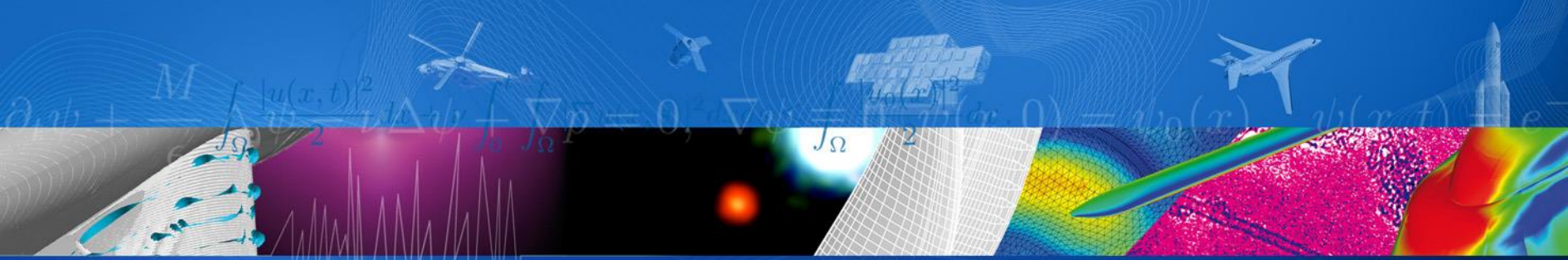


Preliminary study on launcher design with OpenMDAO

Loïc Brevault, Mathieu Balesdent

First European OpenMDAO workshop

Toulouse, 12 - 13 October 2017

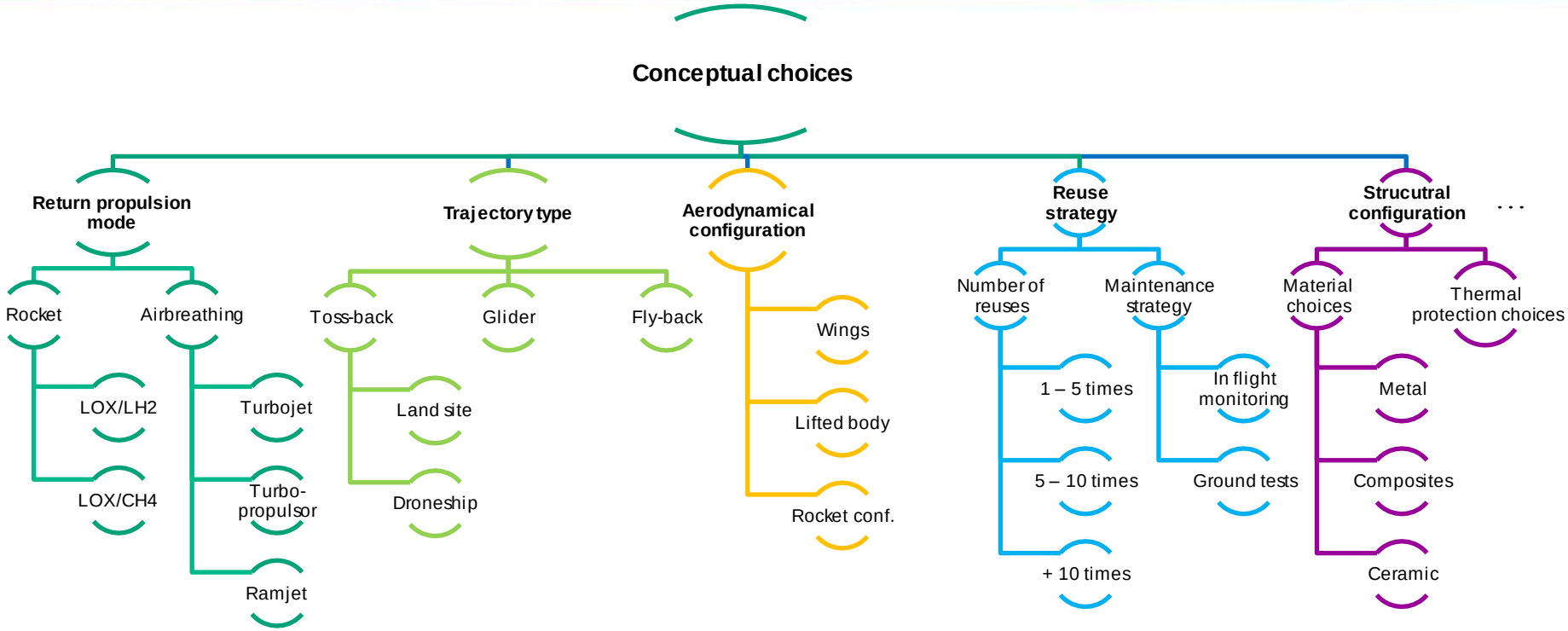
ONERA
THE FRENCH AEROSPACE LAB

retour sur innovation

Outline

1. Overview of Reusable Launch Vehicle (RLV) design
2. Disciplines involved in RLV design process
3. Design process mixing OpenMDAO and ONERA/Opensource methodological libraries:
    1. Step 1: workflow definition
    2. Step 2: workflow generation
    3. Step 3: MDAO studies
    4. Step 4: MDAO results & analysis
4. Remarks on the use of OpenMDAO
5. Conclusions and perspectives

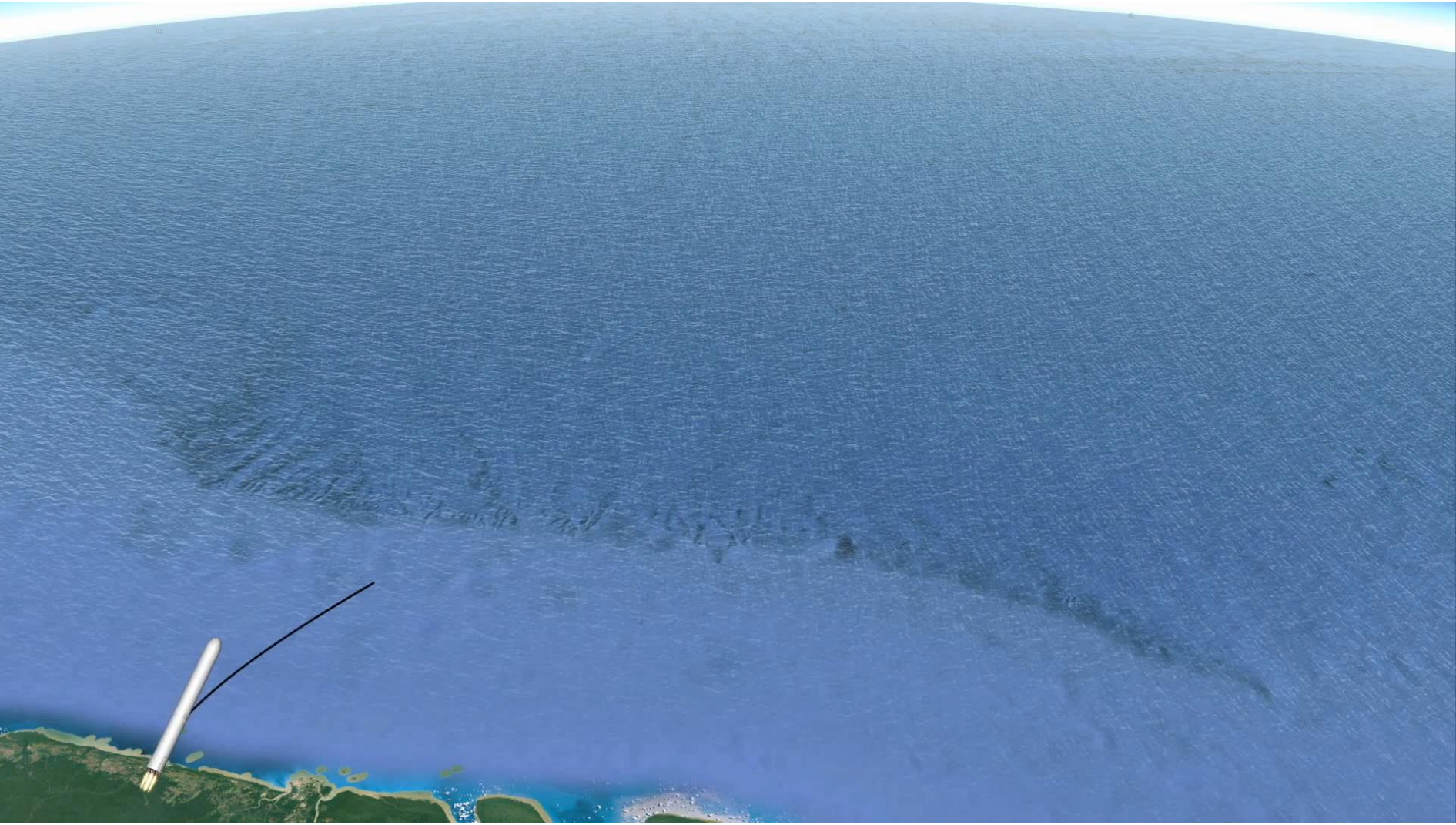
Diversity of Reusable Launch Vehicle concepts



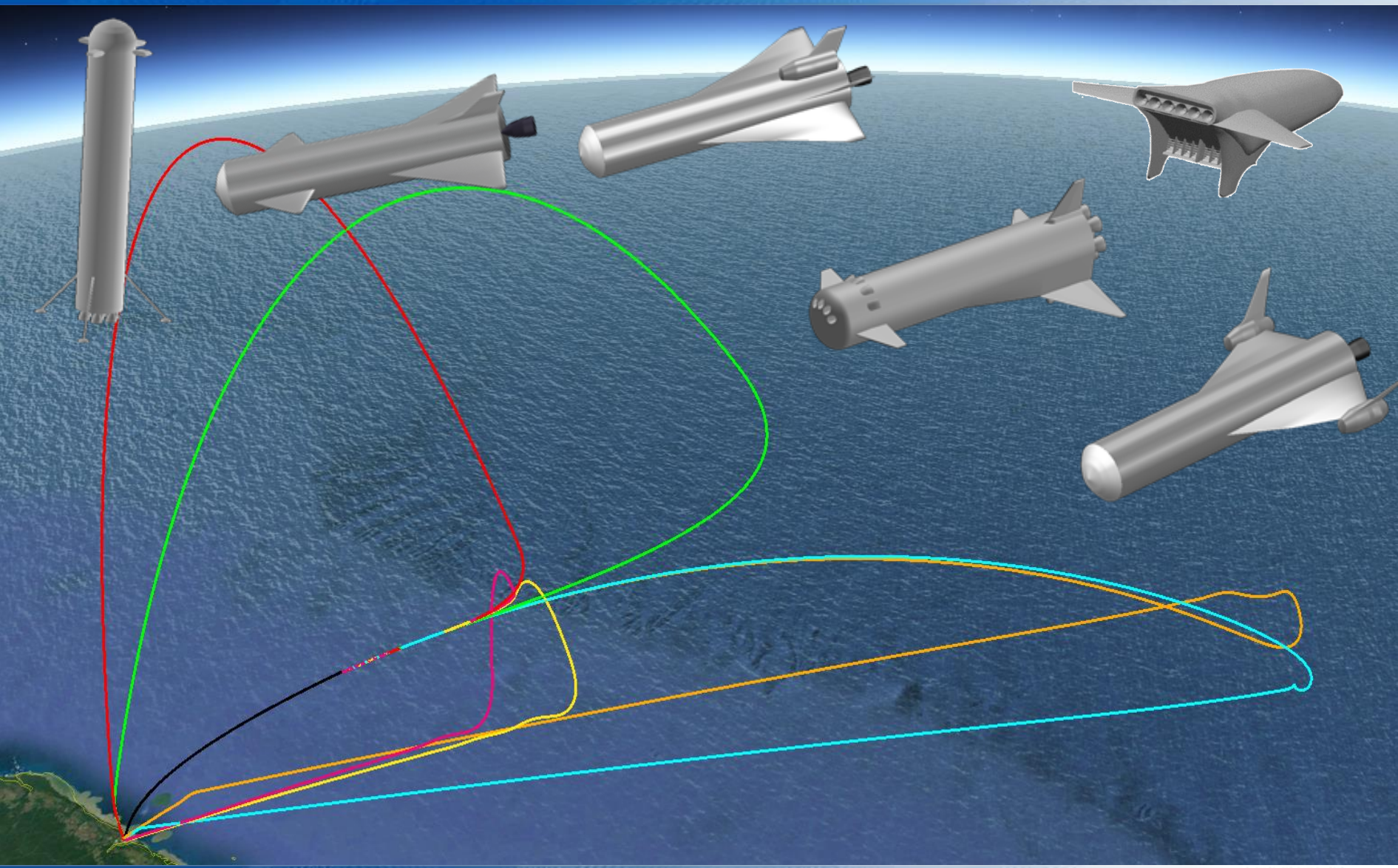
Modeling of complex physical phenomenon ⟷ Exploration of combinatorial « possible technical solutions »

ONERA HERACLES ONERA

Examples of 3 return trajectories for RLV

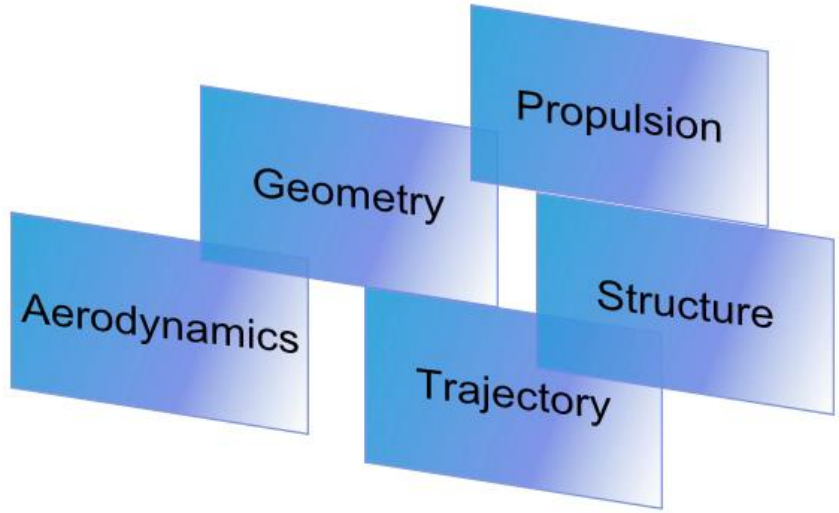


Diversity of RLV concepts



Classical disciplines involved in RLV design

Organization of the design process



How to integrate all the disciplines and to account for **interdisciplinary couplings** within launch vehicle analyses ?

Tools & codes used for RLV early design process

Propulsion

- Chemical Equilibrium with Applications
- Solid grain sizing and combustion

Trajectory

- 3DDL integrator
- GoogleEarth vizualization

Structure

- MER
- CalculiX

Geometry

- OpenVSP

Aerodynamics

- MISSILE DATCOM
- MISSILE (ONERA)
- SHABP « like »

Use of legagy codes, we do not have access to gradient

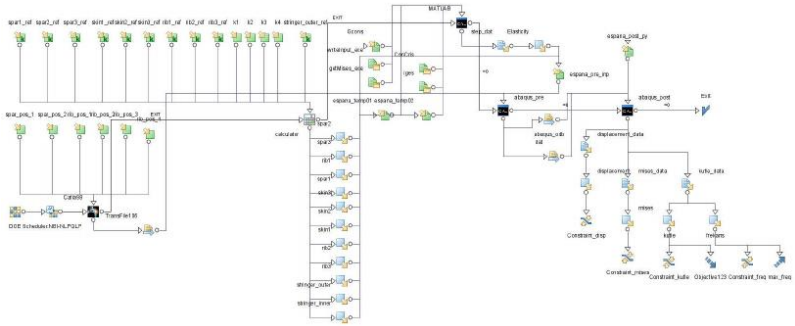
Multiple codes, languages :
executable, matlab, python, excel, *etc*.

Needs for an integrator tool

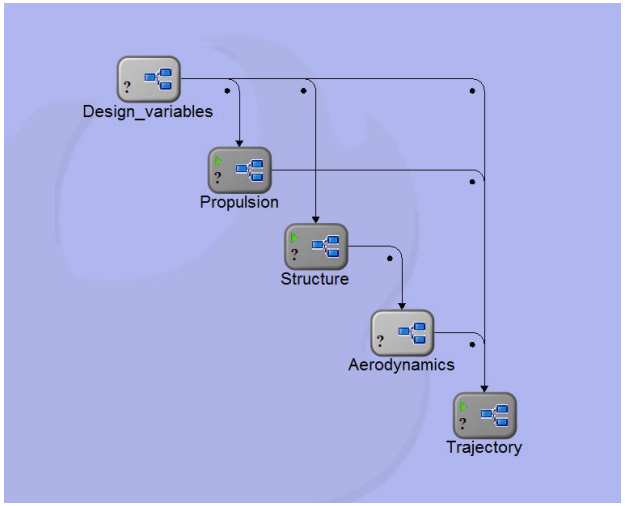
Launch vehicle design with commercial framework

Firstly ONERA used integrators framework such as:

- ModelCenter
- ModeFrontier



mode FRONTIER



PHOENIX® INTEGRATION

Issues:

- Difficulties to use in-house optimization algorithms and dedicated uncertainty methodologies
- Cumbersome workflow and links management

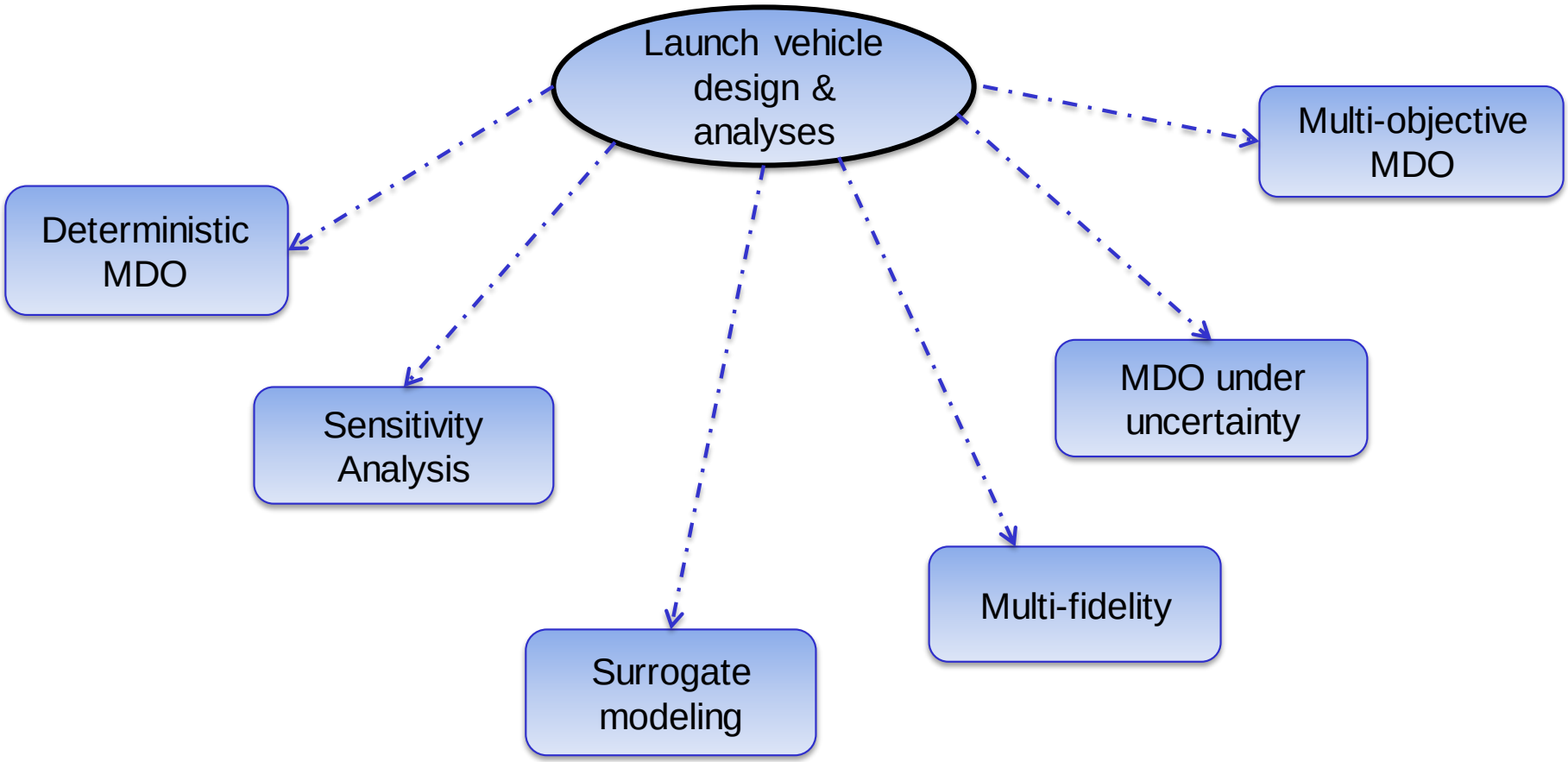
Advantages:

- Easy to use for non-expert
- Graphical interface

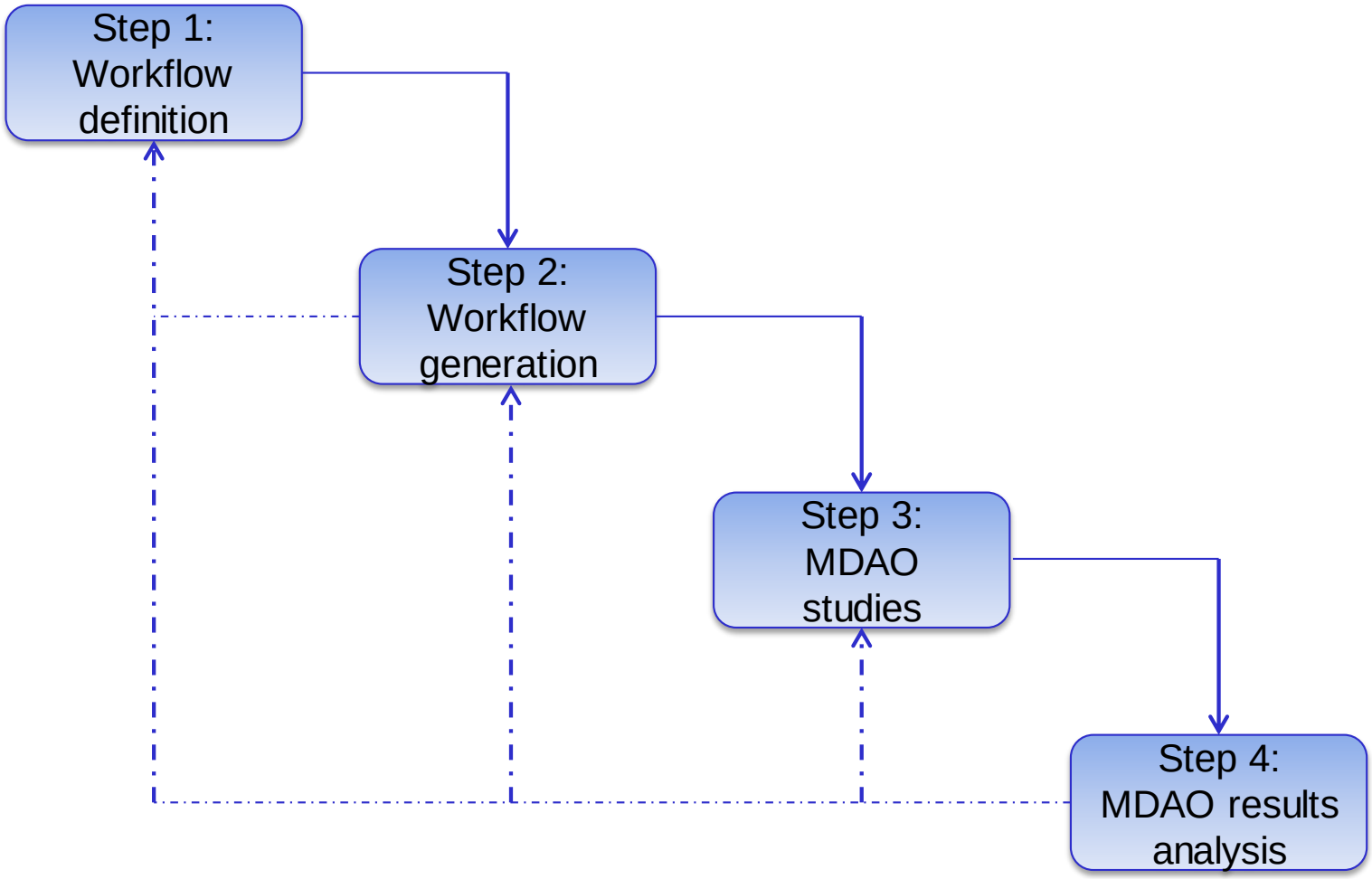
ONERA

Launch vehicle design and analyses

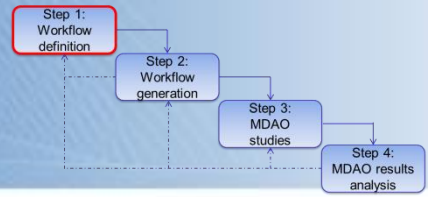
We need to be able to perform different types of analyses



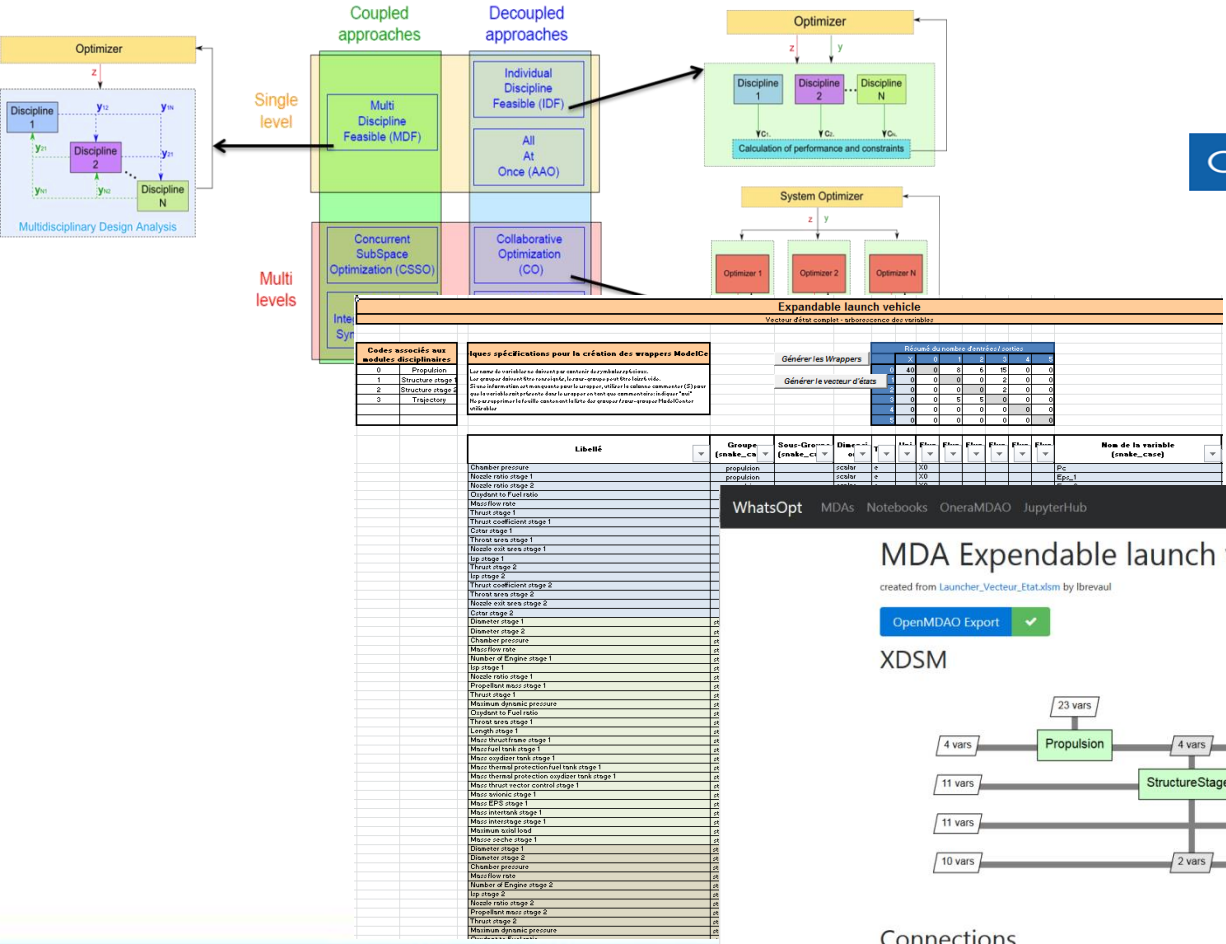
Four steps in the RLV design process

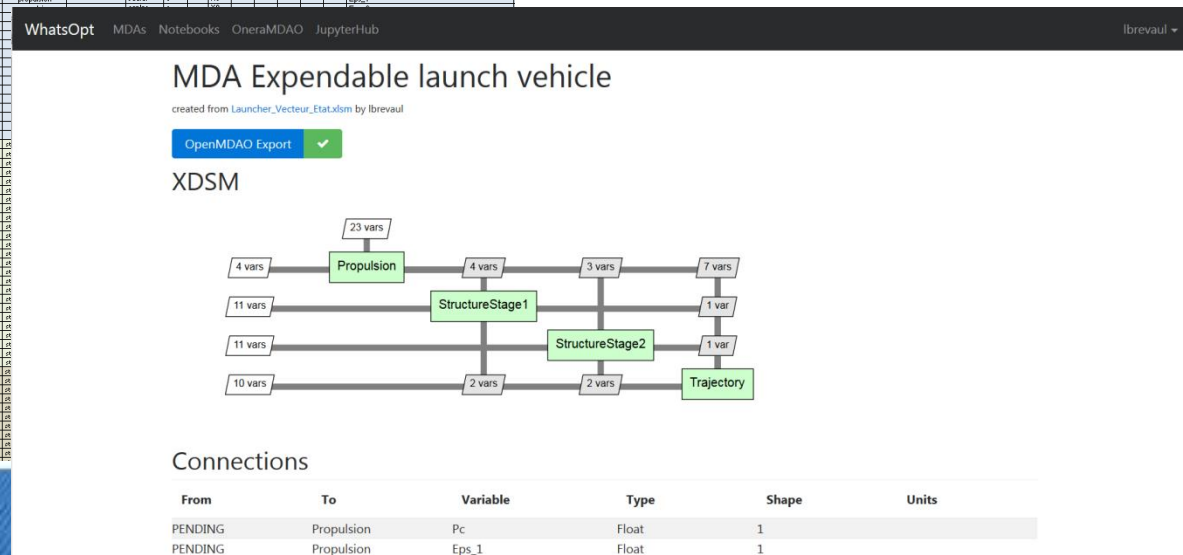


Step 1: workflow definition

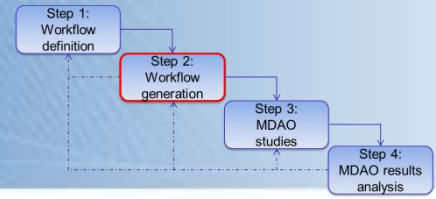


- Use of Excel template to define inputs / outputs of all the disciplines and the associated workflow
- Use of WhatsOpt (more details in R. Lafage presentation - Friday 13/10, 9h30)



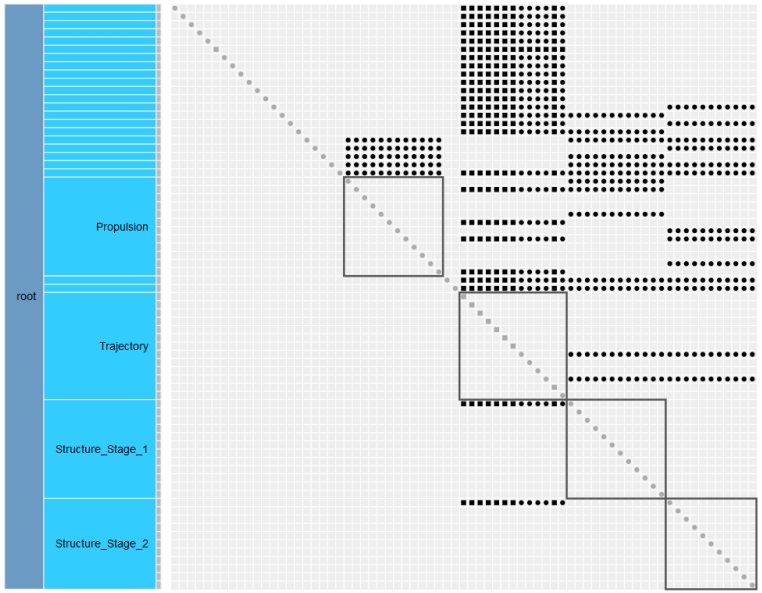


Step 2: worklow generation



ONERA ACADIA

- Use of OpenMDAO + WhatsOpt tools
  - Automatic generation of wrappers for OpenMDAO
  - Use of Group class for the Multidisciplinary Design Analysis
  - Use of Component class for Discipline
- Use of Partition_tree_n2 to visualize and interact



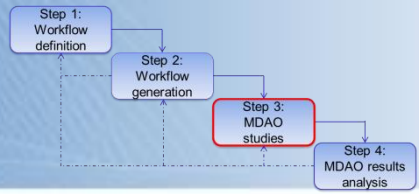

```
class MDA_Exp(Group):

    def __init__(self,fd=False):
        super(MDA_Exp, self).__init__()

        ##Design variables

        self.add('D_1', IndepVarComp('D_1', 5.4), promotes=['*'])
        self.add('D_2', IndepVarComp('D_2', 5.4), promotes=['*'])
        self.add('Debit_liq', IndepVarComp('Debit_liq', 5.4), promotes=['*'])
        self.add('Pc', IndepVarComp('Pc', 5.4), promotes=['*'])
        self.add('Eps_1', IndepVarComp('Eps_1', 5.4), promotes=['*'])
        self.add('Eps_2', IndepVarComp('Eps_2', 5.4), promotes=['*'])
        self.add('OF', IndepVarComp('OF', 5.4), promotes=['*'])
        self.add('N_eng_1', IndepVarComp('N_eng_1', 7.), promotes=['*'])
        self.add('N_eng_2', IndepVarComp('N_eng_2', 1.), promotes=['*'])
        self.add('Mprop_1', IndepVarComp('Mprop_1', 5.4), promotes=['*'])
        self.add('Mprop_2', IndepVarComp('Mprop_2', 5.4), promotes=['*'])
        self.add('thetacmd_i', IndepVarComp('thetacmd_i', 5.4), promotes=['*'])
        self.add('thetacmd_f', IndepVarComp('thetacmd_f', 5.4), promotes=['*'])
        self.add('ksi', IndepVarComp('ksi', 5.4), promotes=['*'])
        self.add('theta_final', IndepVarComp('theta_final', 5.4), promotes=['*'])
        self.add('Duree_basculement', IndepVarComp('Duree_basculement', 5.4), promotes=['*'])
        self.add('Delta_theta_basculement', IndepVarComp('Delta_theta_basculement', 5.4), promotes=['*'])
        self.add('commande_et1', IndepVarComp('commande_et1', np.zeros([4])), promotes=['*'])
        self.add('Duree_phase_verticale', IndepVarComp('Duree_phase_verticale', 5.4), promotes=['*'])
        self.add('Debut_balistique', IndepVarComp('Debut_balistique', 5.4), promotes=['*'])
        self.add('Duree_balistique', IndepVarComp('Duree_balistique', 5.4), promotes=['*'])
        self.add('Me_PAP', IndepVarComp('Me_PAP', 5.4), promotes=['*'])
        self.add('Nb_PAP', IndepVarComp('Nb_PAP', 5.4), promotes=['*'])
```

```
class Propulsion_Liquid_OpenMDAO(Component):
    def __init__(self):
        super(Propulsion_Liquid_OpenMDAO, self).__init__()
        ###Definition des entrees

        self.add_param('Pc', val = 100., type= 'float', desc = 'Chamber Pressure')
        self.add_param('Eps_1', val = 16.4, type= 'float', desc = 'Nozzle ratio')
        self.add_param('Eps_2', val = 20., type= 'float', desc = 'Nozzle ratio')
        self.add_param('OF', val = 2.7, type= 'float', desc = 'OF ratio')
        self.add_param('Debit_liq', val = 219., type= 'float', desc = 'Debit_Liq')

        ###Definition des constantes
        self.Constants = {}

        ###Definition des sorties

        self.add_output('Thrust_Liq_1',val=100000., type= 'array', desc = 'Poussee')
        self.add_output('Isp_Liq_1',val=330., type= 'array', desc = 'Isp')
        self.add_output('C_tau_1',val=1.7, type= 'array', desc = 'C_tau')
        self.add_output('C_star_1',val=1600., type= 'array', desc = 'C_star')
        self.add_output('A_t_1',val=0.05, type= 'array', desc = 'A_t')
        self.add_output('A_e_1',val=0.82, type= 'array', desc = 'A_e')

        self.add_output('Thrust_Liq_2',val=0., type= 'array', desc = 'Poussee')
        self.add_output('Isp_Liq_2',val=330., type= 'array', desc = 'Isp')
        self.add_output('C_tau_2',val=1.7, type= 'array', desc = 'C_tau')
        self.add_output('C_star_2',val=1600., type= 'array', desc = 'C_star')
        self.add_output('A_t_2',val=0.05, type= 'array', desc = 'A_t')
        self.add_output('A_e_2',val=1., type= 'array', desc = 'A_e')

    def solve_nonlinear(self, params, unknowns, resids):

        kr=pickle.load(open('save.p','rb'))
        kr_ctau=pickle.load(open('save_ctau.p','rb'))

        unknowns['Isp_Liq_1'] = kr.predict(np.array([params['OF'],params['Pc'],params['Eps_1']]))
        unknowns['C_tau_1'] = kr_ctau.predict(np.array([params['OF'],params['Pc'],params['Eps_1']]))
        unknowns['C_star_1'] = 9.80665*unknowns['Isp_Liq_1']/unknowns['C_tau_1']
```

→ Interdisciplinary coupling easily handled

ONERA

Step 3: MDAO studies



openMDAO

Launch vehicle design & analyses

Deterministic MDO

pyOpt

DEAP

Multi-objective MDO

Sensitivity Analysis

OpenTURNS

MDO under uncertainty

OpenTURNS

Multi-fidelity

Surrogate modeling

pyKriging

scikit learn

PyBrain

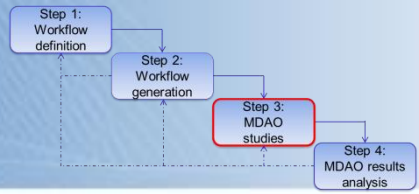
SMT: Surrogate Modeling Toolbox

+ in-house developed libraries

ONERA ACADIA

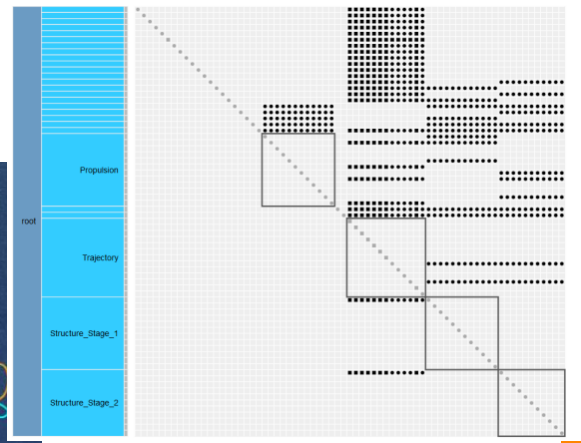
ONERA

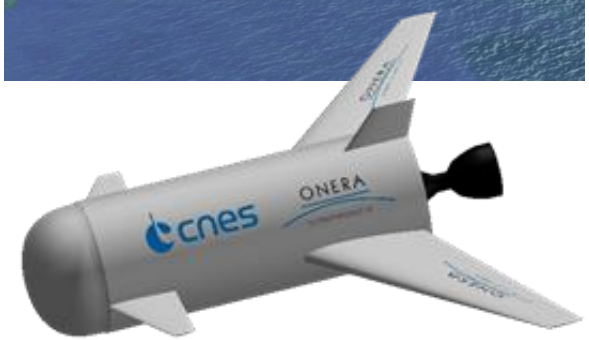
Step 3: Deterministic design of reusable 1st winged stage launch vehicle

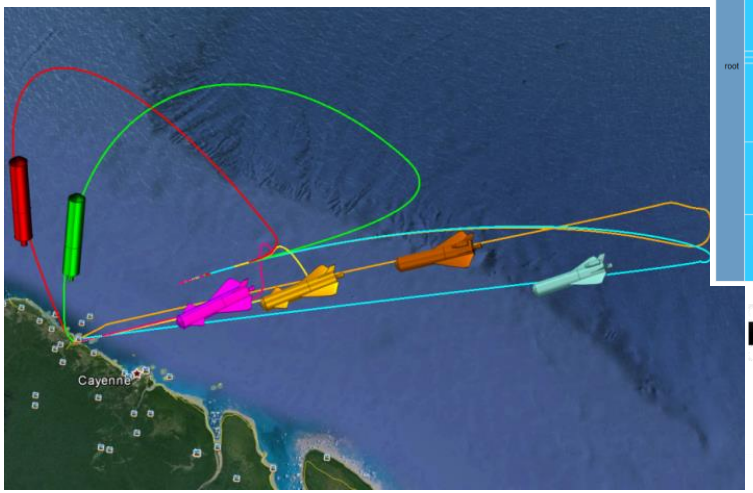


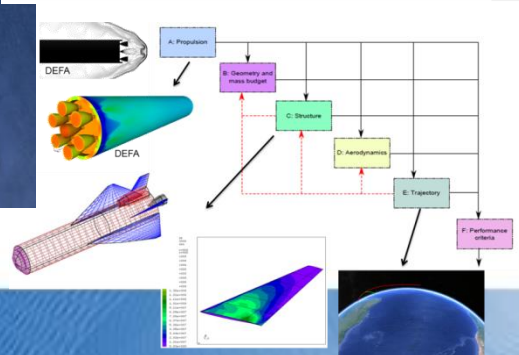
Work with CNES on reusable launch vehicle:

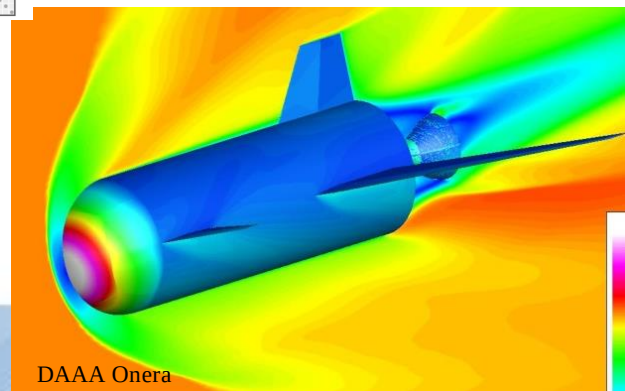
- Trade-off study between different concepts
  - Creation of an OpenMDAO process for each of them
- Use of MDO methodologies (*e.g.* optimization, surrogate model) into an integrated environment





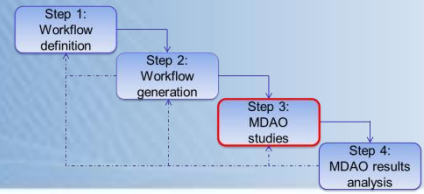


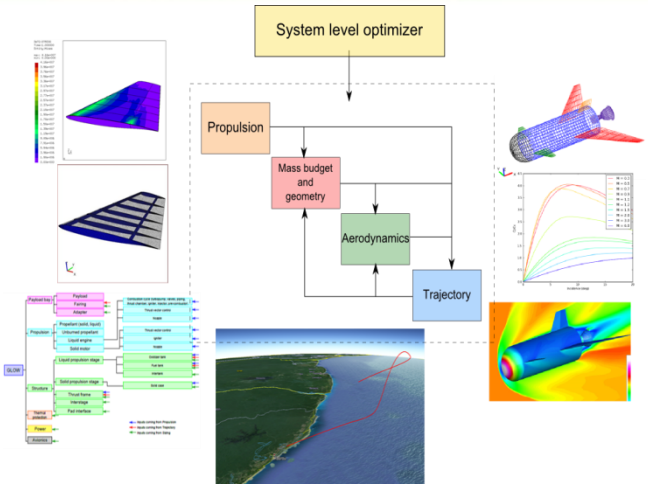


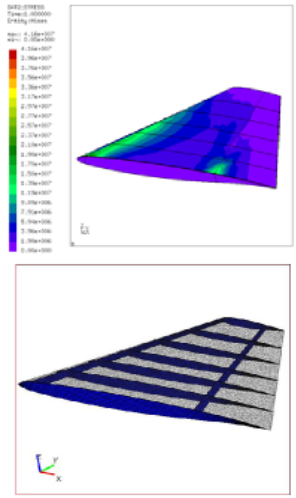


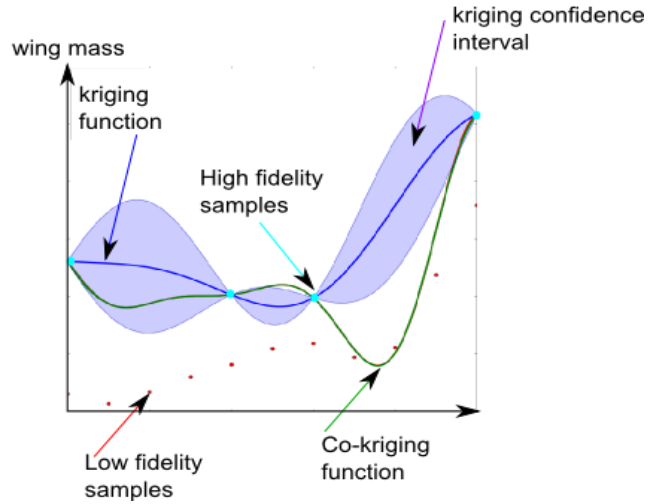
DAAA Onera

Step 3 : multi-fidelity for wing mass estimation









- **Low fidelity model**: regression model based on historical data of existing wings.
- **High fidelity model**: Finite Element Analysis based on CalculiX software (Dhondt, 2015) and OpenVSP meshing (Moore, 2015).

bConverged

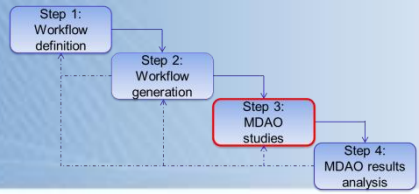
OpenVSP

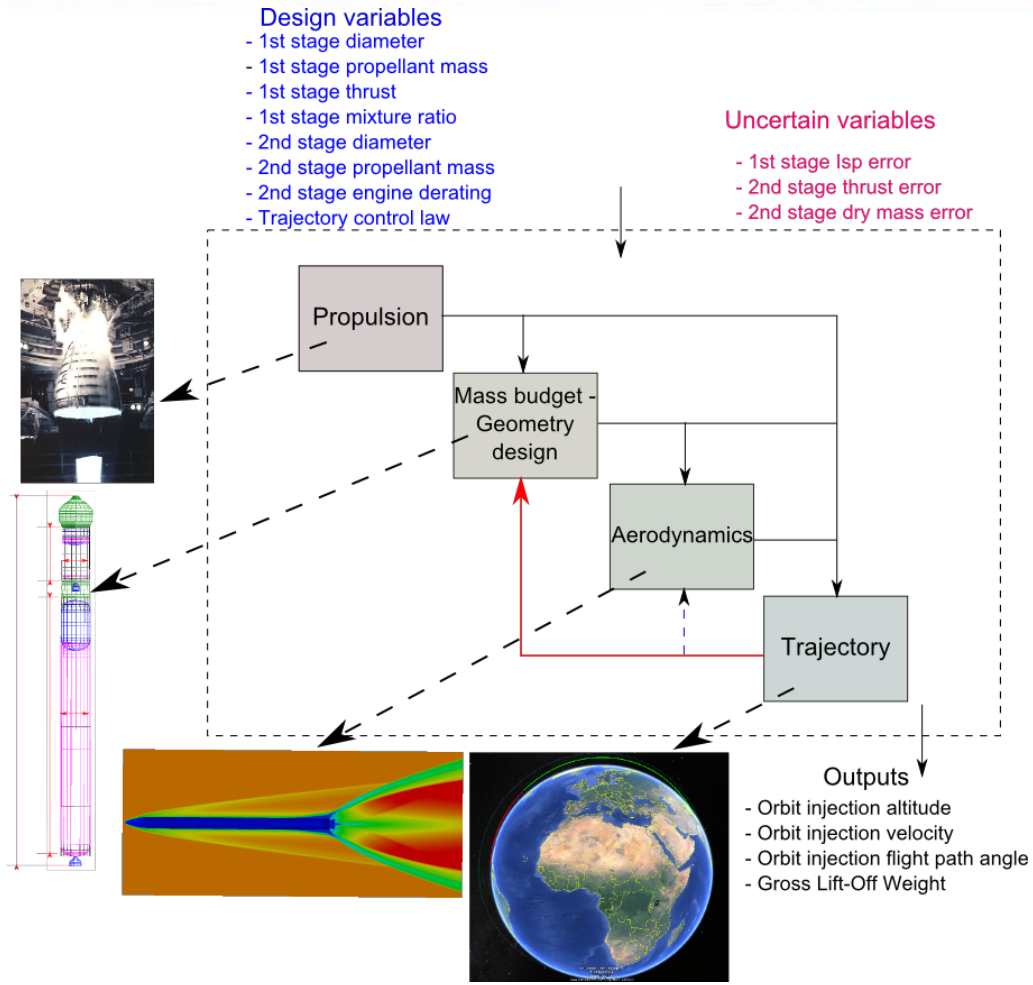
	Low fidelity	High fidelity	Co-kriging
Wing mass	502kg	573kg	570kg
Computational cost	0.001s	517.0s	0.08s

L. Brevault, M. Balesdent and S. Defoort (2017) Preliminary study on launch vehicle design - applications of Multidisciplinary Design Optimization methodologies, Concurrent Engineering: Research and Applications. SAGE, Accepted

ONERA

Step 3: MDO under uncertainty



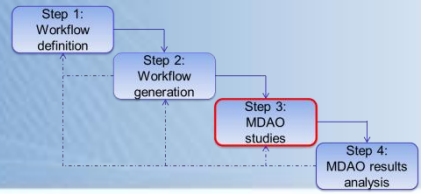


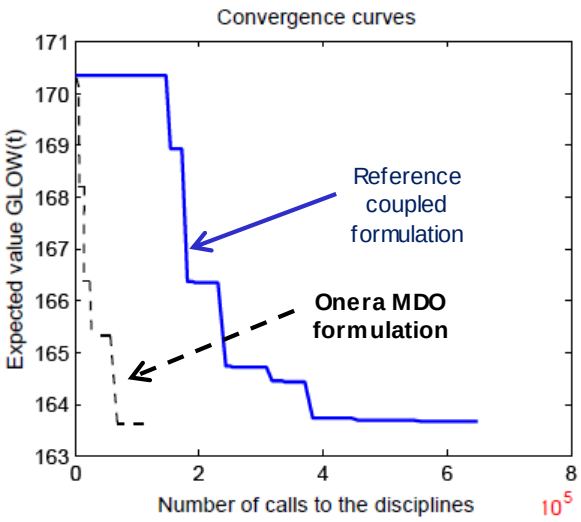
- Minimize the expected value of Gross Lift-Off Weight (GLOW),
- Constraint :
  - Injection altitude, velocity and flight path angle of payload in GTO (perigee 250km),
- Problem dimension : 27,
- 3 uncertain variables,
- Optimization algorithm : patternsearch,
- Constraint calculation : Support Vector Machine + Subset Simulation.

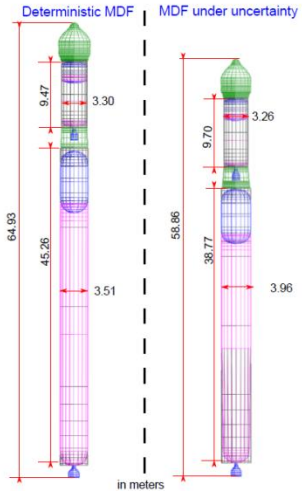
- L. Brevault, M. Balesdent, N. Bérend and R. Le Riche (2015) Decoupled MDO formulation for interdisciplinary coupling satisfaction under uncertainty, AIAA Journal, DOI : 10.2514/1.J054121,
- M. Balesdent, L. Brevault, N. Price, S. Defoort, R. Le Riche, N.H. Kim, R. Haftka and N. Bérend (2016) Advanced space vehicle design taking into account multidisciplinary couplings and mixed epistemic / aleatory unertainties. Space Engineerin: Modelling nd Optimization with Case Studies, Springer
- L. Brevault (2015) Contributions to Multidisciplinary Design Optimization under uncertainty, application to launch vehicle design, Ph.D. thesis

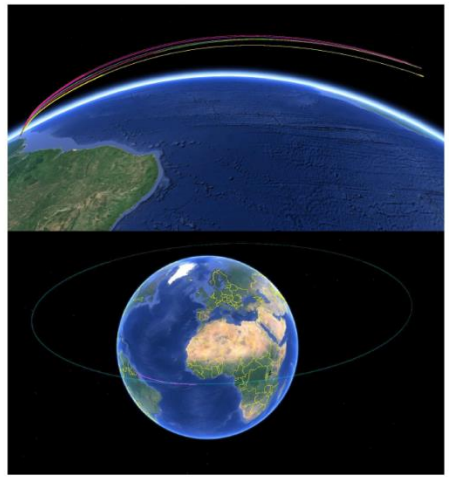
ONERA ACADIA ONERA

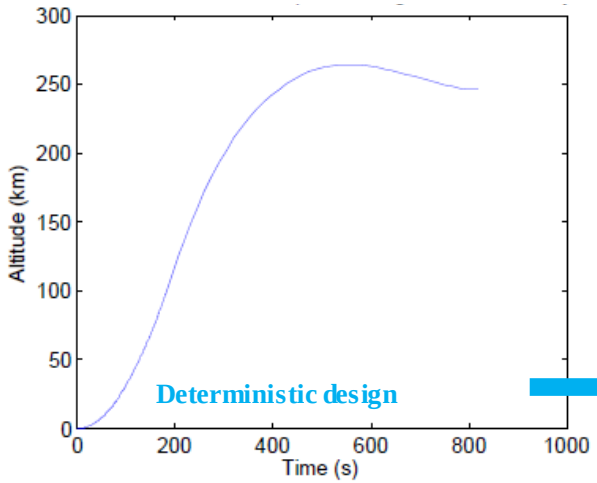
Step 3: MDO under uncertainty

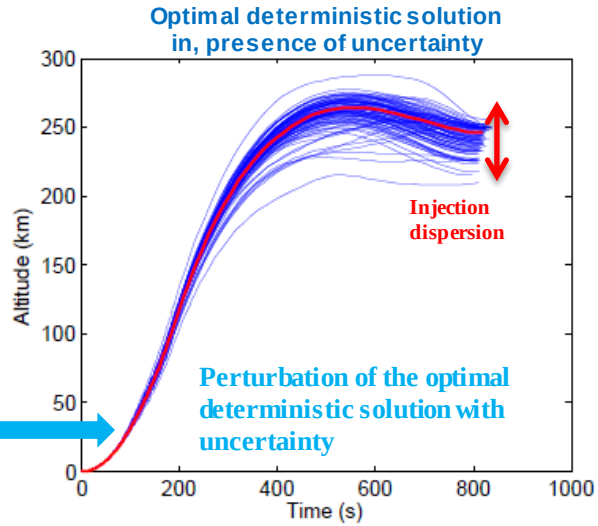


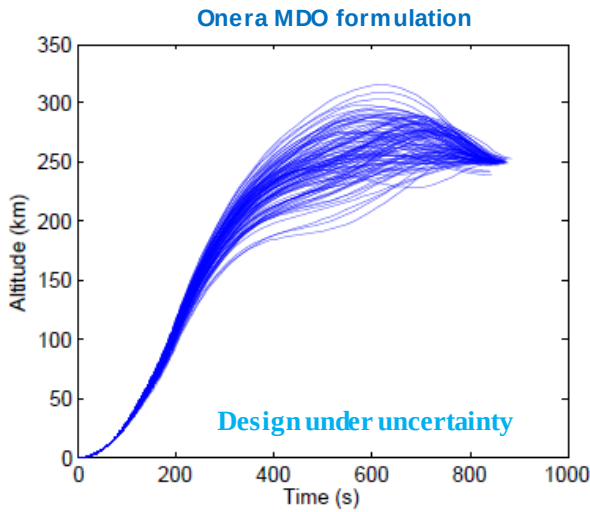




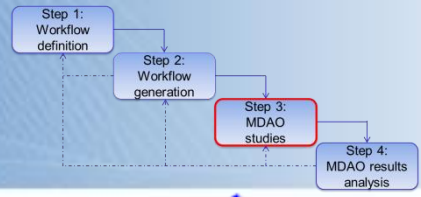








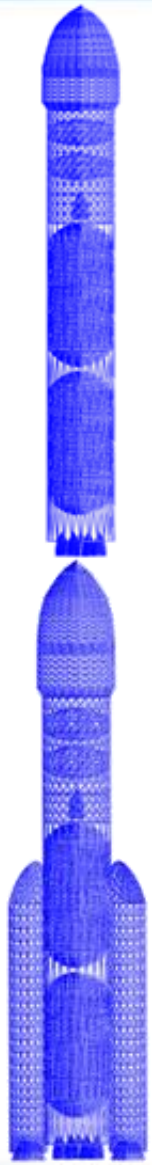
Step 3 : multi-objective MDO
[work in progress, A. Hebbal thesis 2017-2020]



MDO methodology for **multi-objective** problem to design a launcher familly (expendable / reusable) for two different missions with the same 1st stage

- Launch vehicle missions:
  - Payload **3t in GTO** with toss-back for the 1st stage (droneship)
    - Design variables (dimension 23):
      - Propellant mass 1st stage,
      - Propellant mass 2nd stage,
      - Command law trajectory
        - Ascent phase,
        - Descent phase (with two rocket boost)
      - Propellant mass allocation for the return
  - Payload **8t en GTO** with two solid rocket booster
    - Design variables (dimension 18):
      - Propellant mass 1st stage
      - Propellant mass 2nd stage,
      - Command law trajectory:
        - Ascent phase

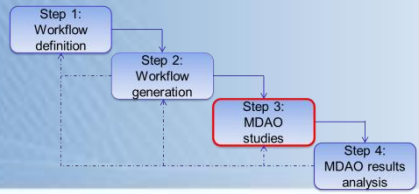
- Design variables for both configurations
- Specific design variables

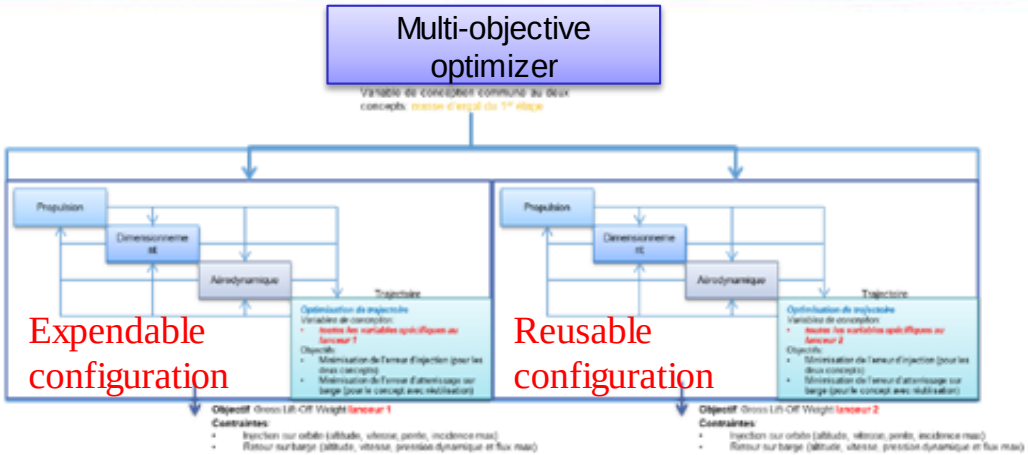


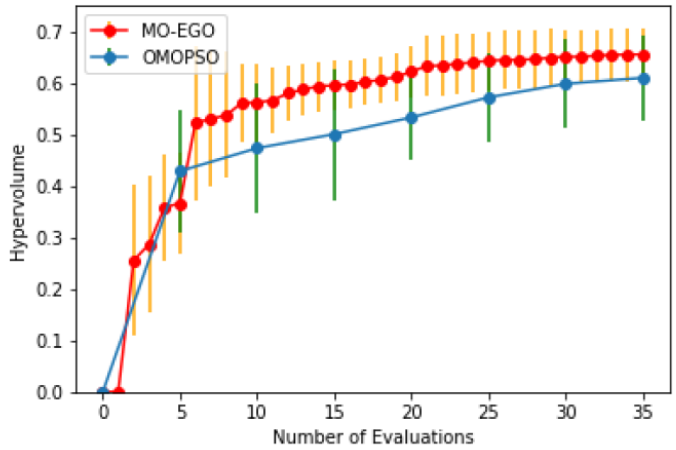
ONERA HERACLES

ONERA

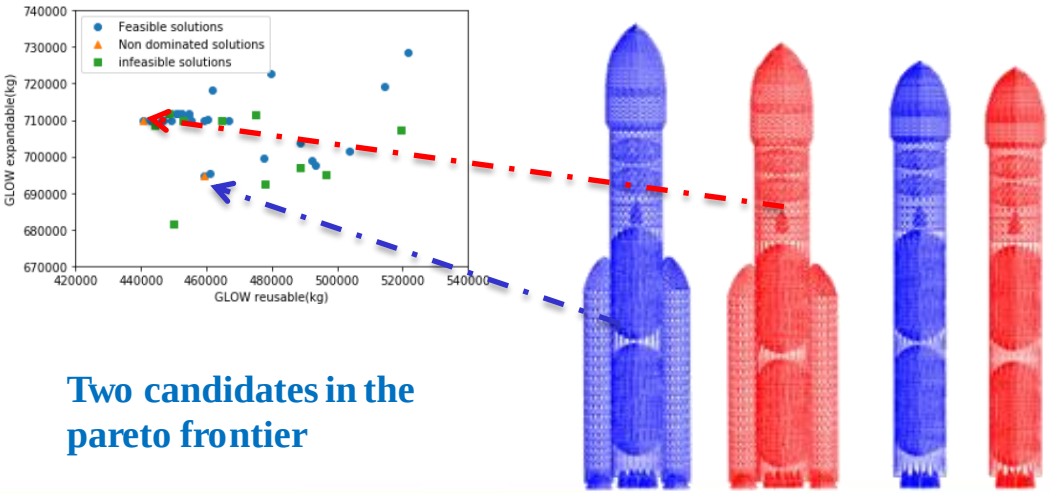
Step 3 : multi-objective MDO
[work in progress, A. Hebbal thesis 2017-2020]



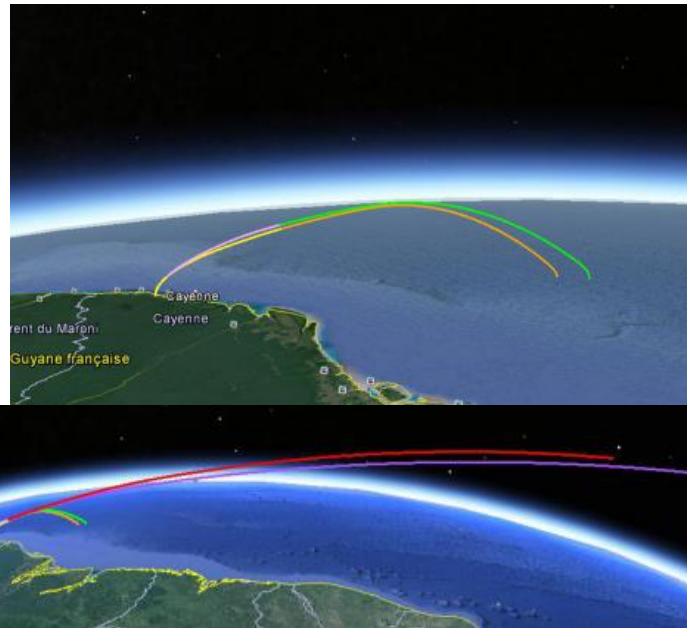




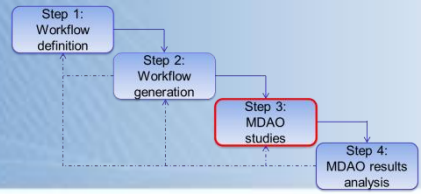
- Use of Efficient Global Optimization [1] (Kriging + Hypervolume Expected Improvement) approach for multi-objective problem



Two candidates in the pareto frontier



Step 3: Surrogate model with technological choices
[work in progress, J. Pelamatti thesis 2017-2020]

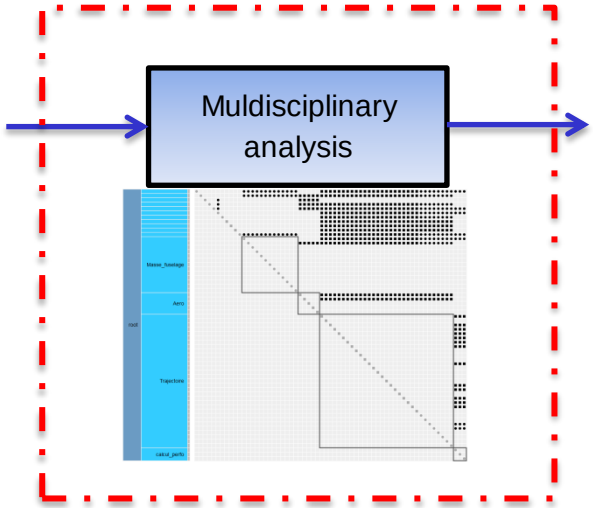


Inputs

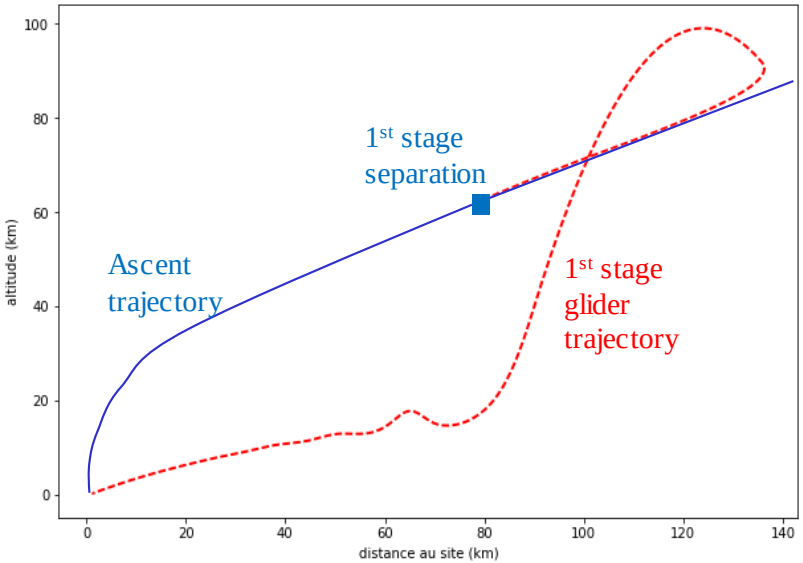
Categorical choices:
- 3 types of wing
- 2 types of fairing

Discrete choices:
- Number of engine for boost back

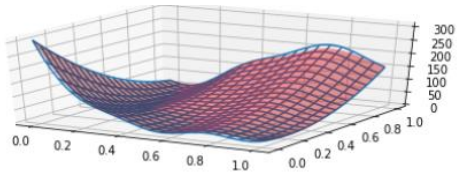
Continuous variables:
- Propellant mass 1st stage
- Specific impulse 1st stage engine
- Wing area

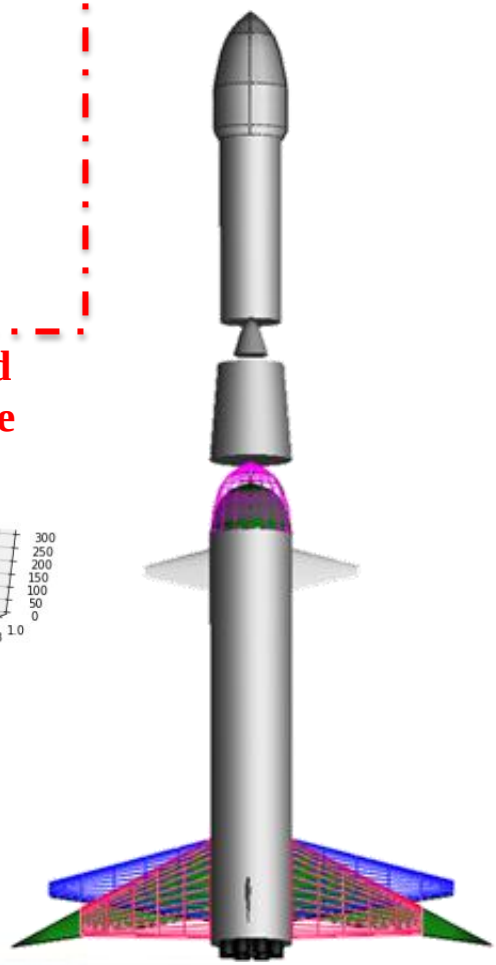


Return site distance



Creation of a mixed continuous / discrete surrogate model

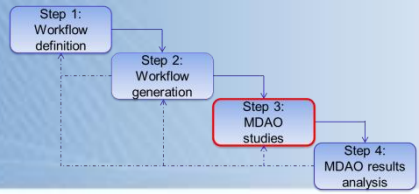


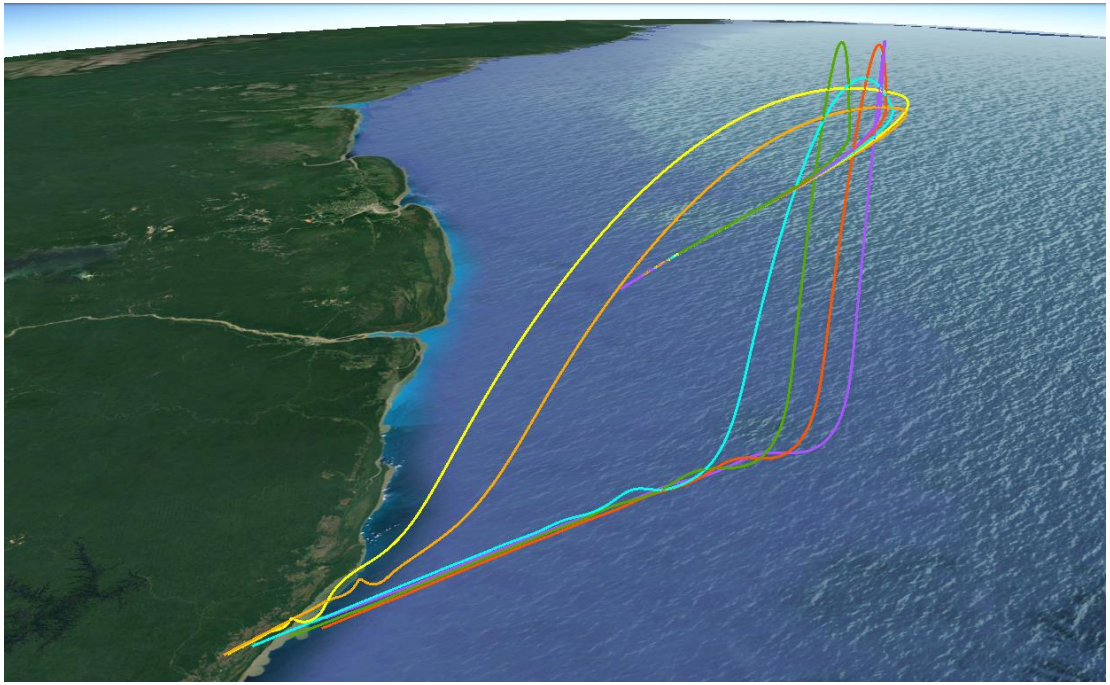


ONERA HERACLES

ONERA

Step 3: Surrogate model with technological choices
[work in progress, J. Pelamatti thesis 2017-2020]

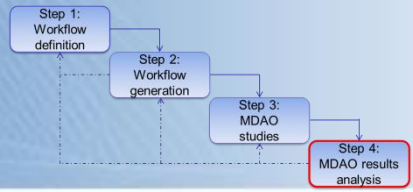




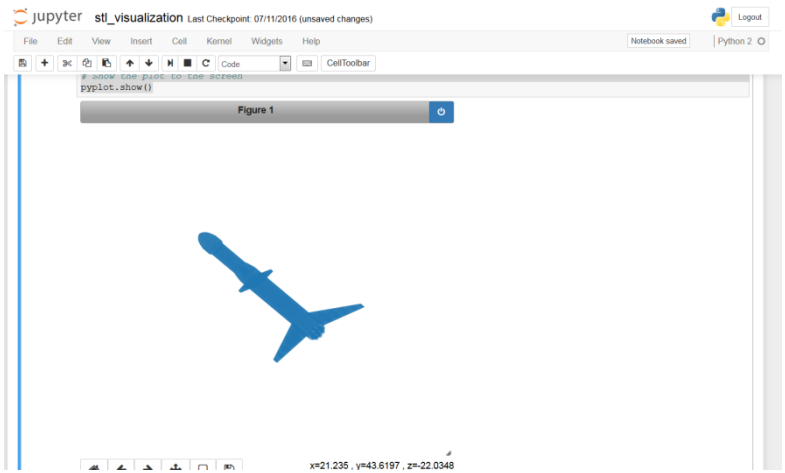
RMSE	Kriging / category	Mixed Kriging 1	Mixed Kriging 2	Mixed Kriging 3
DoE 1	9,353	4,285	4,227	4,628
DoE 2	6,819	3,554	3,532	4,567

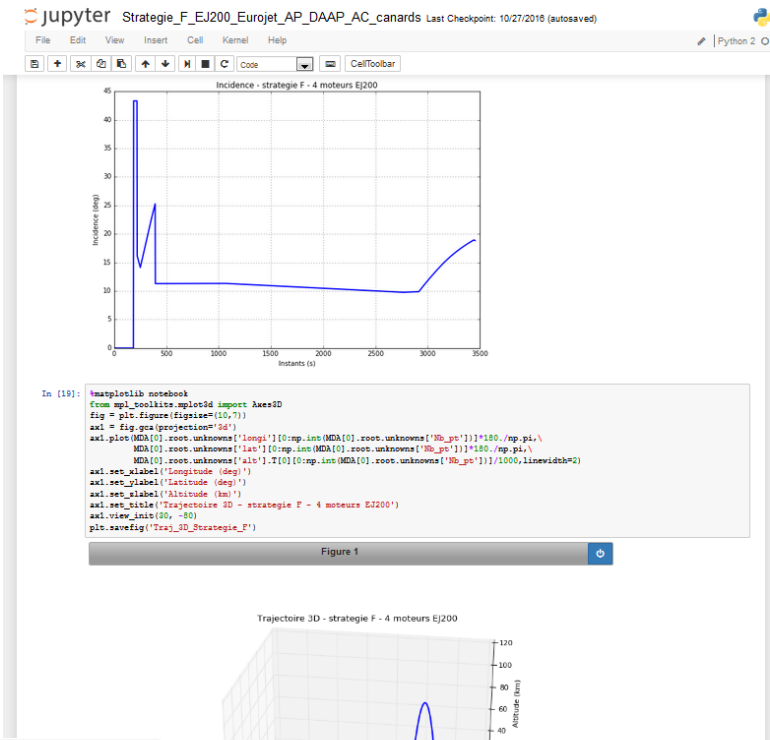
J. Pelamatti, L. Brevault, M. Balesdent, E.G. Talbi, Y. Guerin, Overview and comparison of Gaussian process-based surrogate models for mixed continuous and discrete variables, application on aerospace design problems, 'High-performance simulation based optimization', Springer, Review in progress

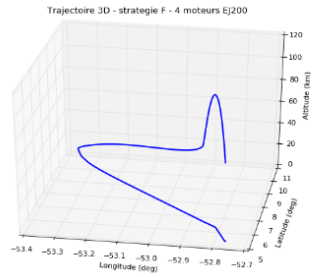
Step 4: MDAO results analyses



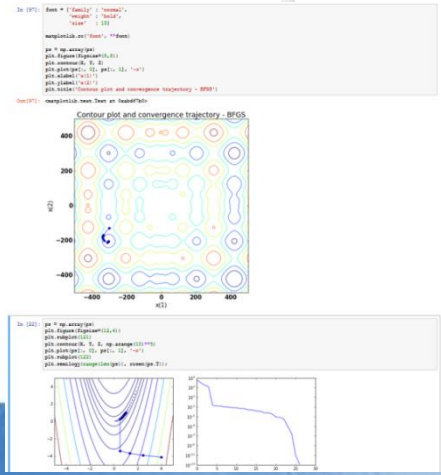
- Use of Jupyter Notebook for archive and visualization
  - Integrated vehicle visualization (*.stl*)
  - Integrated launch vehicle performance visualization
- Export for visualization (GoogleEarth, *etc*.)





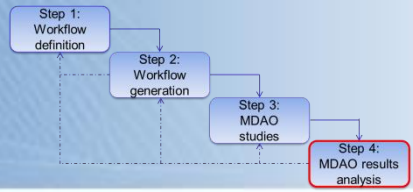


Google Earth



ONERA

Step 4: MDAO results analyses



- Creation of tutorials for WhatsOpt tools and methodological libraries based on Jupyter Notebook

Monte Carlo avec OpenTurns et une fonction définie dans OpenMDAO

Objectifs:

- Définir une discipline dans OpenMDAO
- Utiliser OpenTurns pour effectuer une propagation d'incertitudes par Monte Carlo
- Bénéficier des outils statistiques offert par OpenTurns
- Tracer l'ensemble des points défaillants et non défaillants dans un problème à deux dimensions

Chargement des packages utiles

- OpenMDAO
- OpenTurns
- Matplotlib
- Numpy

```
In [2]: from openmdao.api import IndepVarComp, Component, Problem, Group
        from openturns import *
        import matplotlib.pyplot as plt
        import numpy as np
        from openturns.viewer import View
        %matplotlib inline
        from mpl_toolkits.mplot3d.axes3d import Axes3D
        from __future__ import print_function
        import sys

        from matplotlib import cm
        default_colormap = cm.PuBuGn # the default colormap that we will use for our plots
```

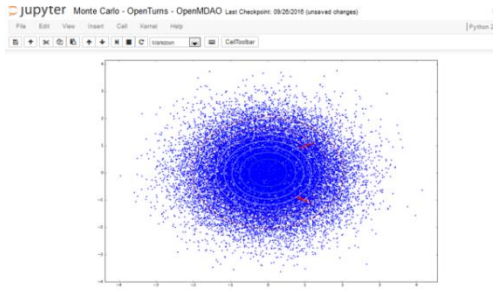
Definition d'une discipline dans l'environnement OpenMDAO

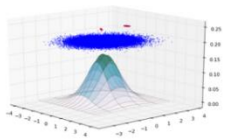
$$\mathbb{R}^2 \to \mathbb{R}$$
$$f(x,y) = 100 * (x - y^2)^2 + (x - 1)^2$$

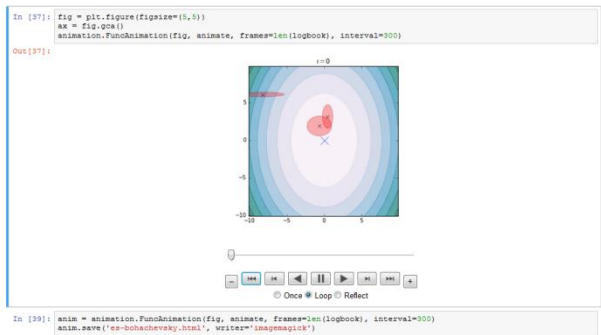
```
In [3]: class Rosenbrock(Component):

            def __init__(self):
                super(Rosenbrock, self).__init__()

                self.add_param('x', val=0.0)
                self.add_param('y', val=0.0)

                self.add_output('f_xy', val=0.0)

            def solve_nonlinear(self, params, unknowns, resids):
                x = params['x']
                y = params['y']

                unknowns['f_xy'] = 100*(x-y**2)**2+(x-1)**2
```

Construction du problème, connexion des variables dans l'environnement OpenMDAO

```
In [4]: top = Problem()
        root = top.root = Group()
        root.add('p1',IndepVarComp('x',2.0))
        root.add('p2',IndepVarComp('y',-4.0))
        root.add('p',Rosenbrock())
        top.setup()
```







ONERA ACADIA

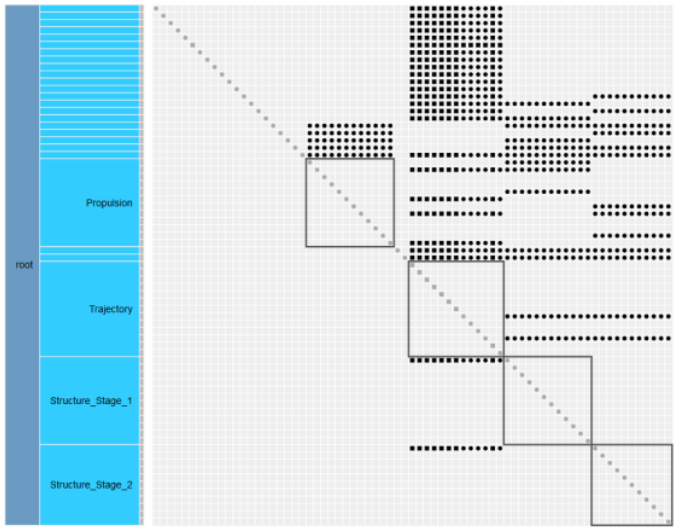
ONERA

Some remarks on the use of OpenMDAO

- Use of OpenMDAO 1.x as an integrator tool to connect the disciplines, no access to the Jacobian and no use of gradient-based optimizer

- Offer the possibility to not «initialize» the size of some state output variables (unknown at the beginning of the MDA)
  - State variables during trajectory: their sizes are triggered by some events (jettison, landing, *etc.*)

- Offer the possibility to «initialize» some output variables (former « params ») for the MDA coupling satisfaction depending on other variable values:
  - Dry mass launch vehicle stages depending on propellant mass (and structural index)



As users, if possible, some stability in the main characteristics of OpenMDAO and a maximum of backward compatibility

Conclusions & Perspectives

- Design of Reusable Launch Vehicles requires:
  - Multidisciplinary design optimization approach to capture complex physical phenomena and their interactions,
  - Management of uncertainty, multi-fidelity, surrogate models, optimization tools to perform trade-off studies.

- ONERA is using OpenMDAO combined with in-house and opensource librairies
  - OpenMDAO facilitates the discipline integration and coupling,
  - OpenMDAO enables to perform integrated multidisciplinary analyses and optimization with more flexibility than « commercial alternatives »

- ONERA is working on several axes of research for aerospace vehicle design:
  - MDO with technological choices,
  - MDO with multiple criteria,
  - MDO under uncertainty for high dimensional problems, *etc*.

ONERA ACADIA | ONERA CICAV | ONERA HERACLES

Thank you for your attention

Any questions ?