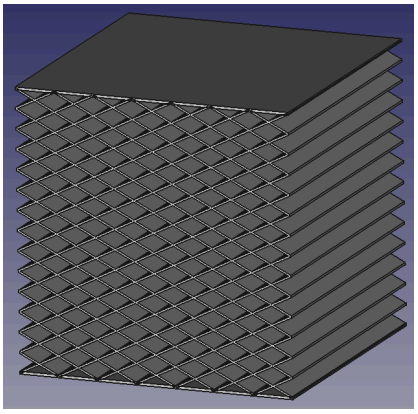


Notice d’Utilisation

Lattybrides

Génération de Structures Lattices Hybrides à Gradients de Réseau

Version 2.2

Ce logiciel a été développé dans le cadre d’un projet de fin d’étude (PLP23INT16) à l’INSA Hauts-de-France. (Promo 2023).
Ce logiciel a été développé dans le but de filtrer et d’afficher des données de crash.
Développeurs : HERMAN Adrien (Noyau et Structure Losange), BACOUT Valentin (Structure Cosinus, Hexagones + Triangles 2D [Alignés ou Non] et Triangles 2D), BENHILAL Salma (Carré + Arcs)
Documentation écrite par : HERMAN Adrien.

Table des Matières

1. Notice d'Installation

a. Dépendances

Les modules python nécessaires au bon fonctionnement du code de génération des structures sont les suivants :

```
- time
- datetime
- matplotlib
- PySide
```

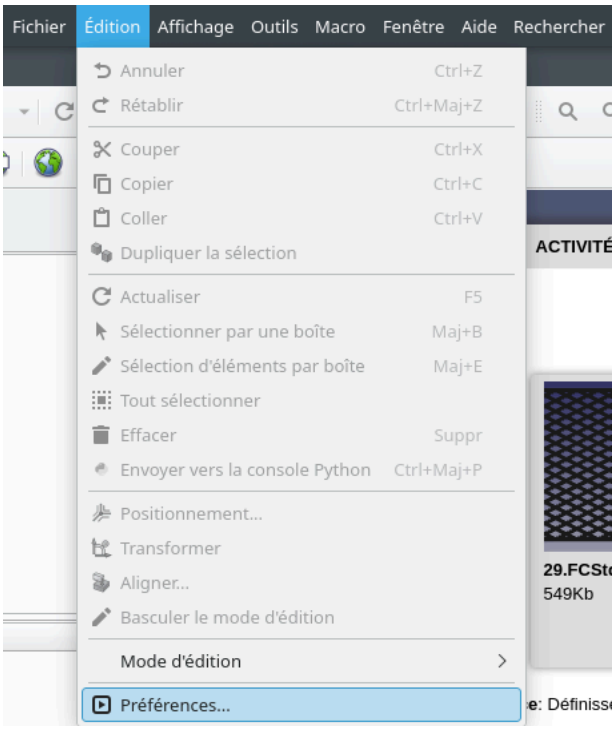
ATTENTION : ***L'atelier n'a été testé qu'à partir de la version 0.19 et jusqu'à la version 0.21.2, d'autres versions de FreeCAD sont susceptibles de faire dysfonctionner l'atelier !***

b. Installation

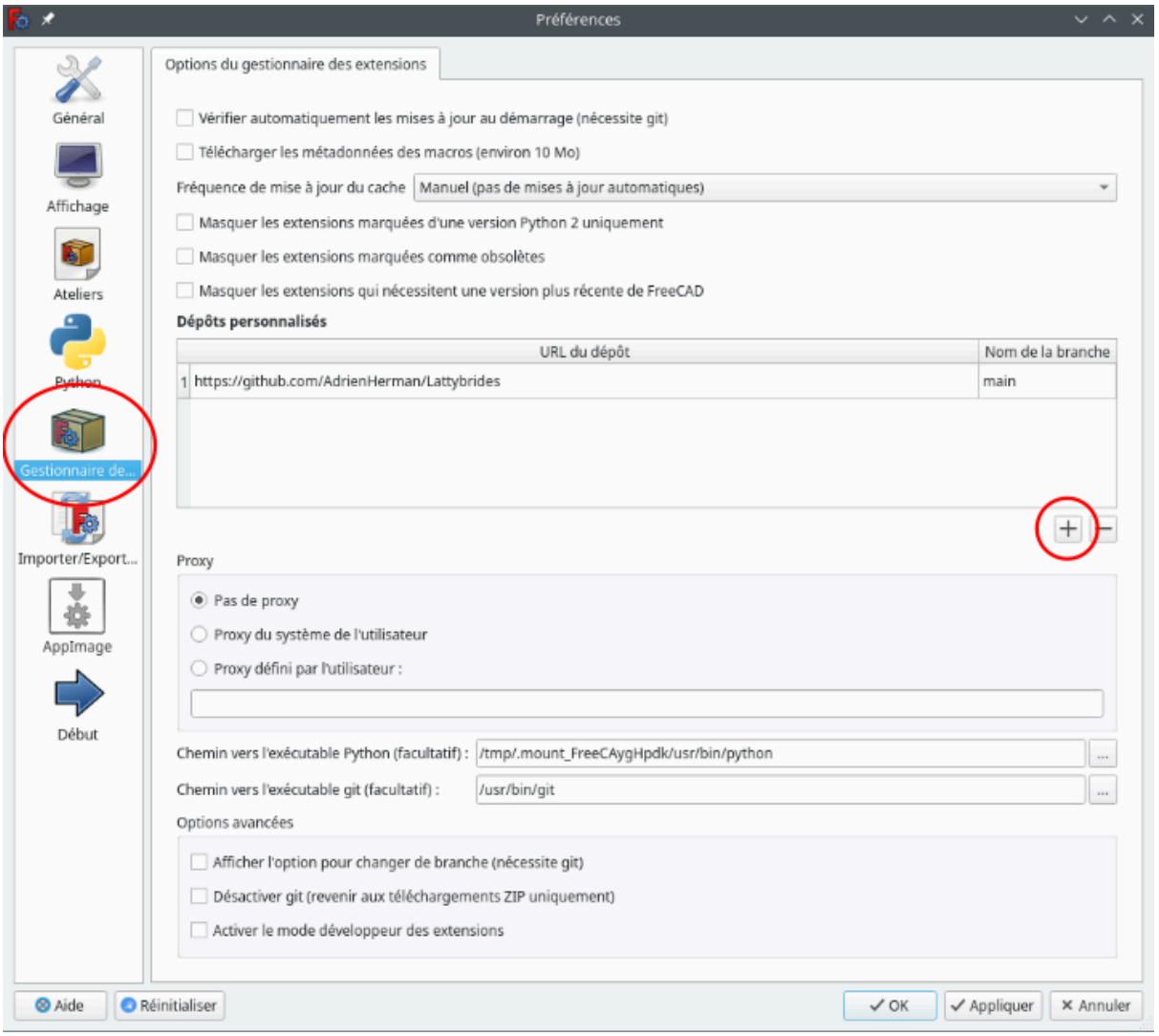
L'installation de l'atelier peut se faire manuellement en copiant le code python dans le dossier Mod de FreeCAD (voir ici : Installation Manuelle) ou en utilisant le gestionnaire des extensions (voir ici : Installation via FreeCAD).

Installation via le gestionnaire des extensions de FreeCAD

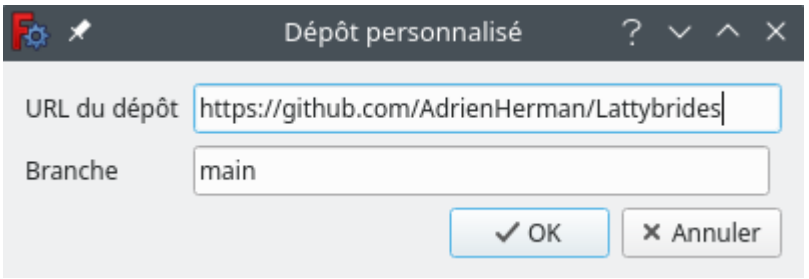
L'atelier n'étant pas répertorié dans les dépôts de FreeCAD, il vous faudra l'ajouter manuellement. Pour ce faire, ouvrez la fenêtre de "Préférences" de FreeCAD (Menu Édition -> Préférences) :



Ouvrez ensuite l'onglet "Gestionnaire des extensions" et cliquez sur le bouton "+" :



Une fenêtre s'ouvre :

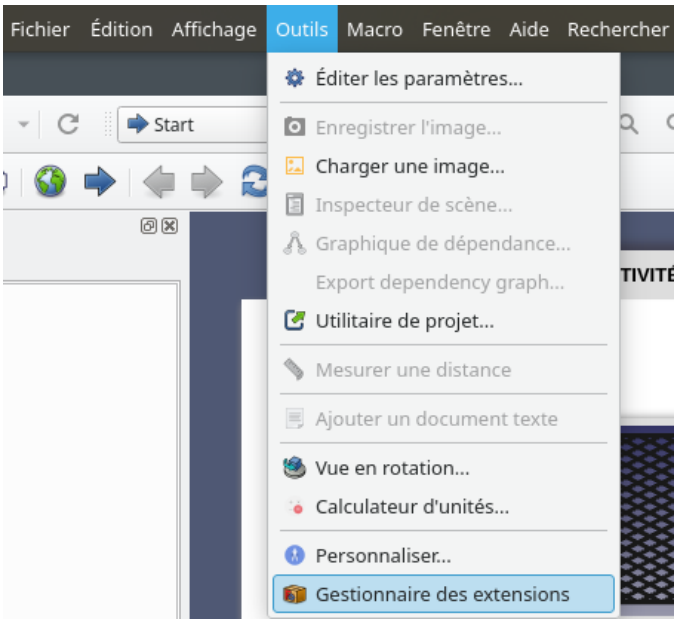


Il vous suffit ensuite de copier-coller les informations suivantes :

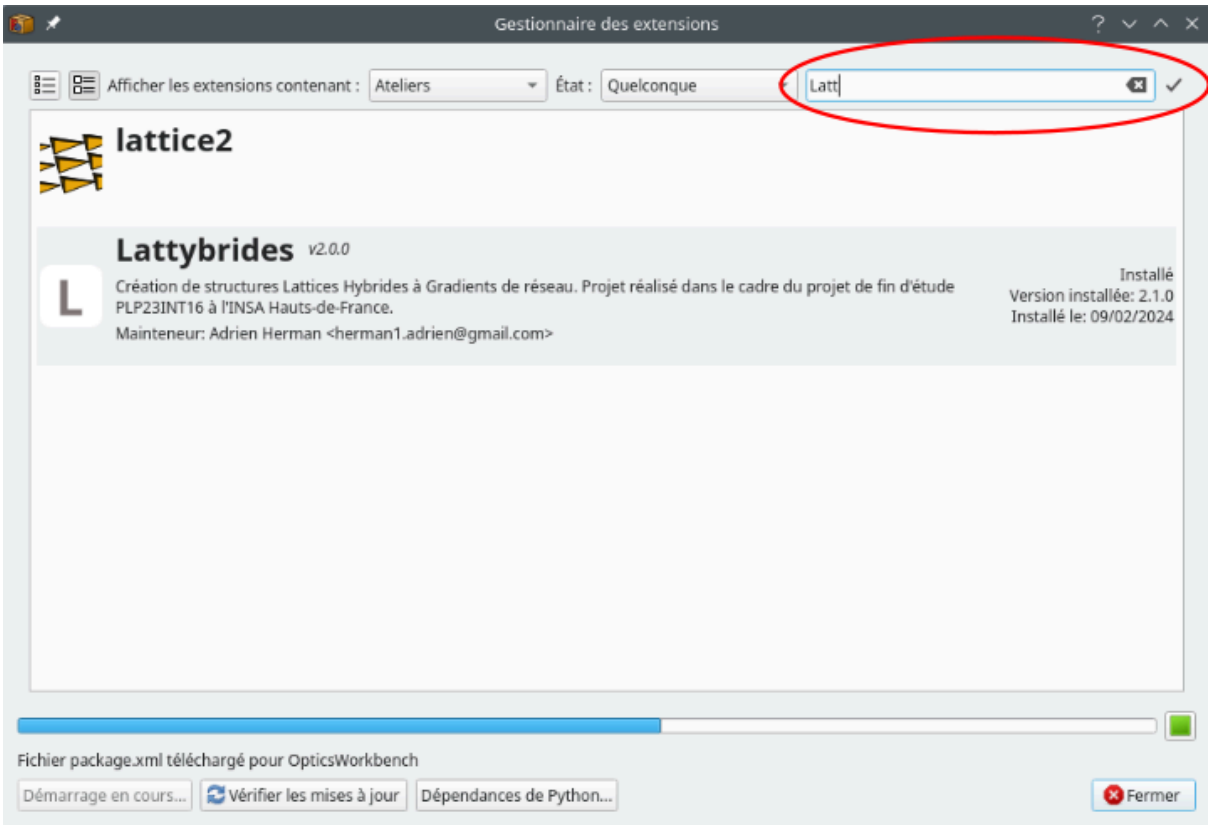
```
URL du dépôt : https://github.com/AdrienHerman/Lattybrides
Branche : main
```

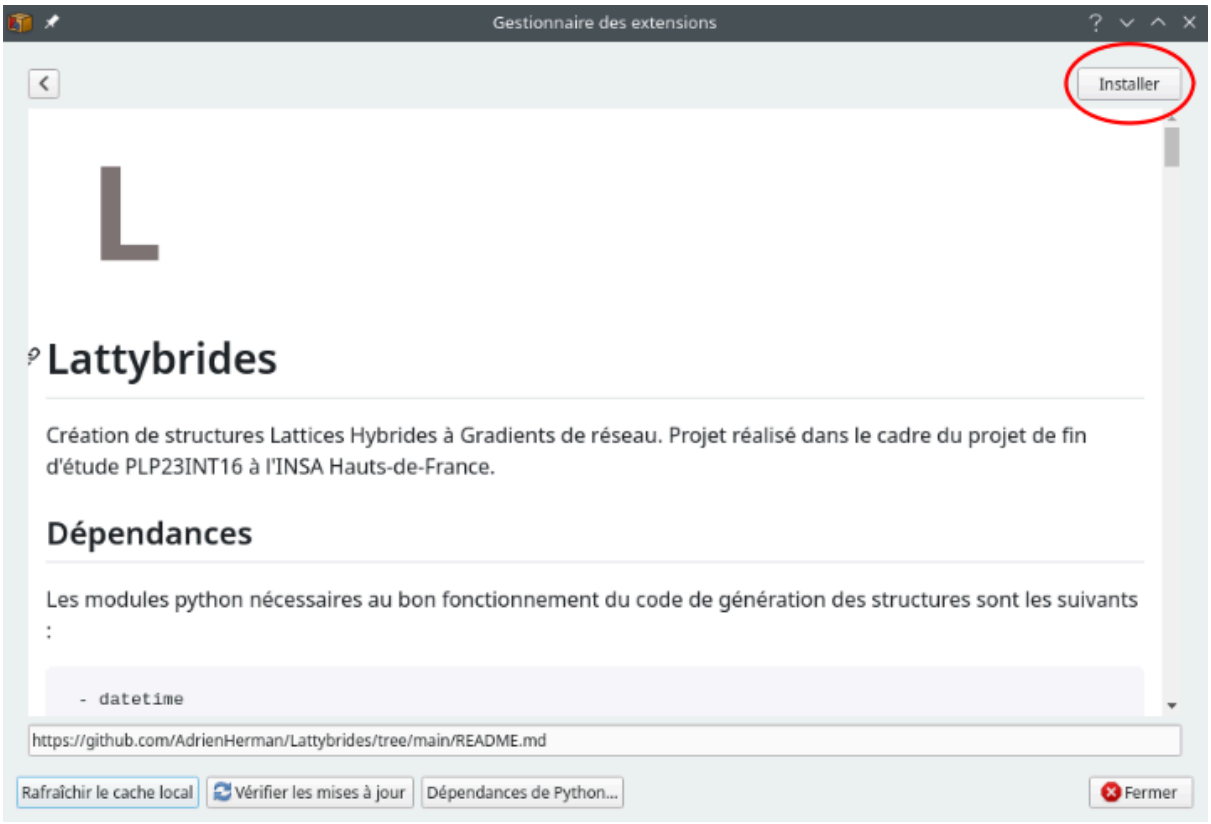
Cliquez sur "OK" et une nouvelle fois sur "OK" dans la fenêtre des préférences de FreeCAD.

Allez ensuite dans le menu "Outils" et cliquez sur "Gestionnaire des extensions" :



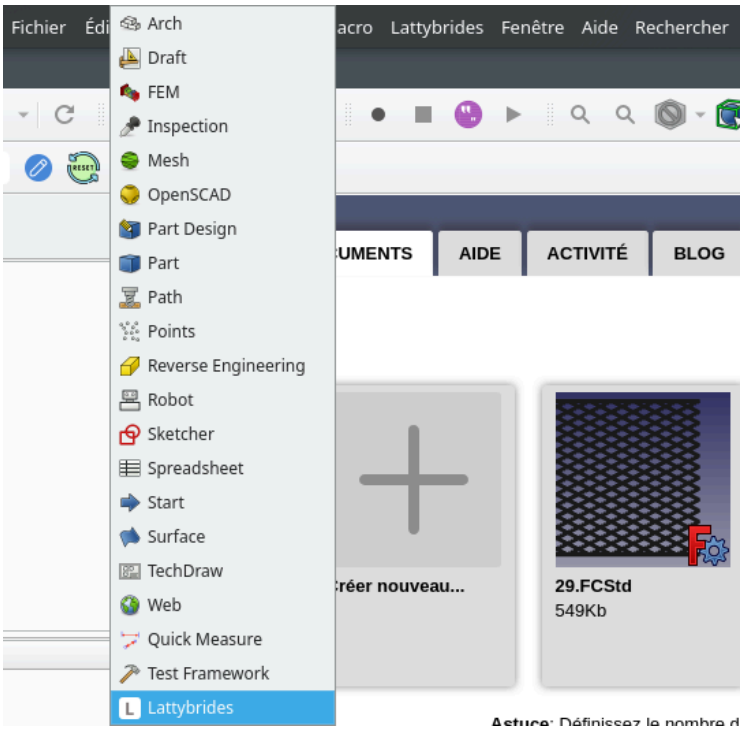
Une fenêtre s'ouvre. Recherchez "Lattybrides", l'atelier s'affiche dans la liste. Il ne vous reste plus qu'à le sélectionner et à cliquer sur le bouton "Installer" :





Les mises à jour s'effectueront également via cette fenêtre. Vous pouvez maintenant fermer cette fenêtre. FreeCAD vous demandera de redémarrer l'application afin de recharger tous les ateliers fraîchement installés.

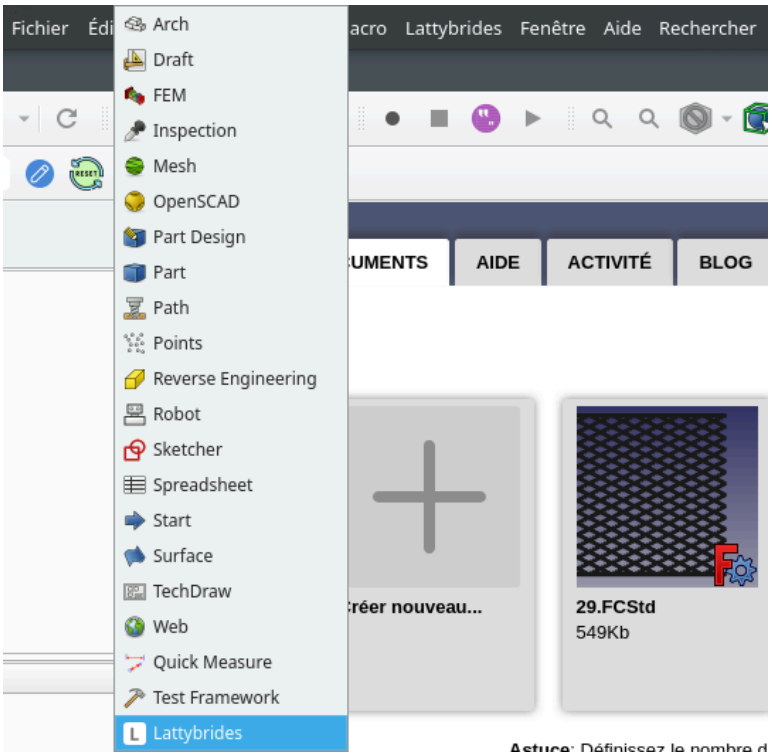
Une fois redémarré, vous pouvez accéder à l'atelier de cette façon :



2. Notice d'Utilisation

a. Interface Graphique

Après avoir démarré FreeCAD, l'atelier est censé être chargé en mémoire. Pour afficher l'atelier, il vous suffit de le sélectionner dans la liste des ateliers :



Une fois l'atelier chargé, les icônes suivantes sont affichées :

Dans ce qui suit, l'icône de gauche est l'icône numéro 1 et l'icône de droite est l'icône numéro 5.

- L'icône numéro 1 sert à lancer le calcul de la structure. Le calcul peut être long (surtout si l'optimisation de la masse est enclenchée) et ne peut pas être arrêté à moins de fermer le programme via le gestionnaire des tâches. Attention, le calcul ne prendra en compte les options du fichier config.py présent dans les fichiers d'installation de l'atelier. Si vous ne sauvegardez pas les modifications effectuées dans ce fichier, la configuration ne sera pas appliquée. À la fin du calcul, si l'exportation du fichier 3D STL est activée, une fenêtre de sauvegarde s'ouvrira afin d'enregistrer la structure à l'endroit souhaité par l'utilisateur. Même chose pour la sauvegarde du fichier FreeCAD. Une fois les fichiers sauvegardés, si l'optimisation de la masse est activée, un graphique s'affiche montrant la masse pour chaque itération du calcul.

- L'icône numéro 2 sert à ouvrir le fichier de configuration config.py dans l'éditeur de texte de FreeCAD. Vous pouvez sauvegarder les changements effectués à ce fichier de configuration avec les raccourcis Ctrl+S ou via la disquette ou le menu Fichier -> Enregistrer de FreeCAD.

- L'icône numéro 3 permet de rétablir la configuration par défaut du fichier de configuration. Cette fonctionnalité permet de revenir totalement en arrière au cas où une variable serait mal orthographiée ou manquante par exemple.

- L'icône numéro 4 permet d'ouvrir un fichier de configuration sauvegardé à un endroit extérieur aux fichiers du logiciel.

- L'icône numéro 5 permet d'effectuer une copie du fichier de configuration config.py à un endroit souhaité par l'utilisateur.

- L'icône numéro 6 permet d'afficher l'aide de l'atelier.

Les icônes 4 et 5 peuvent se révéler très utiles pour sauvegarder les fichiers de configuration pour chaque structure. Ceci permet de les générer une nouvelle fois même si le modèle 3D de la structure a été perdu ou corrompu.

Le détail des variables du fichier de configuration est explicité dans la partie suivante.

b. Paramètres du Fichier de Configuration

i. Syntaxe

Dans ce fichier les commentaires sont représentés par les caractères “`#:`”. Pour les commentaires, la ligne ne sera pas traitée par le script de lecture des paramètres. La ligne doit obligatoirement commencer par “`#:`” sans espaces ou tabulations avant.

Les lignes vides ne sont pas acceptées dans le fichier de configuration. Chaque ligne doit être soit un commentaire soit sous le format “`nom_variable:valeur:`”. Chaque ligne qui n’est pas un commentaire doit obligatoirement terminer par “`:`”.

ii. Variables

1. Partie Optimisation de la Masse

Nom de la Variable	Description	Valeurs Possibles
`optimisation_masse`	Activer ou désactiver l’optimisation de la masse.	`True / False`
`objectif_masse`	Objectif de masse à trouver par l’algorithme d’optimisation de la masse (g).	`Float`
`tolerance`	Tolérance sur l’objectif de la masse (+- x g).	`Float`
`nb_pas_max`	Nombre de pas de calcul maximum avant l’arrêt de l’algorithme d’optimisation de la masse même si le critère d’objectif n’est pas atteint.	`Integer`
`correction_ep_par_pas`	Valeur à niter=0 de correction de l’épaisseur des parois à chaque pas (mm).	`Float`
`pourcentage_modification_correction`	Pourcentage de modification de `correction_ep_par_pas` par pas de calcul (augmentation ou diminution).	`Float`
`seuil_augmentation_correction`	Seuil à partir duquel `correction_ep_par_pas` est modifié. Critère : - Augmentation : `abs(masse[pas] - masse[pas - 1]) <= seuil_augmentation_correction` - Diminution : `abs(masse[pas] - masse[pas - 1]) >= seuil_diminution_correction`	`Float`
`rho`	Masse volumique du matériau utilisé pour la structure.	`Float`

2. Partie Méthode de Génération

Nom de la Variable	Description	Valeurs Possibles
`gen_losange_basic`	Activer / Désactiver la génération de la structure Losange sans gradients.	`True / False`
`gen_losange_grad`	Activer / Désactiver la génération de la structure Losange avec gradients.	`True / False`
`gen_hex_tri1_2D_aligne_basic`	Activer / Désactiver la génération de la structure Hexagones + Triangles 2D Alignés sans gradients.	`True / False`
`gen_hex_tri1_2D_aligne_grad`	Activer / Désactiver la génération de la structure Hexagones + Triangles 2D Alignés avec gradients.	`True / False`
`gen_hex_tri1_2D_nalign e_basic`	Activer / Désactiver la génération de la structure Hexagones + Triangles 2D Non Alignés sans gradients.	`True / False`
`gen_hex_tri1_2D_nalign e_grad`	Activer / Désactiver la génération de la structure Hexagones + Triangles 2D Non Alignés avec gradients.	`True / False`
`gen_tri_2D_basic`	Activer / Désactiver la génération de la structure Triangles sans gradients.	`True / False`
`gen_tri_2D_grad`	Activer / Désactiver la génération de la structure Triangles avec gradients.	`True / False`
`gen_cos_2D_basic`	Activer / Désactiver la génération de la structure Cosinus sans gradients.	`True / False`
`gen_cos_2D_grad`	Activer / Désactiver la génération de la structure Cosinus avec gradients.	`True / False`

3. Partie Plateaux Liants les Extrémités de la Structure

Nom de la Variable	Description	Valeurs Possibles
`generation_plateaux_ex tremitees`	Activer / Désactiver la génération des plateaux liant les extrémités des structures.	`True / False`
`ep_plateau_dessous`	Épaisseur du plateau du dessous de la structure.	`Float`
`ep_plateau_dessus`	Épaisseur du plateau du dessus de la structure.	`Float`

4. Partie Géométrie Commune

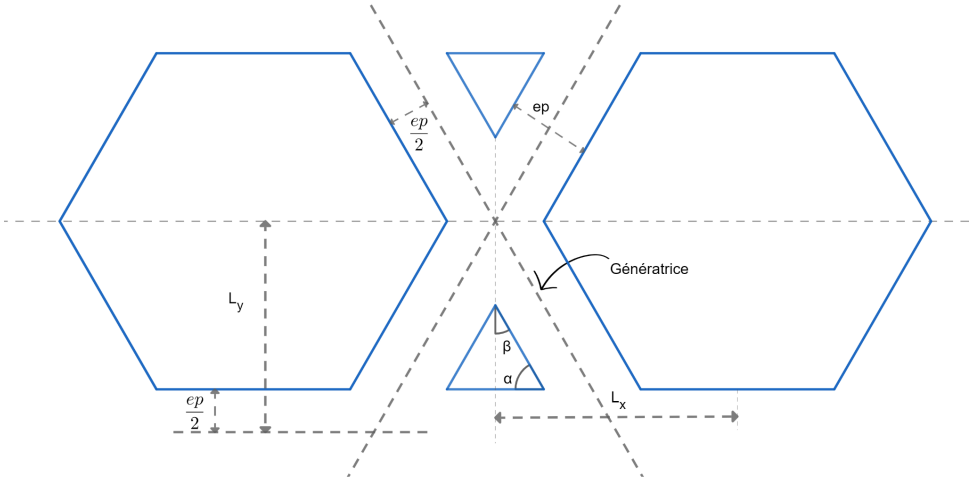
Nom de la Variable	Description	Valeurs Possibles
`ep`	Épaisseur des parois de la structure (mm).	`Float`
`dimlat_ep`	Dimension d'extrusion de la structure (mm).	`Float`
`dimlat_x`	Dimension de la structure dans la direction x (mm).	`Float`
`dimlat_y`	Dimension de la structure dans la direction y (mm).	`Float`
`nb_motif_x_sg`	Nombre de motifs élémentaires à répéter suivant l'axe x.	`Integer`
`nb_motif_y_sg`	Nombre de motifs élémentaires à répéter suivant l'axe y.	`Integer`

5. Partie Géométrie pour chaque Structures

Variables communes :

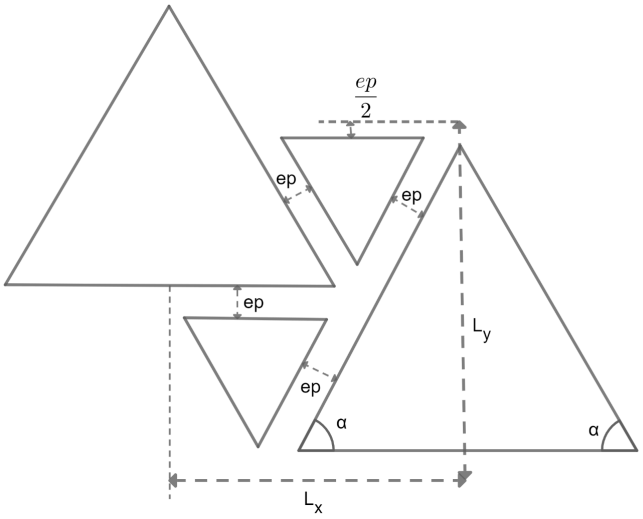
Nom de la Variable	Description	Valeurs Possibles
`nb_x_par_couche`	Nombre de motifs élémentaires à répéter suivant l'axe x par couches. Chaque couche est délimitée par une "`,`".	`List(Integer)`
`nb_y_par_couche`	Nombre de motifs élémentaires à répéter suivant l'axe y par couches. Chaque couche est délimitée par une "`,`".	`List(Integer)`
`dimlat_par_couche_manuel`	Cette option permet de renseigner manuellement la dimension suivant y de chaque couche. Si cette option est activée, la variable `dimlat_par_couche` doit-être renseignée et remplace la variable `dimlat_y`. Sinon, `dimlat_y` est considéré et chaque couche aura une épaisseur au prorata du nombre de motifs.	`True / False`
`dimlat_par_couche`	Dimensions suivant y de chaque couche de gradients.	`List(Float)`
`ep_par_couche`	Pourcentage de l'épaisseur de chaque couche par rapport à l'épaisseur nominale `ep`.	`Float`
`ep_plateaux`	Épaisseur des plateaux entre chaque couche de gradients (nécessairement positif ou nul) en mm.	`Float`

Structure Hexagones + Triangles 2D :



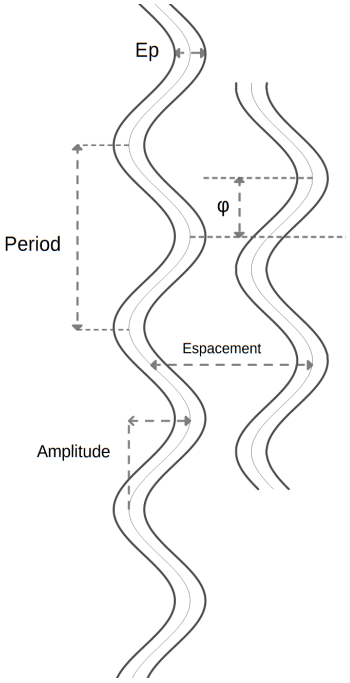
Nom de la Variable	Description	Valeurs Possibles
`alpha_hex_tri1_2D`	Angle (voir schéma).	`Float`
`alpha_hex_tri1_2D_grad`	Angle pour chaque couche de gradients (voir schéma).	`List(Float)`

Structure Triangles 2D :



Nom de la Variable	Description	Valeurs Possibles
`alpha_tri_2D`	Angle (voir schéma).	`Float`
`alpha_hexalpha_tri_2D_grad_tri1_2D_grad`	Angle pour chaque couche de gradients (voir schéma).	`List(Float)`

Structure Cosinus 2D :



- Sans Gradients :

Nom de la Variable	Description	Valeurs Possibles
`phi`	Angle (voir schéma).	`Float` ou `List(Float)`
`period_fact`	Période du cosinus (voir schéma).	`Float` ou `List(Float)`
`amp`	Amplitude du cosinus (voir schéma).	`Float` ou `List(Float)`
`nbpts_cos`	Nombre de points de discrétisation du cosinus sur la longueur `dimlat_y`.	`Integer`

- Avec Gradients :

Nom de la Variable	Description	Valeurs Possibles
`phi_grad`	Angle (voir schéma).	`List(Float)` ou `List(List(Float))`
`period_fact_grad`	Période du cosinus (voir schéma).	`List(Float)` ou `List(List(Float))`
`amp_grad`	Amplitude du cosinus (voir schéma).	`List(Float)` ou `List(List(Float))`
`nbpts_cos_grad`	Nombre de points de discrétisation du cosinus sur la longueur `dimlat_y`.	`List(Integer)`

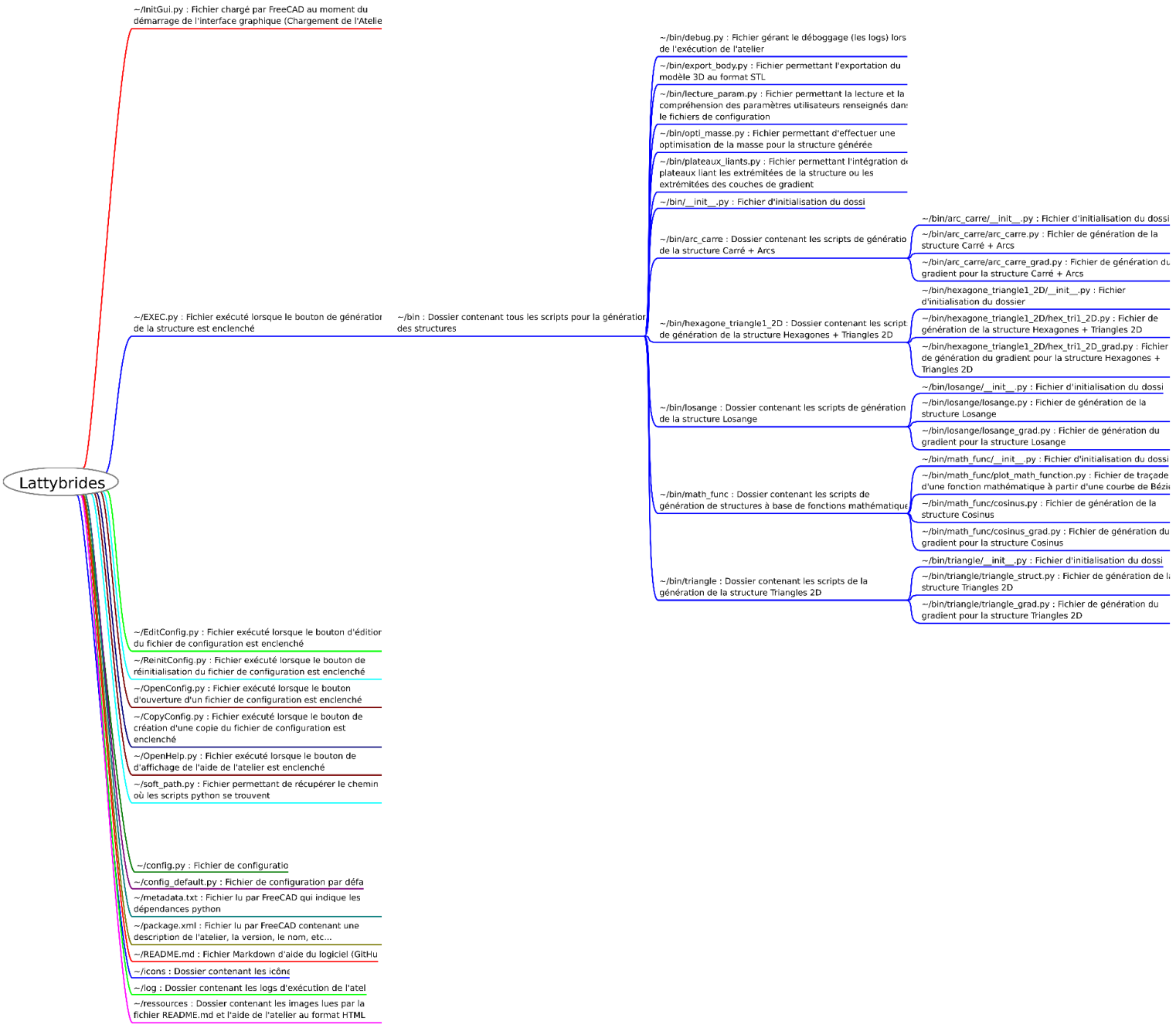
6. Partie Exportation du Modèle 3D

Nom de la Variable	Description	Valeurs Possibles
`extrude`	Activer / Désactiver l'extrusion de la structure.	`True / False`
`export`	Activer / Désactiver l'exportation du modèle 3D de la structure au format STL. Si `export=True` alors `extrude` doit être activé.	`True / False`
`enregistrement_fichier`	Activer / Désactiver l'enregistrement du fichier FreeCAD.	`True / False`
`sketch_visible`	Activer / Désactiver la visibilité des esquisses à la fin du calcul.	`True / False`

7. Partie Débogage

Nom de la Variable	Description	Valeurs Possibles
`semi_debug`	Activer / Désactiver l'e traçage des lignes de construction.	`True / False`
`debug`	Activer / Désactiver l'affichage des messages dans le terminal et l'écriture des messages dans le fichier log. Si cette option est activée alors `debug_current_folder` doit-être renseigné.	`True / False`
`debug_current_folder`	Dossier où doit-être stocké le fichier de log.	`String`

3. Architecture de l’Atelier



4. Développement de Nouvelles Structures

Étape 1 : Création d'une structure via une esquisse

Afin de rendre l'ajout de structures à l'atelier plus facile, chaque structure différente doit se trouver dans un script python différent. Pour ce faire, vous devez créer un nouveau dossier dans le dossier `~/bin` portant le nom de votre structure. Dans le dossier nouvellement créé, placez trois fichiers portant les noms suivants, `__init__.py`, `nom_structure.py`, `nom_structure_grad.py` (`nom_structure` doit-être remplacé par le nom de votre structure et ne doit pas comporter d'accents ou de caractères spéciaux).

Le fichier __init__.py sert à faire lire à python des modules présent dans le dossier et doit donc rester vierge. Dans le fichier `nom_structure.py`, vous devez créer une fonction prenant en argument tous les paramètres de votre structure ainsi que les arguments suivants obligatoires :

Nom de l'Argument	Description
`ep`	Épaisseur de la paroi
`doc`	Objet document FreeCAD
`file_debug`	Objet du fichier de débogage ouvert
`nb_motifs_x / nb_motifs_y`	Nombre de motifs élémentaires suivant l'axe x / y
`dimlat_x / dimlat_y / dimlat_ep`	Dimensions du volume occupé par la structure
`ep_plateaux = [épaisseur 1, épaisseur 2]`	Épaisseur des plateaux liants les extrémitées de la structure
`semi_debug = True / False`	True = Traçage des lignes de construction
`debug = True / False`	True = Écrire tous les messages de débogage dans le terminal et dans le fichier de log
`sketch_visible = True / False`	True = Laisser l'esquisse de la structure visible.
`extrude = True / False`	True = Extruder la structure à partir de l'esquisse
`nom_sketch_nom_structure`	Nom de l'esquisse de la structure
`nom_sketch_plateaux_extremitees = [nom1, nom2]`	Noms des plateaux liants les extrémités de la structure
`nom_body_nom_structure`	Nom du body de la structure
`nom_pad_nom_structure`	Nom du pad de la structure

`nom_pad_plateau_extremitees = [nom1, nom2]`	Noms des pad des plateaux liants les extrémités de la structure
`gen_plateaux`	Fonction de génération des plateaux
`generation_plateaux_extremitees = True / False`	True = Générer les plateaux liants les extrémités de la structure
`wdebug`	Fonction gérant le déboggage
`sketch`	Objet contenant l'esquisse de la structure

Globalement, le script dans `nom_structure.py` doit avoir cette trame :

```
def gen_nom_structure(ep=0.4,
                      doc=None,
                      file_debug=None,
                      nb_motifs_x=10,
                      nb_motifs_y=15,
                      dimlat_ep=5,
                      dimlat_x=20,
                      dimlat_y=20,
                      ep_plateaux=[1,1],
                      semi_debug=False,
                      debug=False,
                      sketch_visible=False,
                      extrude=True,
                      nom_sketch_nom_structure="Sketch_Nom_Structure",
                      nom_sketch_plateaux_extremitees=["Sketch_Plateau_Dessous",
"Sketch_Plateau_Dessus"],
                      nom_body_nom_structure="Body_Nom_Structure",
                      nom_pad_nom_structure="Pad_Nom_Structure",
                      nom_pad_plateau_extremitees=["Pad_Plateau_Dessous",
"Pad_Plateau_Dessus"],
                      gen_plateaux=None,
                      generation_plateaux_extremitees=True,
                      wdebug=None,
                      sketch=""):

    # Importation des modules externes
    import FreeCAD as App
    import Part, Sketcher

    if doc == None:    doc = FreeCAD.newDocument()


    if file_debug != None and debug: wdebug("""dimlat_x:{0}
                                               \ndimlat_y:{1}
                                                    \ndimlat_ep:{2}
                                          \nnb_motifs_x:{3}
                                          \nnb_motifs_y:{4}
                                          \nnom_sketch_nom_structurs:{5}
                                          \n----\n""".format(dimlat_x,
                                                             dimlat_y,
                                                             dimlat_ep,
                                                             nb_motifs_x,
                                                             nb_motifs_y,
                                                             nom_sketch_nom_structure)
                                                             file_debug)









    # Placez ici les dimensions de la structure à calculer
```

```
    lx = dimlat_x / nb_losange_x
    ly = dimlat_y / nb_losange_y
    if file_debug != None and debug: wdebug("""ep:{0}
                                            \nlx:{1}
                                            \nly:{2}
                                            \n----\n""".format(    ep,
                                            lx,
                                            ly),
                                            file_debug)

    """
    ------------------------
    --- Modélisation 2D ---
    ------------------------
    """
    # Création d'une nouvelle esquisse et de la pièce
    if sketch == "":
        if file_debug != None and debug:
                wdebug("Création de l'esquisse : {0}\n".format(nom_sketch_nom_structure),
file_debug)
                wdebug("Création du body : {0}\n".format(nom_body_nom_structure),
file_debug)

        sketch = doc.addObject("Sketcher::SketchObject", nom_sketch_nom_structure)
        doc.addObject('PartDesign::Body', nom_body_nom_structure)

    # Construction du rectangle de délimitation de la structure
    #    Points de délimitation du quadrilatère (dans le sens anti-horaire)
    point_delimitation = (    App.Vector(0, 0, 0),
                              App.Vector(dimlat_x, 0, 0),
                              App.Vector(dimlat_x, dimlat_y, 0),
                              App.Vector(0, dimlat_y, 0))
    #    Construction du quadrilatère si le mode semi_debug est activé
    if semi_debug:
        for i in range(1, 5):
                sketch.addGeometry(Part.LineSegment(point_delimitation[(i - 1) % 4],
point_delimitation[i % 4]), True)
                if file_debug != None and debug:
                        wdebug("\n\n  Construction du rectangle de délimitation de la
structure\n", file_debug)
                        wdebug("Construction de la ligne entre ({0}, {1}, {2}) et ({3},
{4}, {5})\n".format(point_delimitation[(i - 1) % 4].x,
                               point_delimitation[(i - 1) % 4].y,
                point_delimitation[(i - 1) % 4].z,
                point_delimitation[i % 4].x,
                point_delimitation[i % 4].y,
                point_delimitation[i % 4].z),
                file_debug)
                wdebug("\n", file_debug)

    # Curseur de position (repère local à chaque losange)
    current_pos = (0,0,0)

    # i = Numéro de losange y (ligne)
    # j = Numéro de losange x (colonne)
    for j in range(nb_motifs_y * 2):
        for i in range(nb_motifs_x):
                current_pos = (lx * i, (ly / 2) * j, 0)

                # Placez ici le script de génération de l'esquisse
```

```
    # Extrusion de l'esquisse & Génération des plateaux
    if extrude:
        # Créer un Pad
        pad_nom_structure =
doc.getObject(nom_body_nom_structure).newObject('PartDesign::Pad',
nom_pad_nom_structure)
        pad_nom_structure.Profile = sketch
        # Mettre l'esquisse dans le pad
        pad_nom_structure.Length = dimlat_ep
        # Définir la longueur d'extrustion
        pad_nom_structure.ReferenceAxis = (sketch, ['N_Axis'])
        # Définir la direction d'extrusion
        doc.recompute()
        # Lancer les calculs
        sketch.Visibility = sketch_visible


        # Affichage de l'esquisse après l'extrusion
        if file_debug != None and debug:
               wdebug("Extrusion de la structure\n", file_debug)

        if generation_plateaux_extremitees:
               # Génération des plateaux liants les extrémités
               if file_debug != None and debug:
                      wdebug("Création des plateaux liants les extrémités de la
structure.\n", file_debug)

               gen_plateaux(nb_couches=1,
                            ep_plateaux=ep_plateaux,
                            dimlat_x=dimlat_x,
                            dimlat_par_couche=[dimlat_y],
                            dimlat_ep=dimlat_ep,
                            sketch_visible=sketch_visible,
                            nom_body=nom_body_nom_structure,
                            doc=doc,
                            nom_sketch_plateaux=nom_sketch_plateaux_extremitees,
                            nom_pad_plateaux=nom_pad_plateau_extremitees,
                            debug=debug,
                            file_debug=file_debug,
                            wdebug=wdebug)
```

Étape 2 : Création de la fonction gradient

Dans le fichier `nom_structure_grad.py` vous devez créer une fonction qui vient appeler la fonction de génération de la structure une fois par couche de gradient.

Globalement, le script dans `nom_structure_grad.py` doit avoir cette trame :

```
def nom_structure_grad(ep=0.4,
                       doc=None,
                       file_debug=None,
                       nb_couches=3,
                       nb_motifs_par_couche=[2,3,2],
                       dimlat_par_couche=[40/7*2,40/7*3,40/7*2],
                       ep_par_couche=[1,0.5,0.5,1],
nom_sketch_par_couche=["Sketch_Nom_Structure1","Sketch_Nom_Structure2","Sketch_Nom_Struc
ture3"],
nom_pad_par_couche=["Pad_Nom_Structure1","Pad_Nom_Structure2","Pad_Nom_Structure3"],
                       dimlat_x=40,
                       dimlat_ep=40,
                       nb_motifs_x=[7,4,7],
nom_sketch_plateaux=["Sketch_Plateaux1","Sketch_Plateaux2","Sketch_Plateaux3","Sketch_Pl
ateaux4"],
nom_pad_plateaux=["Pad_Plateaux1","Pad_Plateaux2","Pad_Plateaux3","Pad_Plateaux4"],
                       nom_body_nom_structure="Body_Nom_Structure",
                       ep_plateaux=[1,0.5,0.5,1],
                       gen_plateaux=None,
                       gen_structure=None,
                       sketch_visible=False,
                       extrude=True,
                       semi_debug=False,
                       debug=False,
                       wdebug=None):

    # Importation des modules externes
    import FreeCAD as App
    import Part, Sketcher

    if doc == None: FreeCAD.newDocument()  # Création du document
    posy = 0   # Position y de l'origine des esquisses à créer
    sketches = []     # Liste contenant toutes les esquisses

    if file_debug != None and debug:
        wdebug("Création du body : {0}\n".format(nom_body_nom_structure), file_debug)

    body = doc.addObject('PartDesign::Body', nom_body_nom_structure)    # Créer un body

    for no_couche in range(nb_couches):
        # Création de l'esquisse de la couche
        if file_debug != None and debug:
                wdebug("Création de l'esquisse pour la couche {1}:
{0}\n".format(nom_sketch_par_couche[no_couche], no_couche), file_debug)

        sketches.append(doc.addObject("Sketcher::SketchObject",
nom_sketch_par_couche[no_couche]))
        sketches[no_couche].Placement = App.Placement(App.Vector(0, posy, 0),
App.Rotation(0, 0, 0, 1))

        if file_debug != None and debug:
                wdebug("""Génération de la structure pour la couche {0}:\n      posy =
{1}\n""".format(    no_couche,


posy), file_debug)
```

```
        # Génération de la structure sur le couche no_couche
        gen_structure(    ep=ep * ep_par_couche[no_couche],
                          doc=doc,
                          file_debug=file_debug,
                          nb_motifs_x=nb_motifs_x[no_couche],
                          nb_motifs_y=nb_motifs_par_couche[no_couche],
                          dimlat_ep=dimlat_ep,
                          dimlat_x=dimlat_x,
                          dimlat_y=dimlat_par_couche[no_couche],
                          ep_plateaux=[0, 0],
                          semi_debug=semi_debug,
                          debug=debug,
                          sketch_visible=sketch_visible,
                          extrude=False,
                      nom_sketch_nom_structure=nom_sketch_par_couche[no_couche],
                      nom_sketch_plateaux_extremitees=nom_sketch_plateaux[no_couche],
                      nom_body_nom_structure=nom_body_nom_structure,
                      nom_pad_nom_structure=nom_pad_par_couche[no_couche],
                      nom_pad_plateau_extremitees=nom_pad_plateaux[no_couche],
                          gen_plateaux=None,
                          generation_plateaux_extremitees=False,
                          wdebug=wdebug,
                          sketch=sketches[no_couche])

        # Incrément de la position y dans le repère
        posy += dimlat_par_couche[no_couche]

    # Extrusion des l'esquisses & Génération des plateaux
    if extrude:
        # Extrusion de chaque couches
        for no_couche in range(nb_couches):
               body.newObject('PartDesign::Pad', nom_pad_par_couche[no_couche])
                                                  # Créer un Pad
               doc.getObject(nom_pad_par_couche[no_couche]).Profile =
sketches[no_couche]                                                  # Mettre l'esquisse
dans le pad
               doc.getObject(nom_pad_par_couche[no_couche]).Length = dimlat_ep
                                                        # Définir la longueur
d'extrustion
               doc.getObject(nom_pad_par_couche[no_couche]).ReferenceAxis =
(sketches[no_couche], ['N_Axis'])    # Définir la direction d'extrusion
               doc.recompute()

        # Lancer les calculs
               sketches[no_couche].Visibility = sketch_visible
                                                                              # Affichage de
l'esquisse après l'extrusion

               if file_debug != None and debug:
                      wdebug("Extrusion de la structure : Couche no
{0}\n".format(no_couche), file_debug)

        # Génération des plateaux liants les couches de la structure
        if file_debug != None and debug:
               wdebug("Création des plateaux liants les couches de la structure.\n",
file_debug)

        gen_plateaux(nb_couches=nb_couches,
                      ep_plateaux=ep_plateaux,
                      dimlat_x=dimlat_x,
                      dimlat_par_couche=dimlat_par_couche,
                      dimlat_ep=dimlat_ep,
                      sketch_visible=sketch_visible,
                      nom_body=nom_body_nom_structure,
                      doc=doc,
                      nom_sketch_plateaux=nom_sketch_plateaux,
                      nom_pad_plateaux=nom_pad_plateaux,
                      debug=debug,
                      file_debug=file_debug,
                      wdebug=wdebug)
```

Note : Il est possible que pour certaines structures (comme la structure Cosinus) ce script ne soit pas valable. Il doit donc être développé au cas pas cas.

Étape 3 : Ajout des paramètres dans le fichier de configuration.

Au minimum deux paramètres sont obligatoires dans le fichier de configuration. Ce sont les paramètres qui permettent à l'atelier de savoir quelle fonction de génération est utilisée.

Ces paramètres doivent être renseignés de la façon suivante :

```
gen_nom_structure_basic:False:
gen_nom_structure_grad:False:
```

Lorsque la valeur est à True, la fonction de génération est paramétrée comme active et sera utilisée pour le calcul de la structure. Une fois ces deux variables ajoutées, il va falloir modifier le fichier `~/bin/lecture_params.py` pour qu'il lise ces paramètres.

1) Ligne 30 : Modifier le nombre d'éléments de la liste `return_nok`. Il faut ajouter autant de nombre d'éléments que de variables ajoutées au fichier de configuration c'est à dire minimum 2.

```
return_nok = [False for i in range(53)]
```

2) Il faut ensuite ajouter les variables avec une valeur None dans le code python. Exemple :

```
# …
gen_cos_2D_basic = None
gen_cos_2D_grad = None
gen_nom_structure_basic = None
gen_nom_structure_grad = None
nom_structure_param1 = None
# Etc…
```

3) Il faut également ajouter +2 au nombre d'éléments de la liste `gen_func` au début du fichier.

```
gen_func = [None for i in range(10)]
```

4) Ajout de la lecture du paramètre gen_nom_structure_basic/grad dans la boucle de traitement des données.

```
elif lignes[i][0] == "gen_nom_structure_basic":
                if lignes[i][1] == "False":
                        gen_nom_structure_basic = False
                        gen_func[10] = False
                elif lignes[i][1] == "True":
                        gen_nom_structure_basic = True
                        gen_func[10] = True
                else:
                        if debug:
                                log += "lecture_param\nCommande inconnue pour
gen_nom_structure_basic\n"
```

5) Ajout de la lecture des paramètres de la structure dans la boucle de traitement des données.

```
# Exemple pour un paramètre booléen
elif lignes[i][0] == "nom_structure_param1":
                if lignes[i][1] == "False":          nom_structure_param1 = False
                elif lignes[i][1] == "True":    nom_structure_param1 = True
                else:
                        if debug:
                                log += "lecture_param\nCommande inconnue pour
nom_structure_param1\n"

# Exemple pour un paramètre à virgule flottante
elif lignes[i][0] == "nom_structure_param2":
                try:
                        nom_structure_param2 = float(lignes[i][1])
                except:
                        if debug:
                                log += """    lecture_param\nLe type de données entrée
dans nom_structure_param2 n'est pas correct ! \n
nom_structure_param2={0}\n""".format(lignes[i][1])

# Exemple pour un paramètre entier
elif lignes[i][0] == "nom_structure_param3":
                try:
                        nom_structure_param3 = int(lignes[i][1])
                except:
                        if debug:
                                log += """    lecture_param\nLe type de données entrée
dans nom_structure_param3 n'est pas correct ! \n
nom_structure_param3={0}\n""".format(lignes[i][1])

# Exemple pour un paramètre sous forme d'une liste avec la , comme séparateur
elif lignes[i][0] == "nom_structure_param4":
                try:
                        nom_structure_param4 = [int(lignes[i][1].split(',')[j]) for j in
range(len(lignes[i][1].split(',')))]
                except:
                        if debug:
                                log += """    lecture_param\nLe type de données entrée
dans nom_structure_param4 n'est pas correct !\n
nom_structure_param4={0}\n""".format(lignes[i][1])
```

Note : D'autres types de données ou des listes à doubles entrées (voir plus) sont possibles. Inspirez-vous de ce format de récupération de données.

6) Ajouter les paramètres dans la liste `return_ok`.

```
return_ok = [                   True,
                                …
                                gen_nom_structure_basic,
                                gen_nom_structure_grad,
                                nom_structure_param1,
                                …
                                debug_current_folder]
```

7) Ajout de la vérification de remplissage des paramètres par l'utilisateur.

```
if gen_nom_structure_basic:
        if nom_structure_param1 == None:
                if debug:
                        log += "lecture_param\nnom_structure_param1 n'est pas
définie !\n"
                return_nok.append(log)
                return return_nok
        # Etc …

if gen_nom_structure_grad:
        if nom_structure_param1 == None:
                if debug:
                        log += "lecture_param\nnom_structure_param1 n'est pas
définie !\n"
                return_nok.append(log)
                return return_nok
        # Etc …
```

8) Ajout des variables réceptrices dans `~/EXEC.py`.

```
# Lecture des parmaètres du programme
[       lecture_param_ok,
        …
        gen_nom_structure_basic,
        gen_nom_structure_grad,
        nom_structure_param1,
        …
        log] = lecture_param(softpath + "/config.py")
```

Étape 4 : Ajout de l'exécution des fonctions de génération dans le fichier `~/EXEC.py`.

Dans la partie génération des structures sans gradients :

```
elif gen_nom_structure_basic:
    nom_body = "Body_Nom_Structure"
    if optimisation_masse:
        masse, pas_final, ep_finale, porosite = opti_masse(
                        doc,
                        nom_body,
                        "Pad_Losange",
                        ["Pad_Plateau_Dessous", "Pad_Plateau_Dessus"],
                        "Sketch_Losange",
                        ["Sketch_Plateau_Dessous", "Sketch_Plateau_Dessus"],
                        gen_losange,
                        file_debug,
                        wdebug,
                        debug,
                        tolerance,
                        nb_pas_max,
                        [0 for i in range(nb_pas_max + 1)],
                        ep,
                        0,
                        correction_ep_par_pas,
                        pourcentage_modification_correction,
                        seuil_augmentation_correction,
                        seuil_diminution_correction,
                        objectif_masse,
                        rho,
                        volume_max,
                        PARAMETRES DE LA FONCTION DE GENERATION)

    else:
        gen_losange(PARAMETRES DE LA FONCTION DE GENERATION)
```

Dans la partie génération de structures avec gradients :

```
elif gen_losange_grad or gen_hex_tri1_2D_aligne_grad or gen_hex_tri1_2D_naligne_grad or
gen_tri_2D_grad or gen_cos_2D_grad or gen_nom_structure_grad:
    # Etc…
    elif gen_nom_structure_grad:
        nom_body = "Body_Nom_Structure"
        if optimisation_masse:
                masse, pas_final, ep_finale, porosite = opti_masse(
                                doc,
                                nom_body,
                                nom_pad_par_couche,
                                nom_pad_plateaux,
                                nom_sketch_par_couche,
                                nom_sketch_plateaux,
                                losange_grad,
                                file_debug,
                                wdebug,
                                debug,
                                tolerance,
                                nb_pas_max,
                                [0 for i in range(nb_pas_max + 1)],
                                ep,
                                0,
                                correction_ep_par_pas,
                                pourcentage_modification_correction,
                                seuil_augmentation_correction,
                                seuil_diminution_correction,
                                objectif_masse,
                                rho,
                                volume_max,
                               PARAMETRES DE LA FONCTION DE GENERATION)
        else:
                losange_grad(PARAMETRES DE LA FONCTION DE GENERATION)
```