ANATOLE

Tolérancement 3D et analyse de sensibilité avec OpenTURNS

D. Barbier[*], *B. Chaigne*[*], *J. Fourcade*[†], *R. Lebrun*[o]

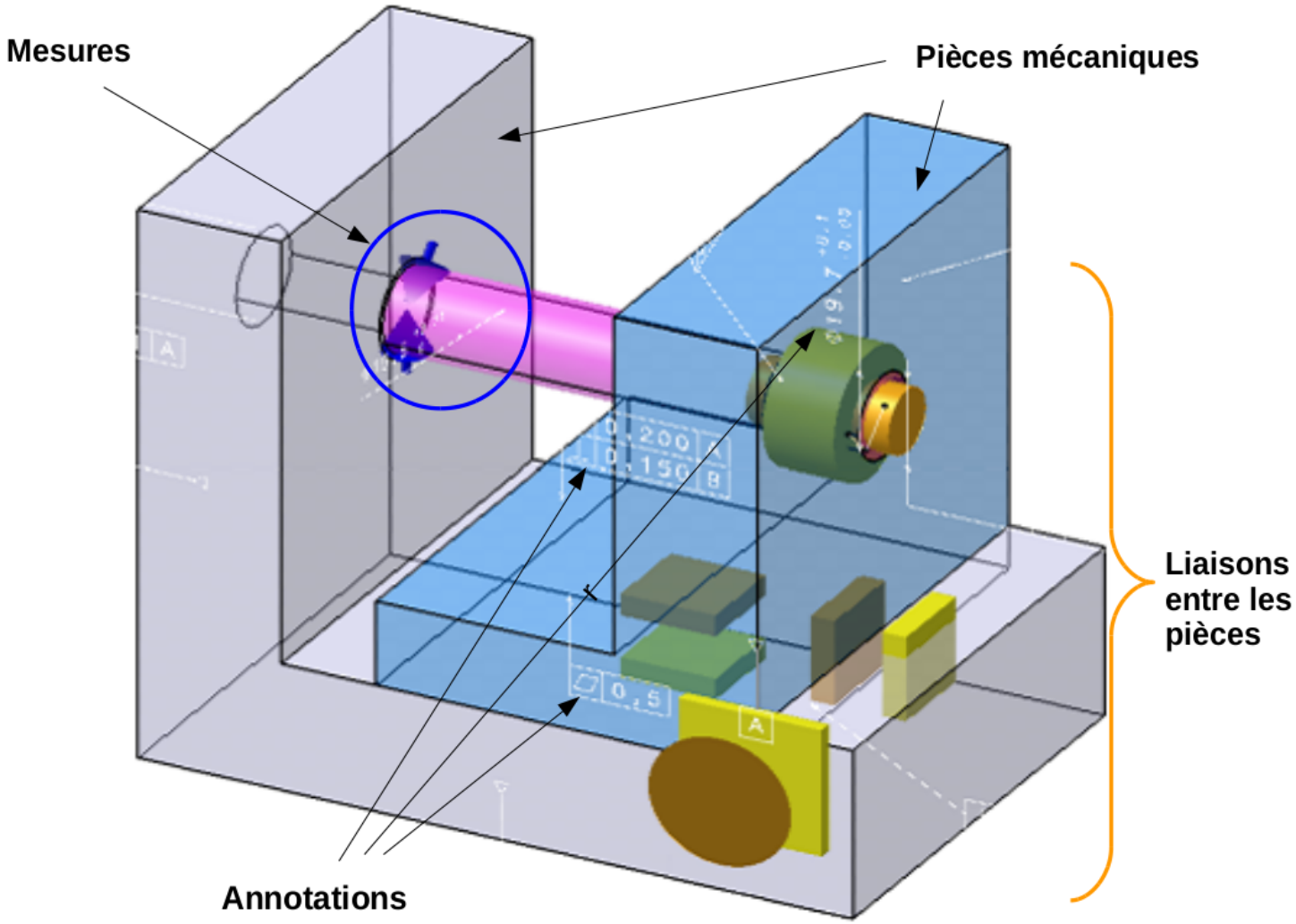
[*]IMACS, [†]DPS, [o]Airbus

Contexte

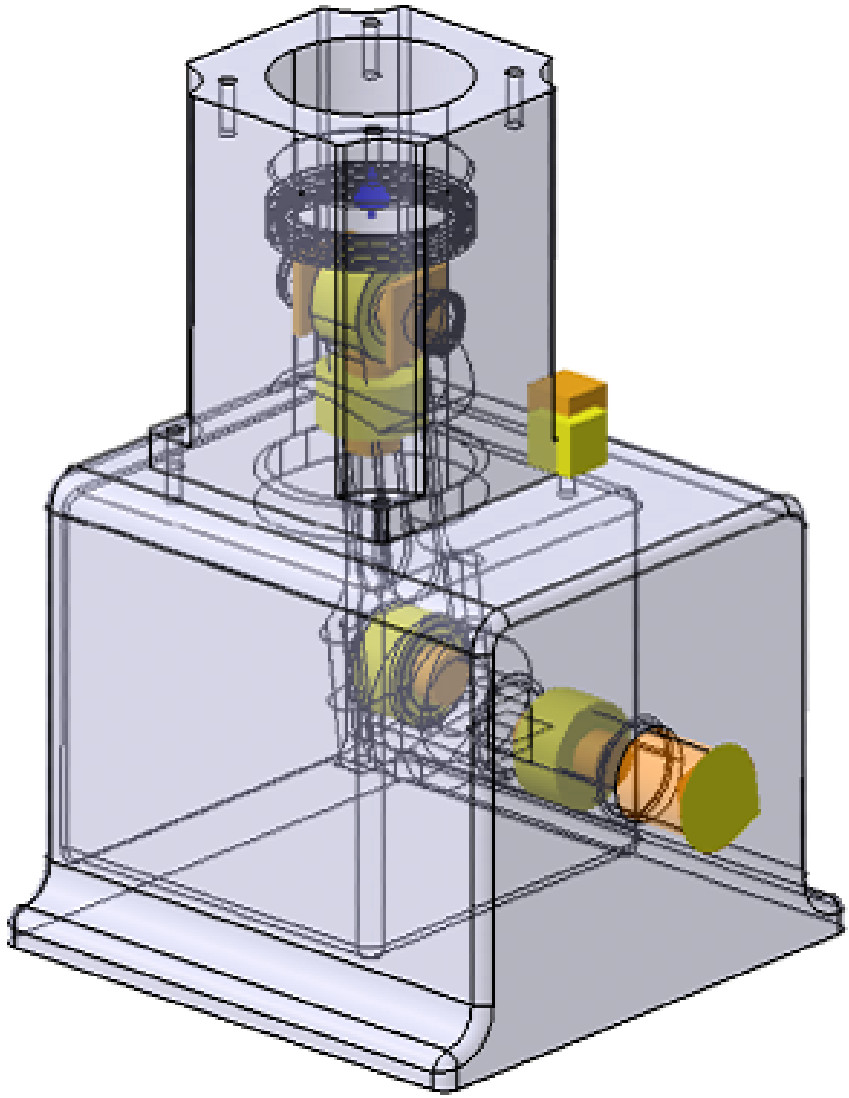
Anatole : Analyse de tolérancement 3D (Airbus)

- Composant CATIA (Airbus/DPS)
- **ToleranceEngine** (Airbus/DPS/IMACS)
- Solveur basé sur OpenTURNS (Airbus/EDF/IMACS/PhiMeca)

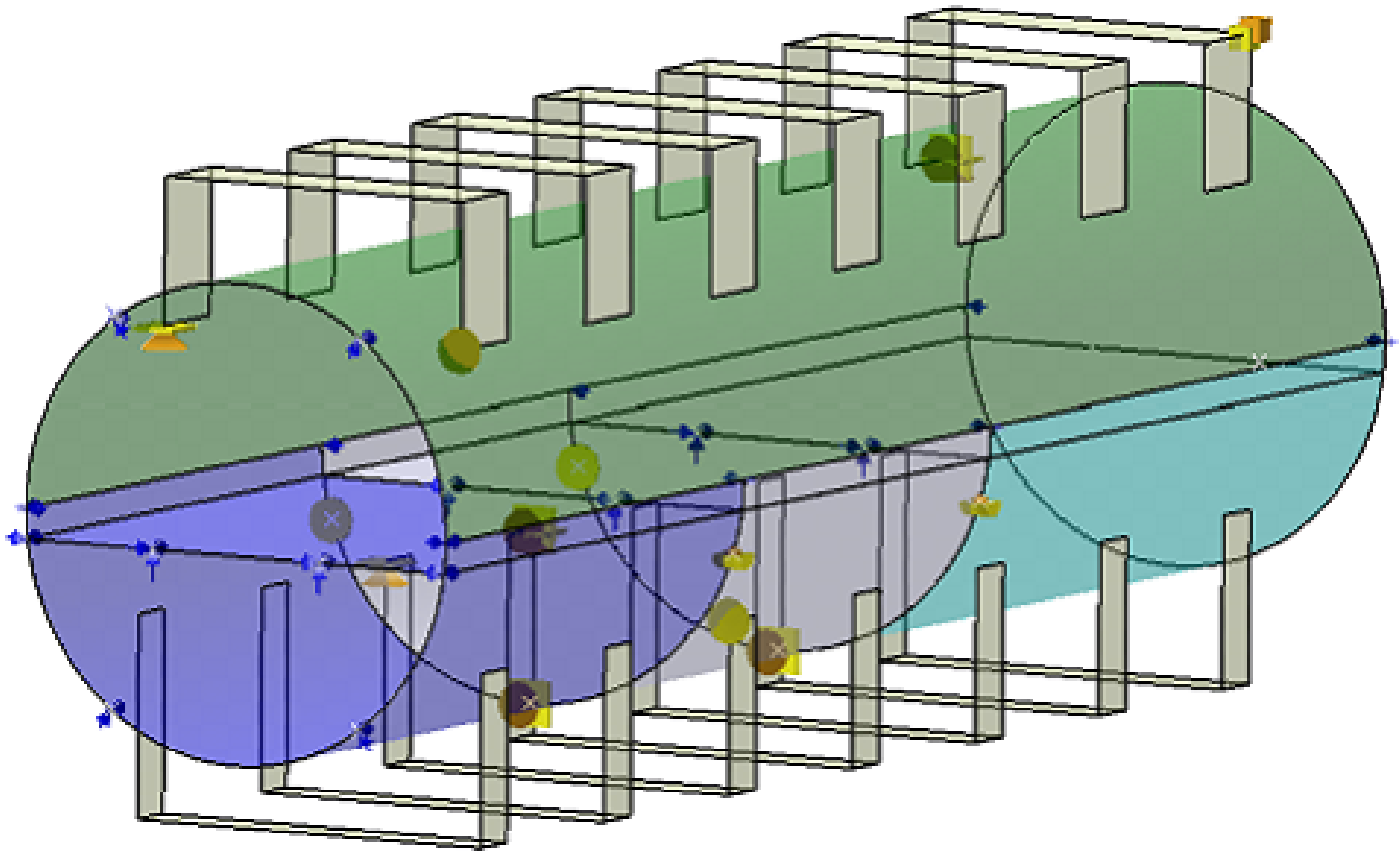
Exemple - tutoriel



Exemple - piston



Exemple - fuselage



Objectifs

1. Calcul des mesures en fonction
   - des dimensions des pièces
   - des degrés de liberté entre les pièces
   - des annotations

2. Calcul de sensibilité des mesures
   - aux annotations : sensibilité déterministe
   - à la variabilité des torseurs de liaison : sensibilité stochastique

Analyse des cas pires

Problème d'optimisation quadratique

- Objectif linéaire
- Contraintes d'égalité linéaires
- Contraintes d'inégalité linéaires et quadratiques
- Gradient "très" creux
- τ : annotations, paramètres par rapport auxquels on souhaite calculer la sensibilité du résultat

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad J(x) &= c \cdot x \\ h(x) &\geq b(\tau) \end{aligned}$$

Sensibilité déterministe

Lagrangien

$$\mathcal{L}(x;\lambda) = J(x) + (h(x) - b(\tau)) \cdot \lambda$$

Sensibilité

$$\nabla_\tau J(x^*) = -\nabla_\tau b(\tau)\lambda^*$$

Calculs stochastiques

- On considère les variables des torseurs de liaisons X_k comme des variables aléatoires (RandomMixture)

$$J = \sum_k c_k X_k$$

- On cherche des informations complémentaires à l'analyse pire cas : *e.g.* loi de la mesure J
- Calculs (exacts) basés sur la formule sommatoire de Poisson
- Calculs de sensibilité en terme de variance : $s_k := \frac{c_k^2 \mathrm{Var}(X_k)}{\mathrm{Var}(Y)}$

Travaux 2015-2017

Historiquement

- TE : Optimisation indépendante d'OpenTURNS
- TE : Calcul stochastique soumis à un exécutable externe : **DriverOT** (2009-2011)
- OT : Portage Windows librairie C++ (2014)

Depuis 2015 : travaux autour des solveurs d'optimisation

- Refonte de l'optimisation dans OpenTURNS (disponible depuis la version 1.7)
- Ajout du calcul de sensibilité déterministe
- Ouvre le remplacement du **DriverOT** à une implémentation directe à la librairie

Mise en oeuvre

- Utilisation de l'API C++
- Windows (module CATIA), compilation ot-superbuild (Visual 2005/2008/2010)
- Constructions des fonctions analytiques à partir de la description des fonctions dans le **ToleranceEngine**
  `OT::SymbolicFunction`
- Construction des problèmes d'optimisation
  `OT::OptimizationProblem`
- Solveur SLSQP
  `OT::NLopt nlopt("LD_SLSQP");`
- Calcul des multiplicateurs de Lagrange
  `OT::OptimizationResult::getLagrangeMultipliers()`
- Calcul de sensibilité : produit matrice-vecteur "à la main" prenant en compte la structure creuse de $\nabla_\tau b(\tau)$

Exemple de fonctions symboliques

- Contraintes liées à la nature de la liaison (pivot glissant)

```
Ctx_S26_S70_pivot_glissant = Ctx_S26_S70_s1+Ctx_S26_S70_s2+Ctx_S26_S70_J;
Cty_S26_S70_pivot_glissant = Cty_S26_S70_s1+Cty_S26_S70_s2+Cty_S26_S70_J;
Crx_S26_S70_pivot_glissant = Crx_S26_S70_s1+Crx_S26_S70_s2+Crx_S26_S70_J;
```

- Bornes liées à une annotation

```
Ctx_S26_S70_s1 >= -0.1; // Rotor_Loc_0.2_K/Val:0,2:-0.5
Ctx_S26_S70_s1 <= 0.1; // Rotor_Loc_0.2_K/Val:0,2:0.5
```

- Contrainte sans annotation liée à la géométrie/liaison

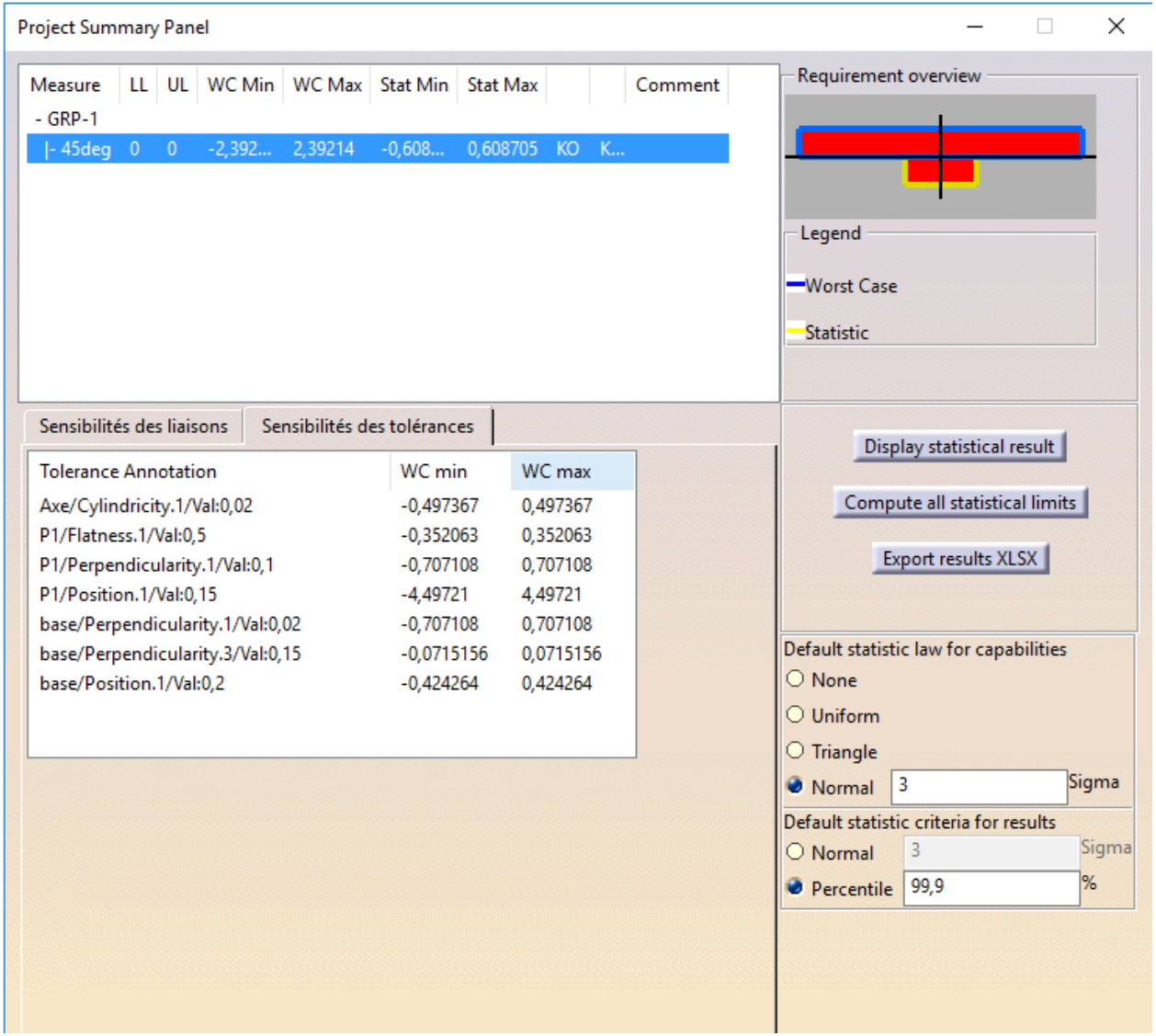
```
SQR(Ctx_S26_P10S70_Jeu)+SQR(ABS(Cty_S26_P10S70_Jeu)+8.5*ABS(Crx_S26_P10S70_Jeu)) < 0.01;
```

- Contraintes avec annotation liées à la géométrie/liaison

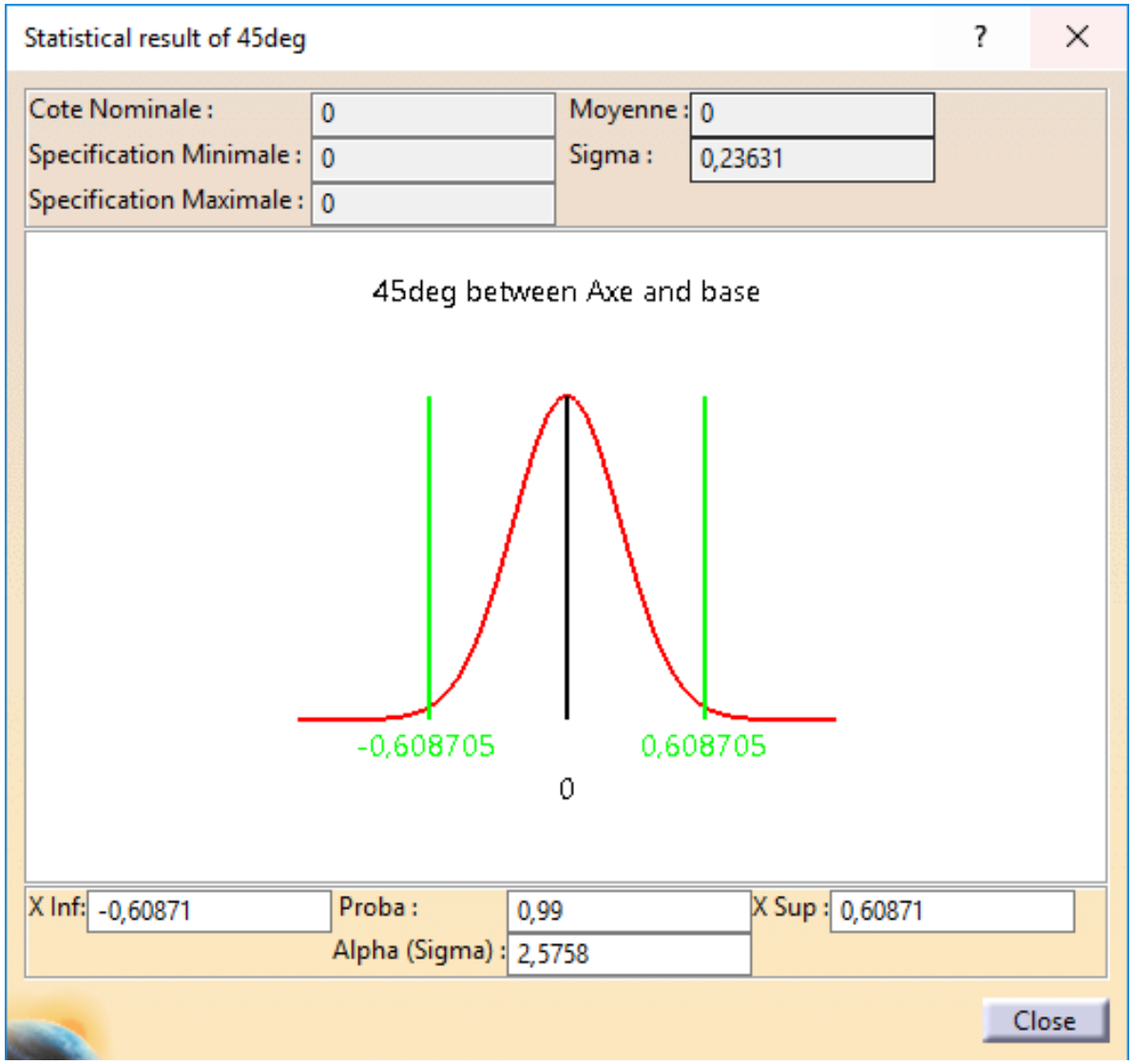
```
SQR(Ctx_S26_S70_s1)+SQR(ABS(Cty_S26_S70_s1)+8.6525*ABS(Crx_S26_S70_s1)) < 0.01;
// Rotor_Loc_0.2_K/Val:0,2:0.1
SQR(Ctx_S26_S70_s1)+SQR(ABS(Cty_S26_S70_s1)+8.6525*ABS(Crx_S26_S70_s1)) < 0.0025;
// Rotor_Loc_0.1_X/Val:0,1:0.05
```

NB : pré-traitement pour gérer les contraintes inutiles avant soumission du problème à OpenTURNS

Quelques résultats



Quelques résultats



Quelques résultats

J2_Lineaire_GRP_1_2		Min	Max
	status	0	0
	value	-0.300000	0.300000
F6137-A36798.1/Localisation.5/Val:0,3		-0.500008	0.500008
F6137-A38187.1/Localisation.1/Val:0,1		-0.499999	0.499999

J4_Lineaire_GRP_1_4		Min	Max
	status	0	0
	value	-9.400457	9.400456
F6137-A36798.1/Localisation.5/Val:0,3		-17.443008	17.443263
F6137-A38187.1/Localisation.1/Val:0,1		-12.196269	12.196455

J3_Lineaire_GRP_1_3		Min	Max
	status	0	0
	value	0.000000	0.000000
F6137-A36798.1/Localisation.5/Val:0,3		0.000000	0.000000
F6137-A38187.1/Localisation.1/Val:0,1		0.000000	0.000000

Comparaison NLopt vs scipy

Comparaison des algorithmes SLSQP sur des cas Anatole

- Sauvegarde des fonctions à l'aide du mécanisme de persistence `OT::Study::save()`
- Relecture de ces fonctions dans un script python
- Wrapper des fonctions et de leurs gradients pour scipy

Deux cas :

1. 292 variables, 182 contraintes d'égalité, 132 contraintes d'inégalité
2. 60 variables, 39 contraintes d'égalité, 34 contraintes d'inégalité

Comparaison NLopt vs scipy

- Tolérance de convergence : 1e-05
- Tolérance sur les contraintes d'égalité : 1e-05

Cas 1	**NLopt**		**scipy**	
Mesure	J^*	time (s)	J^*	time (s)
J1	-0.302	40.83	-0.302	1.35
J2	-0.300	39.31	-0.300	1.37
J3	-9.409	191.94	-9.411	28.46
J4	-9.400	123.77	-9.400	4.66

Comparaison NLopt vs scipy

- Tolérance de convergence : 1e-05
- Tolérance sur les contraintes d'égalité : 1e-05

Cas 2	**NLopt**		**scipy**	
Mesure	J^*	time (s)	J^*	time (s)
J1	0.000	0.14	-2.230	0.03
J2	-2.415	0.11	-2.415	0.02
J3	-2.392	0.46	-2.392	0.29

Comparaison NLopt vs scipy

- Tolérance de convergence : 1e-03
- Tolérance sur les contraintes d'égalité : 1e-03

Cas 2	**NLopt**		**scipy**	
Mesure	J^*	time (s)	J^*	time (s)
J1	0.000	0.09	-2.231	0.03
J2	-2.419	0.11	-2.415	0.03
J3	-2.405	0.38	-2.405	0.27

Comparaison NLopt vs scipy

- Tolérance de convergence : 1e-05
- Tolérance sur les contraintes d'égalité : 1e-03
- NB : pour scipy ces deux critères sont liés

Cas 2	**NLopt**		**scipy**	
Mesure	J^*	time (s)	J^*	time (s)
J1	-2.233	0.14	x	x
J2	-2.419	0.11	x	x
J3	-2.405	0.46	x	x

Taille cas industriel

Taille définie dans le module CATIA :

- 40 pièces
- 100 liaisons
- 60 annotations

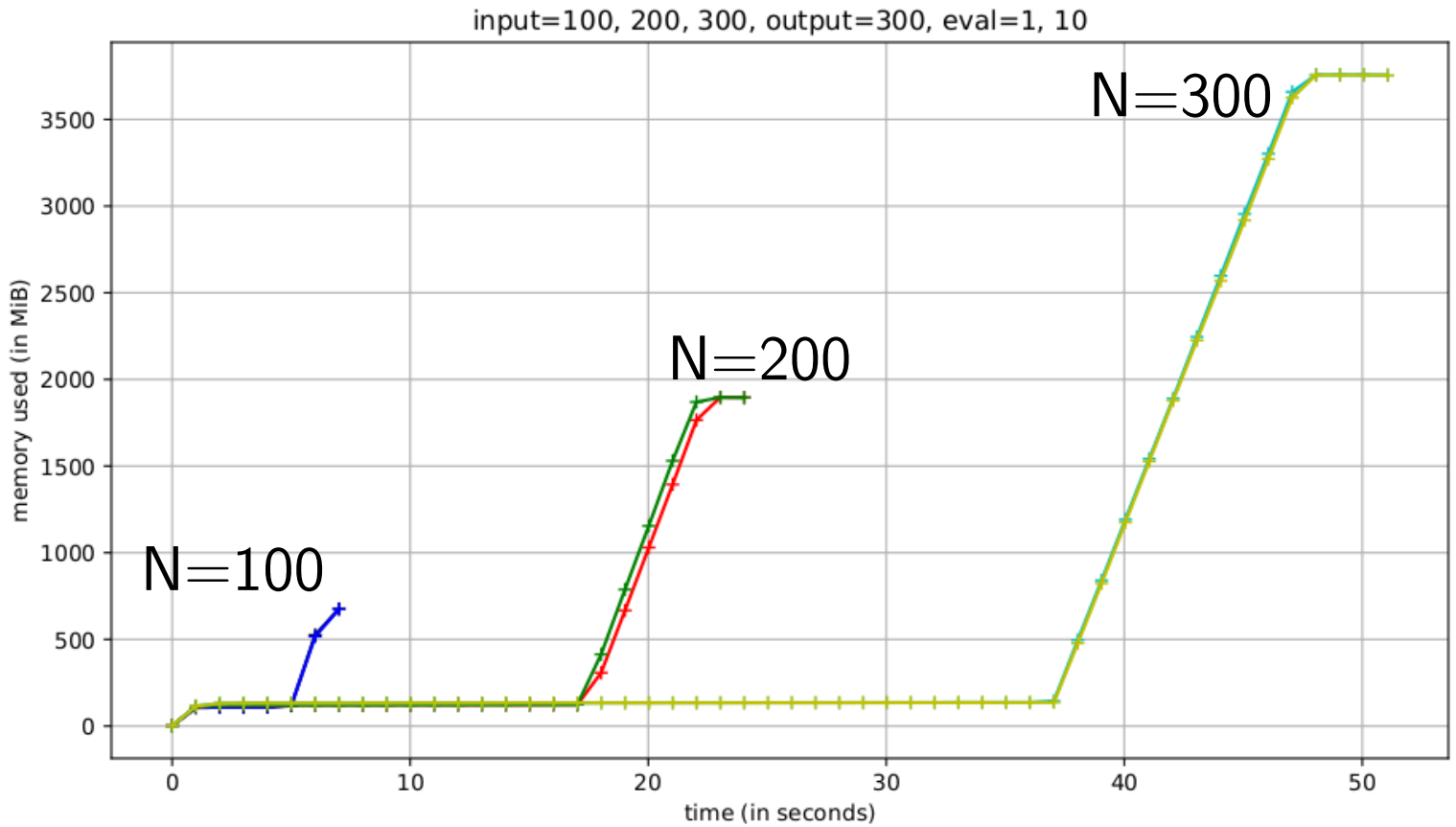
Taille obtenue dans le problème d'optimisation :

- 200 variables
- 300 contraintes

$\rightarrow$ Cela conduit à 300x200=60000 objets pour la construction des dérivées symboliques et autant pour leur évaluation ; gourmand en mémoire : limitant pour passer à de plus gros cas.

Illustration du problème mémoire

Construction et évaluation d'une fonction symbolique (linéaire) et de son gradient. Input=N, Output=300.



Conclusion

- Refonte des solveurs d'optimisation du **ToleranceEngine**
- Ajout du calcul de sensibilité déterministe
- Implémentation des solveurs via l'API C++ d'OpenTURNS (plus par des exécutables/fichiers)
- Bons résultats, validés par un ensemble de tests qui reprend les différents types de liaisons et surfaces de manière unitaire
- Application à des cas industriels

Perspectives

- Problème mémoire, pistes :
  - ExprTk : semble consommer plus de mémoire
  - Différence finies
  - Créer des fonctions adaptées aux besoins d'Anatole
  - Utiliser des LinearFunction/QuadraticFunction
- Remplacement de **DriverOT** pour les calculs stochastiques
- Résoudre les problèmes de robustesse du solveur d'optimisation liée à la tolérance sur les contraintes d'égalité
- Passage à de plus gros cas