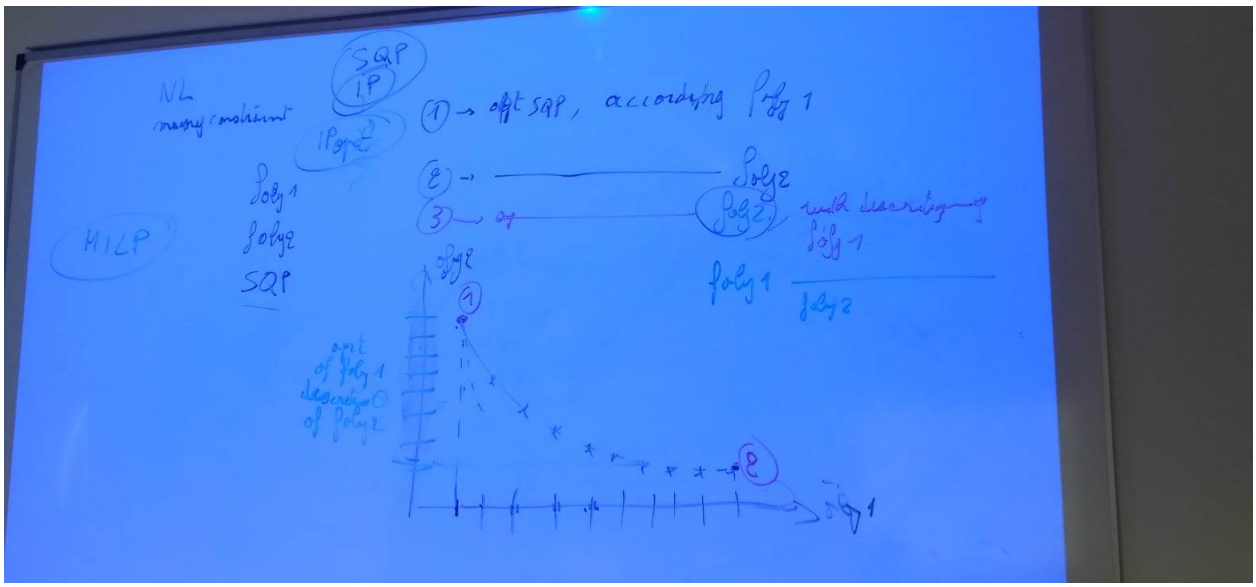
```
from jax import grad, map, jit
gradient_fun = jit(grad(mse_loss)
perexample_grads = jit(vmap(grad(mse_loss), in_axes=(None, 0)))
parallel_grads = jit(pmap(grad(mse_loss), in_axes=(None, 0)))
```

NoLoadj project - Phan Manh Tung - Week 3 (Report 3 - 20-21/03/2023)

I. Supervisors' advice summarization

(Prof. Benoit Delinchant - email)
Regarding Pareto front, don't forget to analyse the existing from NoLoad which is an **'advanced' epsilon constrained technique.**
1 - write a pseudo code of the algorithm and discuss it with us to check that everything is understood
2 - improve pareto front by analysing the solutions set and removing dominated solutions.
3 (later) - generalize bi-objective by providing independance with the optimization algorithm. The bi-objective has to be available for SLSQP and DifferentialEvolution or any constrained scalar optimzer that we want to use.
Before do this previous objective you have to fulfill GUI objectives defined by Laurent :
1 - selecting input/output to plot (x-axis : iteration, or variable)
2 - display bounds
3 - generate results file using CADES syntax (to link with geomaker, to be reloaded for reinitiatization of initial guess before a new optimization)

(Prof. Laurent Gerbaud - Monday discussion)



a. Pseudocode - Our strategy

1. Using SQP optimization method according to f1, find the first anchor point of the Pareto front.

2. Using SQP optimization method according to f2, find the second anchor point of the Pareto front.

3. Using SQP optimization method according f2, with discretizing f1 (divide the constraint of f1 into n even point, then solve n sub-problems with f2). Or vice versa.

4. Obtaining the Pareto front by connect n optimizing points in the step 3 with 2 anchor points in step 1, 2.

b. Interface

- Have the min-max constraints and out of the constraints evolution. Important, for example we have to manage 1000 inputs !!

- Able to Plot 1 var according to another.

- Allow user to perform each step: (first find 2 anchor points, then find 10 points in-between)

- Important to be able to analyse for each variable, to be sure to have conversions.

c. Optimization algorithm

IPOPT, SQP (Sequential quadratic programming) is currently the best technology for nonlinear optimization; MiLP (Mixed-Integer Linear Programming) is the best one for linear optimization.
SQP is prebuilt with Scipy Python (scipy.optimize.minimize(method='SLSQP') solver) and in Matlab.

d. Expected works

Pareto front (2 3 weeks)
Then, other analysis

II. Analysis on example code with NoLoad-Jax package (exemples_noload_pareto.ipynb)

Link: https://colab.research.google.com/drive/1yN9mDe4LI6ouuhOryBm2opYgau470pNB?usp=sharing

What is Jax package? Jax is like Numpy but with better performance.
Official doc: https://jax.readthedocs.io/en/latest/notebooks/quickstart.html

I follow the example code and read the source code of the NoLoad-jax, trying to understand the built-in functions, as follows:

+ `noloadj.tutorial.plotTools - plot3D`: Plot the formation of given functions with x, y as variables.
+ `noloadj.analyse.simulation - computeOnce`: Compute the value of given functions (objective function and constraints) with a set of specific values for x and y
+ `noloadj.analyse.simulation - computeParametric`: Fix y and vary x, then compute all the outputs of all functions
+ `noloadj.analyse.simulation - computeJacobian`: This function calculate the value and the first-order gradient with respect to constraints of a function, given specific input (x,y)
  (Jacobian matrix: contains all first-order partial derivative)
+ `noloadj.optimization.optimProblem - Spec`: provide the specifications of an optimization problem.
+ `noloadj.optimization.optimProblem - OptimProblem`: solve the optimization problem using SLSQP in Scipy.
  After getting the result with `OptimProblem(..).run()` :
+ result.`printResults`: print out all variables after conversion: x, y, fobj, constraints.
+ result.`plotResults(['fobj','ctr1','ctr2'])`: plot the value of each variable with its upper/lower constraints for each iteration (evolution process).
+ result.`exportToXML`: allow user to save to result to .xml (probably for the initialization of the next process).

+ result.`plotPareto(['problem_name'],'Pareto Front')`: plot the Pareto front after optimization process (problem with 2 objective functions).

Notes:

+ The process of initialization is the same for a single-objective problem or a multi-objective problem.
+ `optim.run(method2d='epsconstr')`: (for 2 objective functions) using the epsilon-constraint strategy to obtain the Pareto front.
+ For the last 2 scripts in the example code, we simulated the epsilon-constraint strategy by running the SQP optimization algorithm according to fx, with discretizing fy, with 10 point in-between. Afterward, connect all the points to get the final Pareto front.

NoLoadj project - Phan Manh Tung - Week 2 (Report 2 - 13-14/03/2023)

I. Note for 3 discussions with the supervisors

a. Answer to the proposed plan via email
(1. Focus on the underlined multi-objective algorithm to improve its performance.
2. Try to improve the users' experience by working mainly on the input-output process and visualization. Consider the algorithm as a black box.)

1. Analyse the existing strategies (epsilon constraint, NSGA-II). Formalize with pseudocode. Then, try out SLSQP from Scipy.

2. Try to improve the algorithms or its robustness.

3. (Important aspect!) How to treat constrained optimization (Ex: Differential Evolution from Scipy)

b. Online meeting with Lucas Agobert (07/03/2023)

Topic: A supposed direction with the underlining algorithm

- A better strategy of drawing a point in the Pareto front
- Focus more on epsilon constraint _ gradient-based method (NSGA II is more for genetics)
- Human Interface: a demo for the software next Monday.
- Proposed article: Contribution à l'optimisation Multi-objectifs sous contraintes- A Tchvagha Zeine.

c. Online meeting with Prof. Laurent Gerbaud (13/03/2023)

- Topic: Human Interface/Interaction proposed features for NoLOAD.
- A detailed important-point summarization is presented blow.
- Next meeting: on Monday 20/03/2023 - 9 am at the G2ELab.

II. Analysis of existing strategies

1. Population-based method: NSGA-II Optimization: (a brief lookup)

- Individual properties: Rank, Domination count, Dominates, ..*Constraint Count*
- Non-dominated sorting
- Crowding distance sorting
- Gene mutation, crossover -> genetics
- Implementation: pymoo

2. Scalarization method: (our major focus)

Definition:

- Turn a multi-objective function (f1, f2, fn) into single-objective sub-problems (P1, P2, Pn).
- Weight the importance of individual objective function.
- Solve each subproblem to get 1 point in the Pareto front.
- In case a subproblem fails to be solved (the algorithm terminates without convergence) -> make a solution analysis, find a better solution (ex: gradient-based -> local minimum, but not global).

a. Weighted sum method

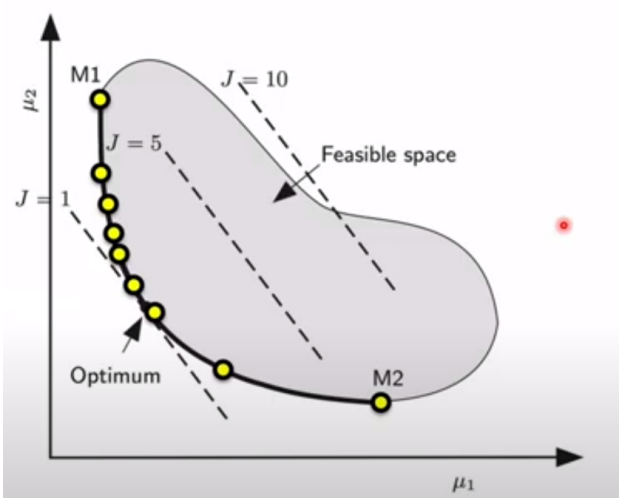
- A linear combination with weight to each objective function.
- Iso-cost lines: straight line with a particular slope; **optimum is the slope of the pareto front, this then becomes one point in the Pareto front**.

Example:
min J(x) = w1 * u_1(x) + w2 * u_2(x) + .. wn * u_n(x)
-> express in vector : w = [w1, w2, .. wn]^T

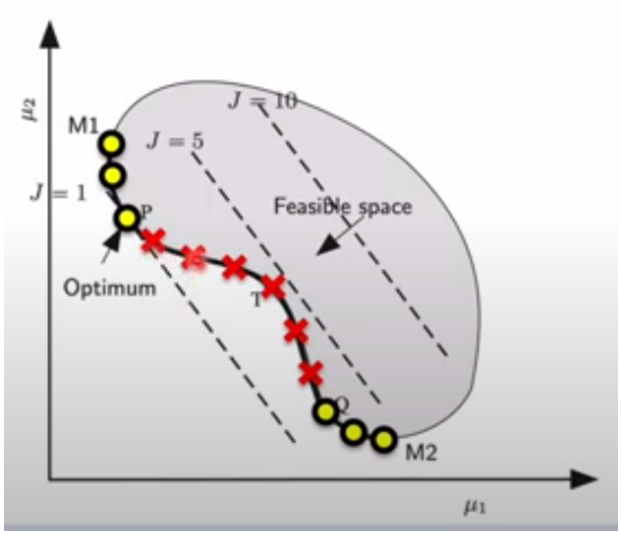
+ When wi = 0 (u_i is disregarded), iso-cost line is parallel to u_i -> anchor point (extreme point in the Pareto front)
+ Vary systematically wi, s.t 0 <= wi <= 1 , sum of wi = 1 (to find all the point in the Pareto front)

Drawbacks:

+ Unevenly-distributed set of non-dominated points, provided a big gap which is hard to fill in. (in the example, point M2 is very far away from the next point)



+ Incapable of detect non-convex region

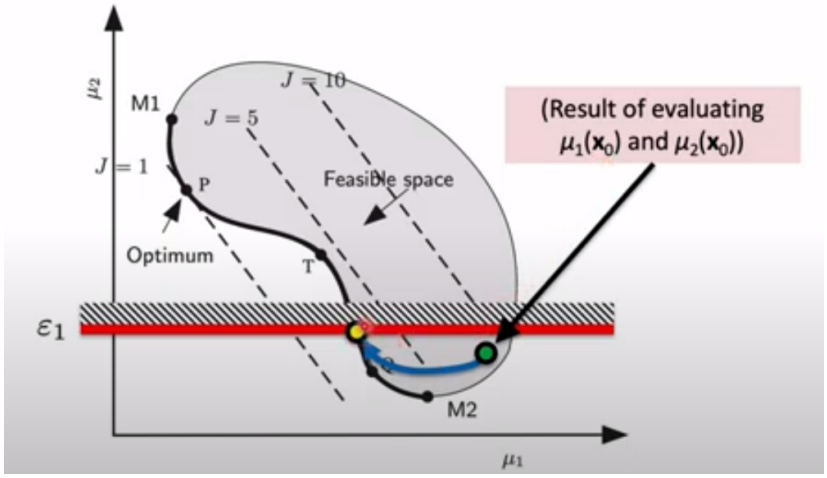


b. Epsilon-constraint method

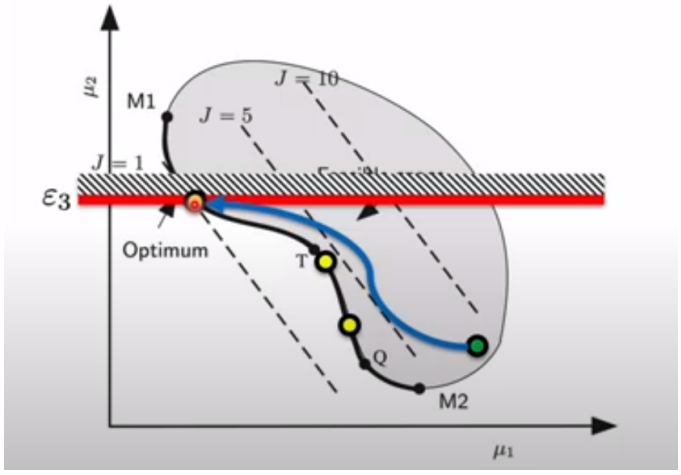
- convert all objectives, except for 1, into contraints with bounds.
- systematically vary bounds on constraints to generate Pareto frontier.
- not aggregate objective function.

Example:

- Problem: min {f1(x), f2(x)} subject to g(x) <= 0 and h(x) <=0
- Formulate transformation parameters: {P1, P2, .., Pn} => {ε1 < ε2 < .. < εn} (we want n points in the Pareto frontier)
- Replace upperbound with f2 -> reduce f1 but at a cost of f2
- Formulate and solve n subproblems:
  - ❖ min f1(x) subject to g(x) <= 0 and f2(x) - ε1 <= 0
  - ❖ min f1(x) subject to g(x) <= 0 and f2(x) - ε2 <= 0
    ..
  - ❖ min f1(x) subject to g(x) <= 0 and f2(x) - εn <= 0
- We change the epsilon values and find all the local minimums to formulate the Pareto frontier.



first problem with ε1



third problem with ε3

- How to find the epsilon range?

- ❖ We need to detect 2 ε values for 2 anchor points of the Pareto frontier (2 ends).
- ❖ One possible strategy could be first set the ε value very large (which is the anchor point) and reduce it until the anchor point changes.

Advantages:

- Able to detect non-convex region of pareto frontier.
- Can produce even distribution of the Pareto frontier.
- Most preferred/reliable solution at the moment.

III. Work on a simple problem with scalarization methods

There are 2 objective functions dependent on variables x, y. With constraints for x and y as follows:

$$\begin{gathered}\min f_1(x,y) = 2x^2 + y^2 \\ \min f_2(x,y) = (x-1)^2 + 2(y-1)^2 \\ s.t. \\ -2 \le x \le 2 \\ -2 \le y \le 2\end{gathered}$$

1. Weighted sum solution

Formulation process as follows:
+ We assign weights for both objective function, in which total weight sum is equal to 1 -> α for f1 and 1-α for f2).

+ If α differs from 0 to 1, go get various value pair for (f1, f2), with 2 extremes of f1=3, f2=0 and f1=0, f2=3.

+ A balanced value could be when α = 0.5, where f1=f2=0.67. However, in fact, the unit of f1 and f2 in reality could be totally different (ex: gallon, dollar, sales, percentage, ..), such that even f1=f2 could pose imbalance -> normalization process?

- Weighted Sum

$$\min \alpha f_1(x,y) + (1-\alpha) f_2(x,y)$$
$$= \alpha(2x^2 + y^2) + (1-\alpha)\left((x-1)^2 + 2(y-1)^2\right)$$
$$s.t.$$
$$-2 \le x \le 2$$
$$-2 \le y \le 2$$

α	x^*	y^*	f_1	f_2
0.00	1.00	1.00	3.00	0.00
0.10	0.82	0.95	2.24	0.04
0.20	0.67	0.89	1.68	0.14
0.30	0.54	0.82	1.26	0.28
0.40	0.43	0.75	0.93	0.45
0.50	0.33	0.67	0.67	0.67
0.60	0.25	0.57	0.45	0.93
0.70	0.18	0.46	0.28	1.26
0.80	0.11	0.33	0.14	1.68
0.90	0.05	0.18	0.04	2.24
1.00	0.00	0.00	0.00	3.00

2. E-constraint solution

Formulation process as follows:
+ We only consider f1 as the objective function as long as we could control the f2 (put f2 as a constraint, f2 is smaller than a threshold denoted as ε). In another word, we want to minimize f1 at the cost of f2.

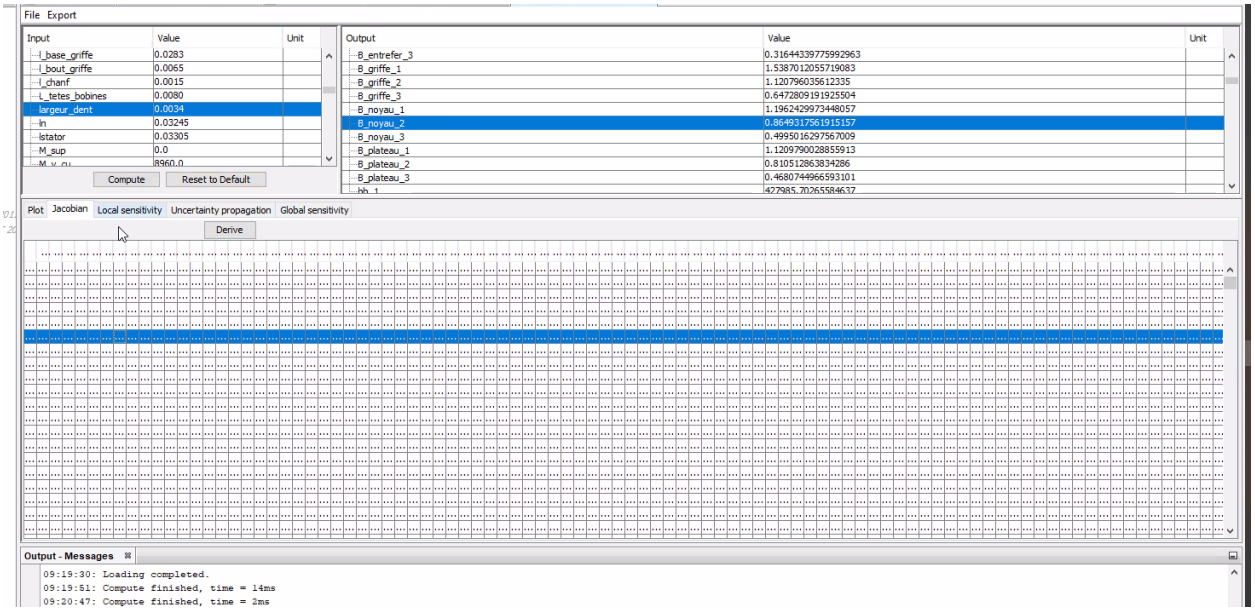
$$\min f_1(x,y)$$
$$s.t.$$
$$-2 \le x \le 2$$
$$-2 \le y \le 2$$
$$f_2(x,y) \le \epsilon$$

ϵ	x^*	y^*	f_1	f_2
0.00	1.00	1.00	3.00	0.00
0.50	0.40	0.73	0.86	0.50
1.00	0.23	0.55	0.41	1.00
1.50	0.14	0.39	0.19	1.50
2.00	0.07	0.24	0.07	2.00
2.50	0.03	0.12	0.02	2.50
3.00	0.00	0.00	0.00	3.00

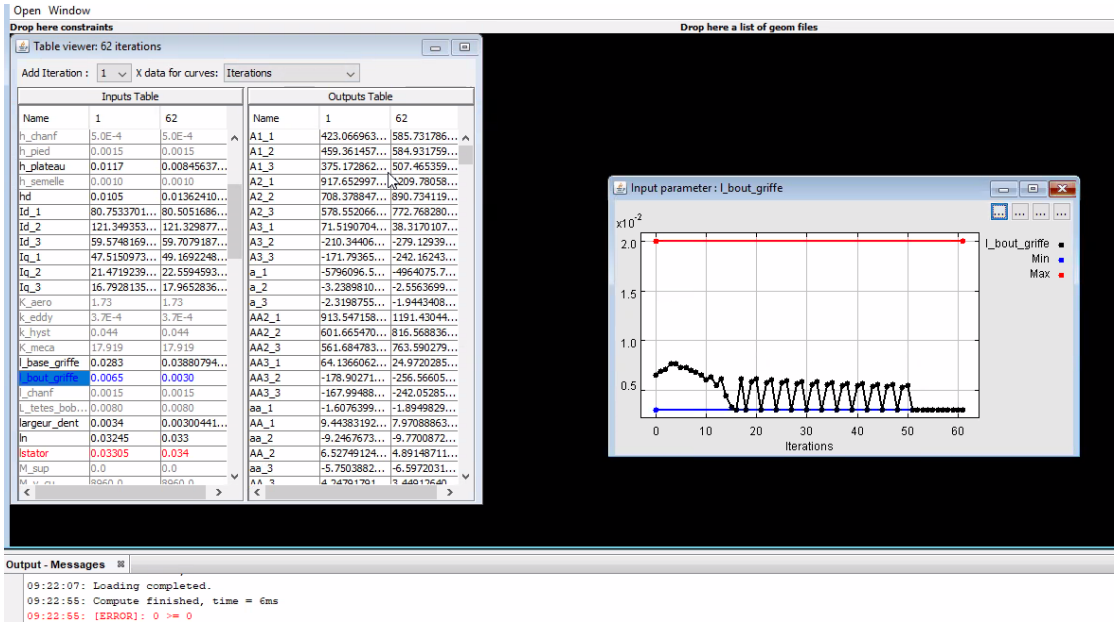
+ How to pick the range for ε? We could first minimize f2 -> lower bound for ε.
+ In the real world, there could be some expectation for the constraint of each objective function, we could use that info to determine ε.

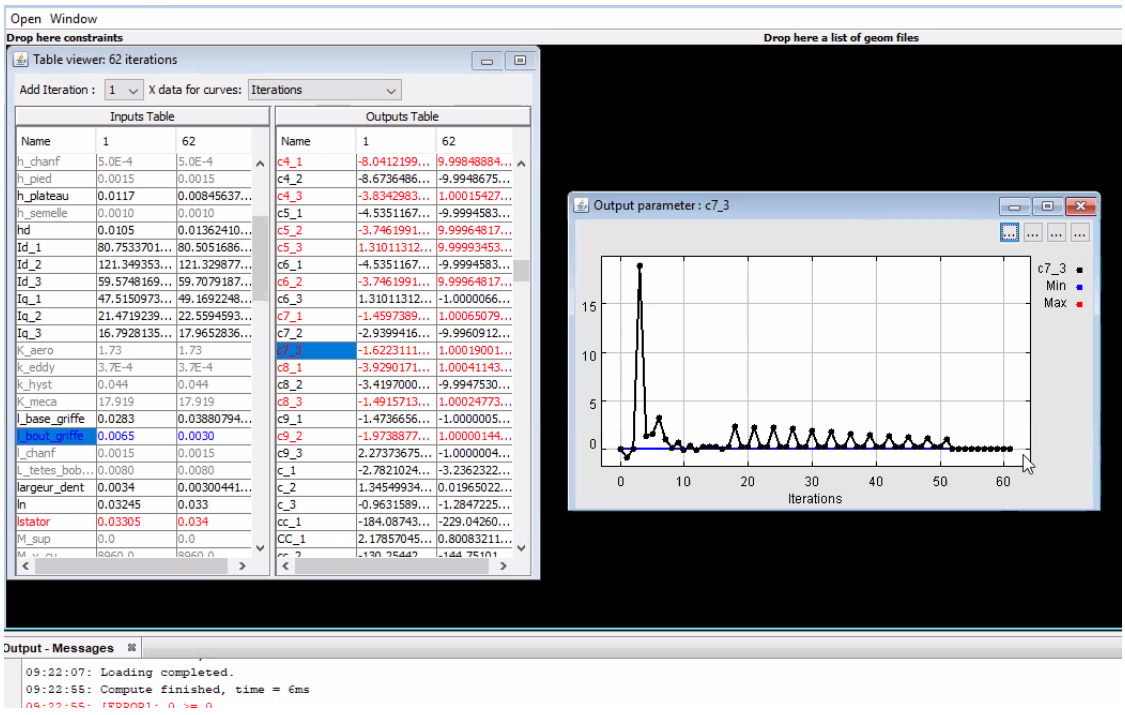
IV. Important points on Human Interface / Interaction

1. User is able to choose output-input values to plot (output according to input), with respect to Jacobian, local/global sensitivity, uncertainty propagation.

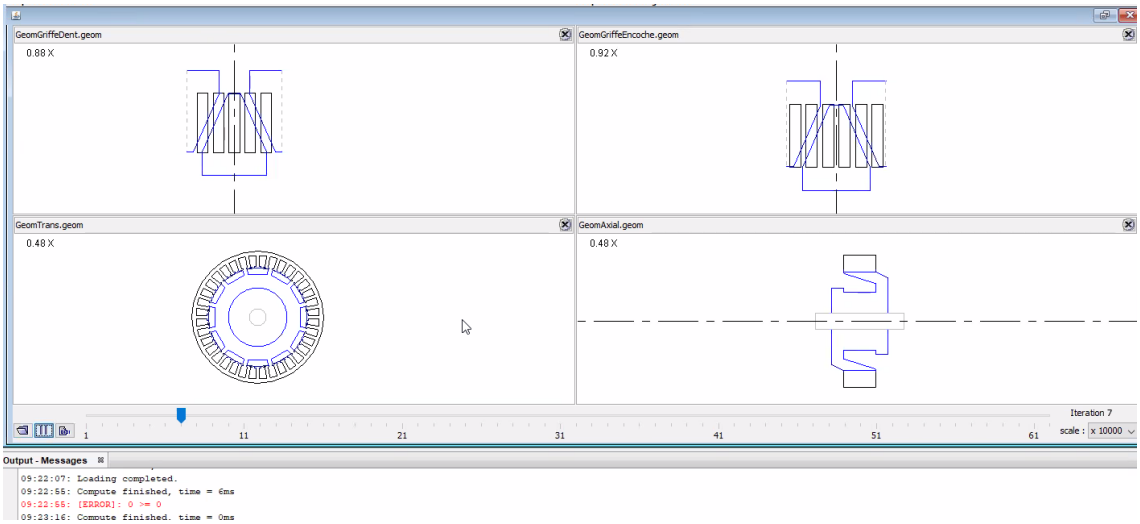


2. Display of high / low constraint of the inputs. Outputs of the optimization process are saved. All iterations are saved, allowing user to see the evolution/changes during the process.





3. Important for the designer, to exploit the evolution of input to plot the geometric data (separated tool - geomaker)



*Already implemented in NoLOAD, but required more effort (hand writing code from the user)?

4. Load files, save files automatically (constraints files, data, process evolution), particularly in .xml where the results of the current process could be the initialization of the next one.

```
<SPECIFICATIONS>
<INPUT name="Id_1"><CONSTRAINT><FIXED value="80.75337015594165"/></CONSTRAINT></INPUT>
<INPUT name="Id_2"><CONSTRAINT><FIXED value="121.3493530193408"/></CONSTRAINT></INPUT>
<INPUT name="Id_3"><CONSTRAINT><FIXED value="59.57481692719375"/></CONSTRAINT></INPUT>
<INPUT name="Iq_1"><CONSTRAINT><FIXED value="47.51509738035035"/></CONSTRAINT></INPUT>
<INPUT name="Iq_2"><CONSTRAINT><FIXED value="21.471923915061282"/></CONSTRAINT></INPUT>
<INPUT name="Iq_3"><CONSTRAINT><FIXED value="16.792813519613894"/></CONSTRAINT></INPUT>
<INPUT name="K_aero"><CONSTRAINT><FIXED value="1.73"/></CONSTRAINT></INPUT>
<INPUT name="K_meca"><CONSTRAINT><FIXED value="17.919"/></CONSTRAINT></INPUT>
<INPUT name="L_tetes_bobines"><CONSTRAINT><FIXED value="0.0080"/></CONSTRAINT></INPUT>
<INPUT name="M_sup"><CONSTRAINT><FIXED value="0.0"/></CONSTRAINT></INPUT>
<INPUT name="M_v_cu"><CONSTRAINT><FIXED value="8960.0"/></CONSTRAINT></INPUT>
<INPUT name="M_v_fer"><CONSTRAINT><FIXED value="7870.0"/></CONSTRAINT></INPUT>
<INPUT name="N_1"><CONSTRAINT><FIXED value="2000.0"/></CONSTRAINT></INPUT>
<INPUT name="N_2"><CONSTRAINT><FIXED value="6000.0"/></CONSTRAINT></INPUT>
<INPUT name="N_3"><CONSTRAINT><FIXED value="6000.0"/></CONSTRAINT></INPUT>
<INPUT name="N_cond_par_encoche"><CONSTRAINT><FIXED value="8.0"/></CONSTRAINT></INPUT>
<INPUT name="N_phases"><CONSTRAINT><FIXED value="3.0"/></CONSTRAINT></INPUT>
<INPUT name="Nepp"><CONSTRAINT><FIXED value="1.0"/></CONSTRAINT></INPUT>
<INPUT name="Nex"><CONSTRAINT><FIXED value="370.0"/></CONSTRAINT></INPUT>
<INPUT name="R_ext_rotor"><CONSTRAINT><FIXED value="0.05265"/></CONSTRAINT></INPUT>
<INPUT name="R_ext_stator"><CONSTRAINT><FIXED value="0.06775"/></CONSTRAINT></INPUT>
<INPUT name="R_int_stator"><CONSTRAINT><FIXED value="0.053"/></CONSTRAINT></INPUT>
<INPUT name="Ra"><CONSTRAINT><FIXED value="0.00865"/></CONSTRAINT></INPUT>
<INPUT name="Rd"><CONSTRAINT><FIXED value="0.0094"/></CONSTRAINT></INPUT>
<INPUT name="Rn"><CONSTRAINT><FIXED value="0.02965"/></CONSTRAINT></INPUT>
<INPUT name="S_fil_rotor"><CONSTRAINT><FIXED value="7.4E-7"/></CONSTRAINT></INPUT>
<INPUT name="S_fil_stator"><CONSTRAINT><FIXED value="1.54E-6"/></CONSTRAINT></INPUT>
<INPUT name="T_cu_rotor"><CONSTRAINT><FIXED value="130.0"/></CONSTRAINT></INPUT>
<INPUT name="T_cu_stator_1"><CONSTRAINT><FIXED value="120.0"/></CONSTRAINT></INPUT>
<INPUT name="T_cu_stator_2"><CONSTRAINT><FIXED value="120.0"/></CONSTRAINT></INPUT>
<INPUT name="T_cu_stator_3"><CONSTRAINT><FIXED value="120.0"/></CONSTRAINT></INPUT>
<INPUT name="Ub"><CONSTRAINT><FIXED value="14.0"/></CONSTRAINT></INPUT>
<INPUT name="Ub_1"><CONSTRAINT><FIXED value="14.0"/></CONSTRAINT></INPUT>
<INPUT name="Ub_2"><CONSTRAINT><FIXED value="14.0"/></CONSTRAINT></INPUT>
<INPUT name="Ub_3"><CONSTRAINT><FIXED value="14.0"/></CONSTRAINT></INPUT>
<INPUT name="Vbb"><CONSTRAINT><FIXED value="0.75"/></CONSTRAINT></INPUT>
<INPUT name="Vd"><CONSTRAINT><FIXED value="0.75"/></CONSTRAINT></INPUT>
<INPUT name="Vr"><CONSTRAINT><FIXED value="0.8"/></CONSTRAINT></INPUT>
<INPUT name="coef_rend_1"><CONSTRAINT><FIXED value="0.0"/></CONSTRAINT></INPUT>
<INPUT name="coef_rend_2"><CONSTRAINT><FIXED value="0.0"/></CONSTRAINT></INPUT>
<INPUT name="coef_rend_3"><CONSTRAINT><FIXED value="-1.0"/></CONSTRAINT></INPUT>
```

Suggestions for the starter:

- Firstly, try to play with function in Noload. Graphically plot the variable, input-output, analysis of sensibility, analysis the result of optimization Pareto front, evolution of all variables.
- Then the gradients, the results.
- Finally, the exploitation of the results.