TO GET STARTED :

ANACONDA®

+

Python 2.7

+

openMDAO V 1.7.3

HOW TO START:

- openMDAO tutorials : http://openmdao.readthedocs.io/en/1.7.3/usr-guide/tutorials.html
- Open an AnacondaPrompt Instance and write the command **pip install openmdao==1.7.3**

FOCUS ON:

- Problem structure
- Management of variables
- Connections between components

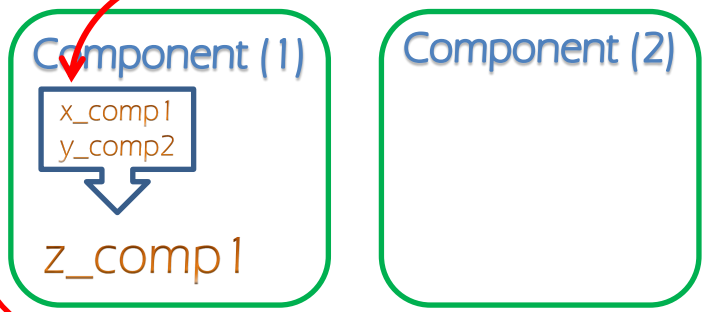
PROBLEM STRUCTURE

Problem

Group

Create openMDAO variables

add.(x ,IndepVarComp())
add.(y ,IndepVarComp())



MAIN SCRIPT

Define your variables

x y

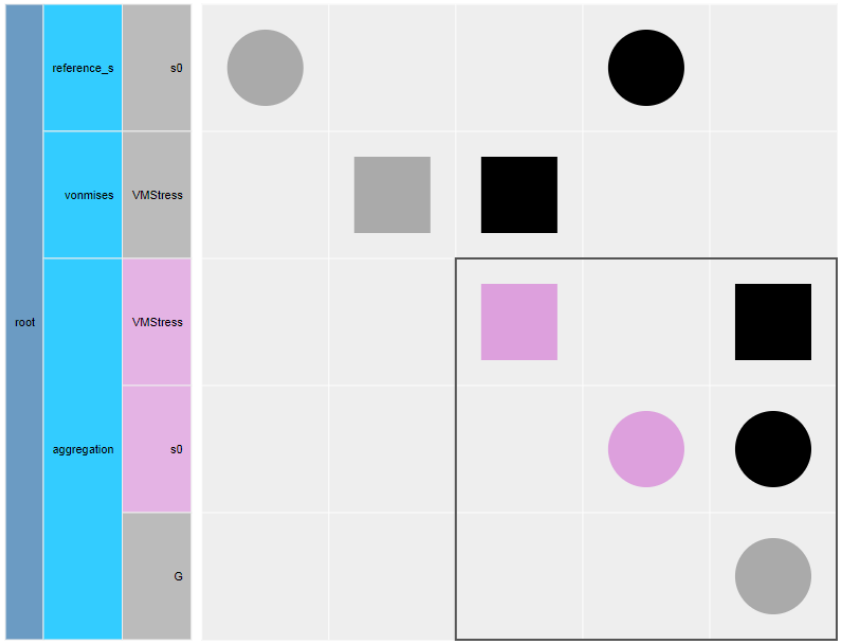
Connect the variables to the component's varibles

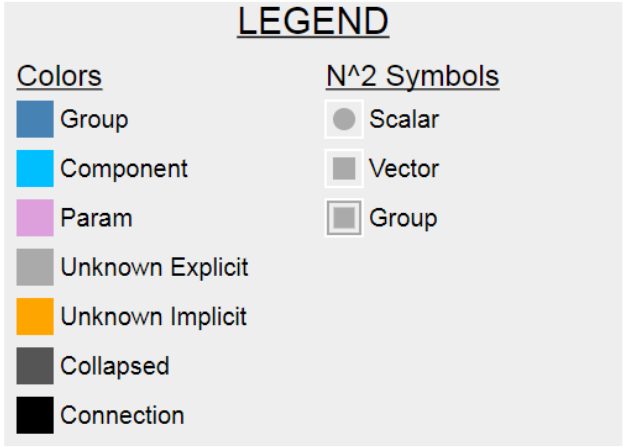
Set the problem (objective of optimization, constraint, design variables)

RUN IT

MANAGEMENT OF VARIABLES:

OpenMDAO storages variables into a dictonary named 'params'.
Each component has their params, and one or more outputs saved as 'unknowns'.





Is important to connect the variables of each component, so, when during the process the value of a variable changes, the other components have access to the modified value of the variable.

STRUCTURE OF A COMPONENT :

```
#Component which gives the aggregation function (G) given the Von Mises Stresses
class Aggregation(Component):
    def __init__(self,n_stress,p,function):
        super(Aggregation,self).__init__()
        #Number of Von Mises Stresses obtained
        self.n_stress=n_stress
        #Draw-down factor
        self.p=p
        #Aggregation function type
        self.function=function
        #Von Mises stresses vector
        self.add_param('VMStress',np.zeros(self.n_stress))
        #Reference vqlue for normalization of stresses
        self.add_param('s0',val=0.0)
        #Aggregation function
        self.add_output('G',val=0.0)

    def solve_nonlinear(self,params,unknowns,resids):
        n_stress=self.n_stress
        p=self.p
        function=self.function
        VMStress=params['VMStress']
        s0=params['s0']
        gmax=max(VMStress)
        summ=0.0
            for k in range (n_stress):
                summ+=np.exp(p*((VMStress[k]-gmax)/s0))
            G=((gmax/s0)+(np.log(summ)/p)-(np.log(n_stress)/p))*s0

        unknowns['G']=G
```

INITIALIZATION

Define variables used in the function, defined in the main, and define params and outputs of the component

FUNCTION

Define local variables, do operation on this variables and set the outputs in the unknowns dictionary

```
top = Problem()
top.root = root = Group()
root.add('vonmises', IndepVarComp('VMStress', VMStress), promotes=['*'])
root.add('reference_s', IndepVarComp('s0', s0), promotes=['*'])
root.add('aggregation',Aggregation(n_stress,p,function),promotes=['*'])
```

SETTING PROBLEM

Define the Problem, the group, add and init the params, and the component that u need

ACCESS TO THE DATA :

To access variables during the optimization loop you have to set a recorder.
The recorder create a dictonary with all the params and unknowns for each iteration.
Using the openMDAO module you can open the dictionary for each iteration and save the value of each variables in a numpy array.

```
import sqlitedict
import numpy as np
import matplotlib.pyplot as plt
db = sqlitedict.SqliteDict( 'opti_g_16', 'iterations' )
print( list( db.keys() ) )
n=len(db)
x=np.arange(0,n)
cd=np.array([])
g=np.array([])
m=np.array([])

#=============================================================
for j in range(0,n):
        data = db['rank0:COBYLA|'+str(j)+'|root|'+str(j+1)]
        print('rank0:COBYLA|'+str(j)+'|root|'+str(j+1))
        c = data['Unknowns']
        g = np.append(g,c['G'])
        cd =np.append(cd, c['CDi'])
        a = np.append(a,c['alpha'])
        m = np.append(m,c['mass'])

#=============================================================
```

Create a dictonary from the recorder file

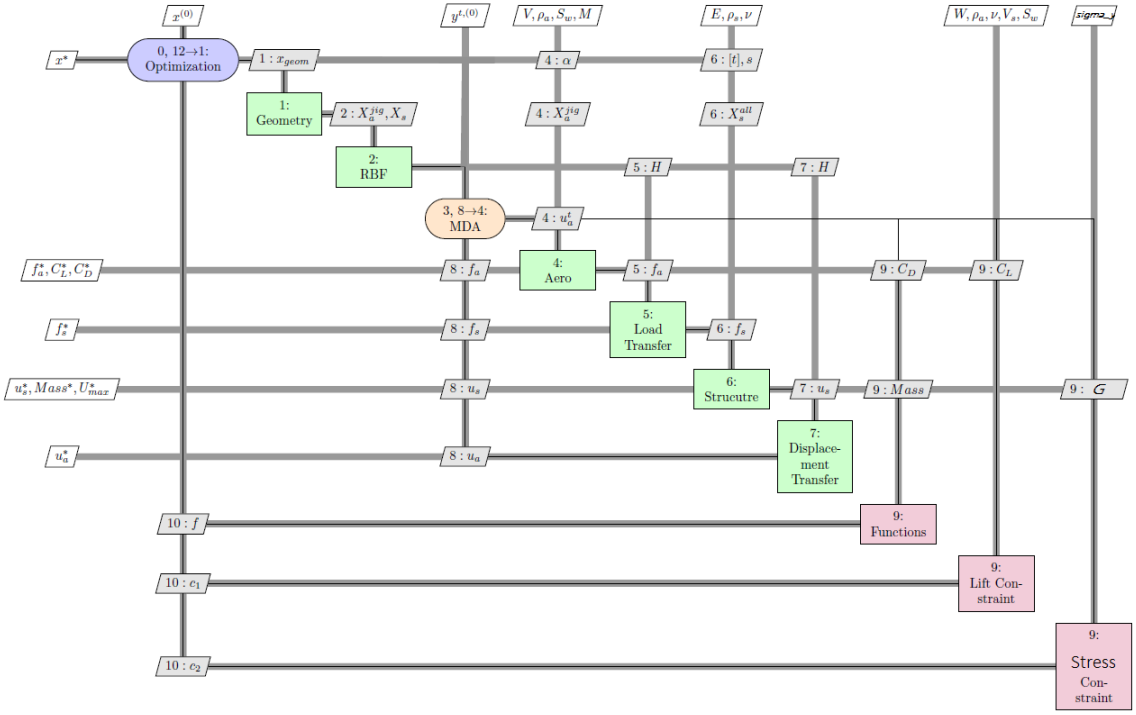
Print the name of each iteration

Extract the dictionary relative to one iteration

Extract from the dictionary the dictionary relatated to the variables of interest

Save the value of that variables into a list

AN EXAMPLE OF OPTIMITAZION:



Aeroelastic Optimization:
Minimize an object like induced drag or wing's mass, modifying geometry or flight variables.

X_0 = [α_0, t_0]

f_a=f(u) | u=f(f_a)

MDA LOOP

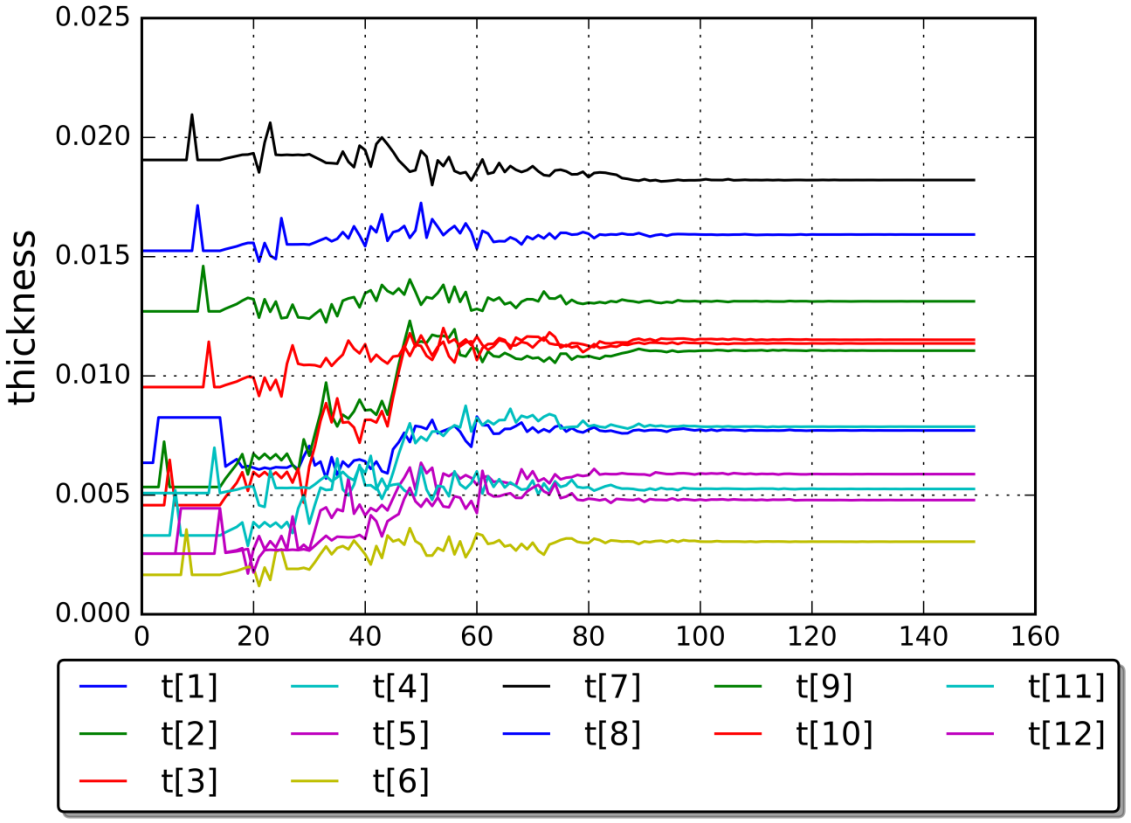
f_a(x_0) | U(x_0)

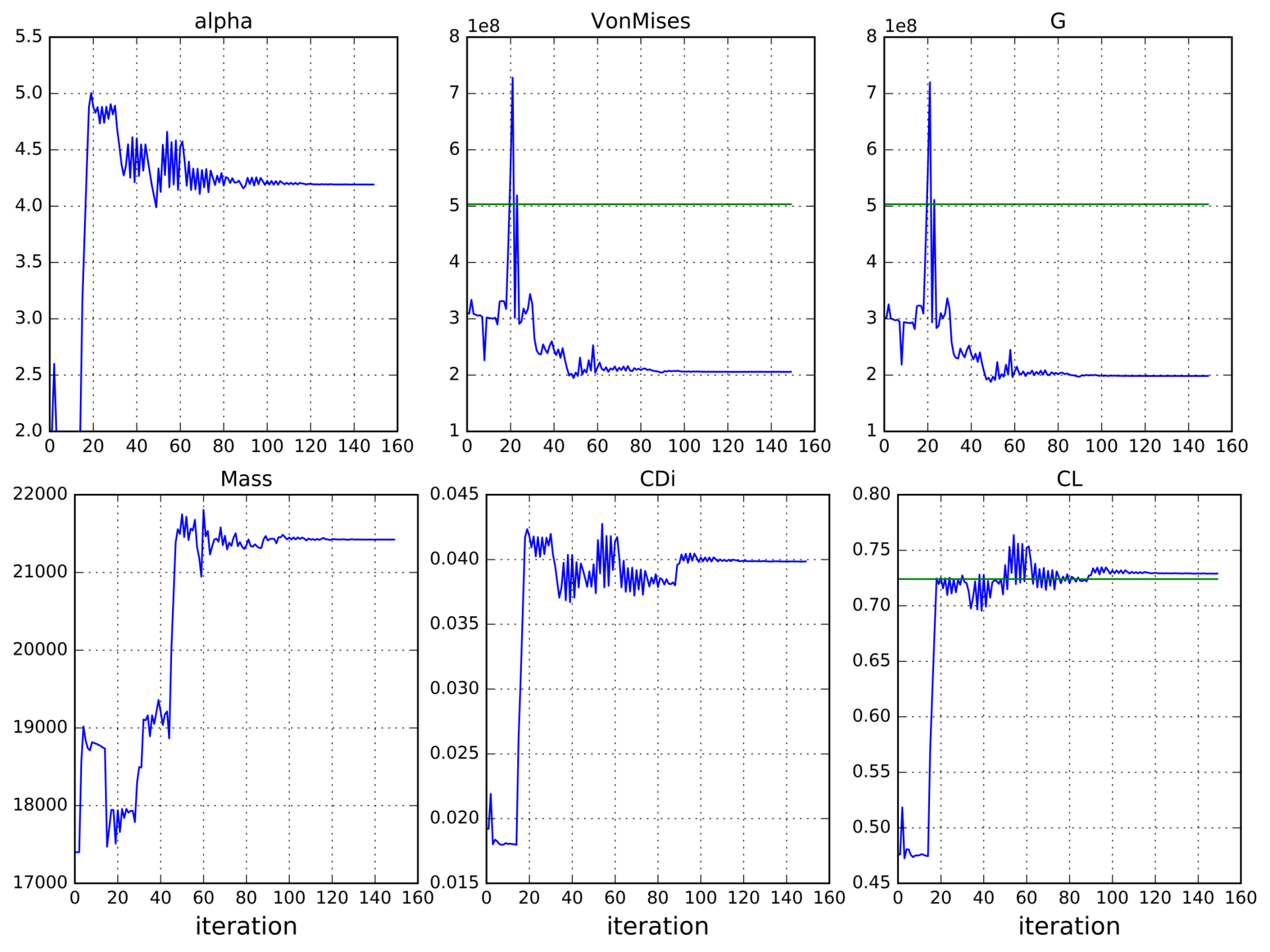
Objective value [mass, CDi]

C_1=CL>W/q | C_2=σ_max<σ_y

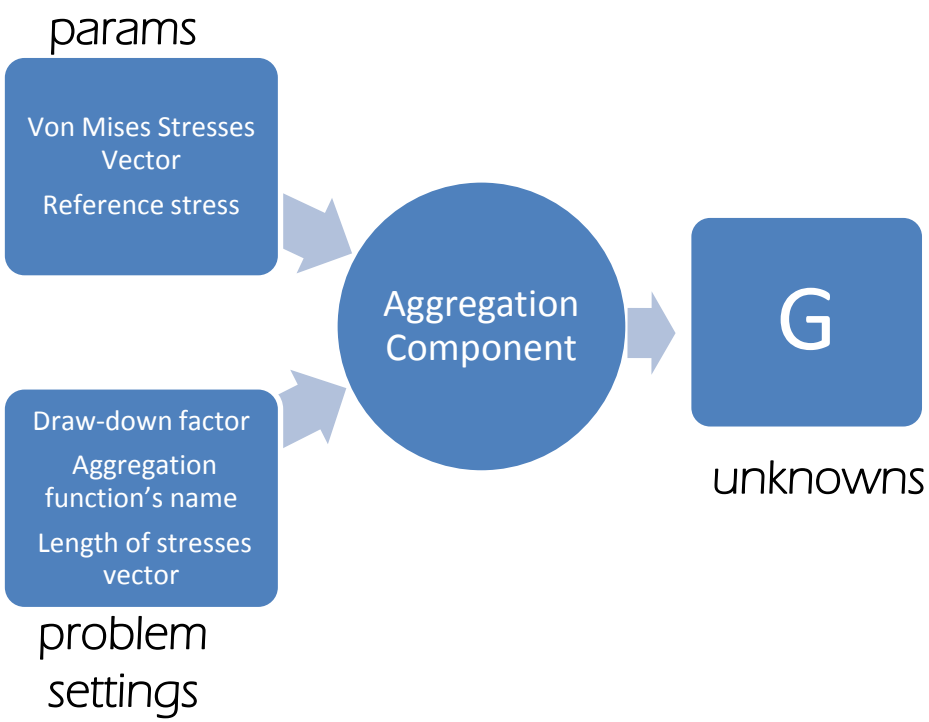
X_1 = [α_1, t_1]

Design Variables		Constraint		Objective	Optimizer
Alpha	Thickness	CL	VSM	CDi	COBYLA
0°-6°	0.001-0.02 mm	> W/q	< Yield Stress		




alpha
VonMises
1e8
G
1e8
Mass
CDi
CL
iteration
iteration
iteration

AGGREGATION COMPONENT:



$$\sigma_{v\_s} = \{\sigma_i, \dots, \sigma_n\}$$
$$n = 2 * n_e$$

$$G \approx \max(\sigma_{v\_s})$$

$$\Psi_{KS}^{L} = f_{max} + \frac{1}{p} \ln \sum_{i=1}^{N} e^{p(f_i - f_{max})} - \frac{\ln N}{p}$$

Passing from 2 * number of element constraint to 1 constraint, by a continue function representative of the maximum of the stresses vector.

DRAW-DOWN FACTOR INFLUENCE: