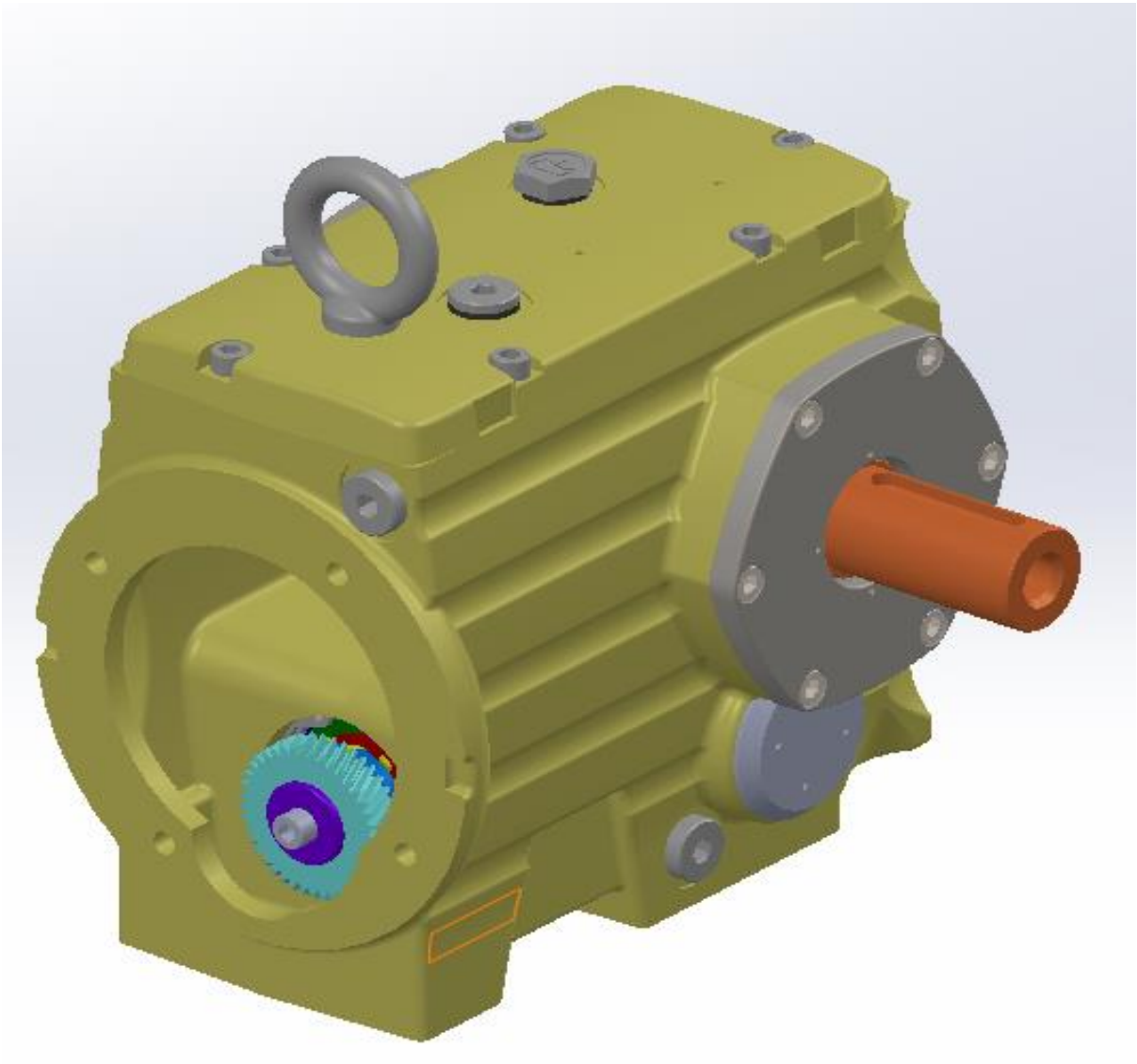
Projet SW : Conception d’un réducteur



SOMMAIRE :

I - Présentation

II - Etude Statique

1. Statiques des engrenages
2. Statique
3. La Statique sur Python

III - Calcul de durée de vie

1. Roulements rigides à une rangée de billes
2. Roulements à rouleaux coniques
3. Affichage sur python des calculs et choix des roulements

IV - Solidworks

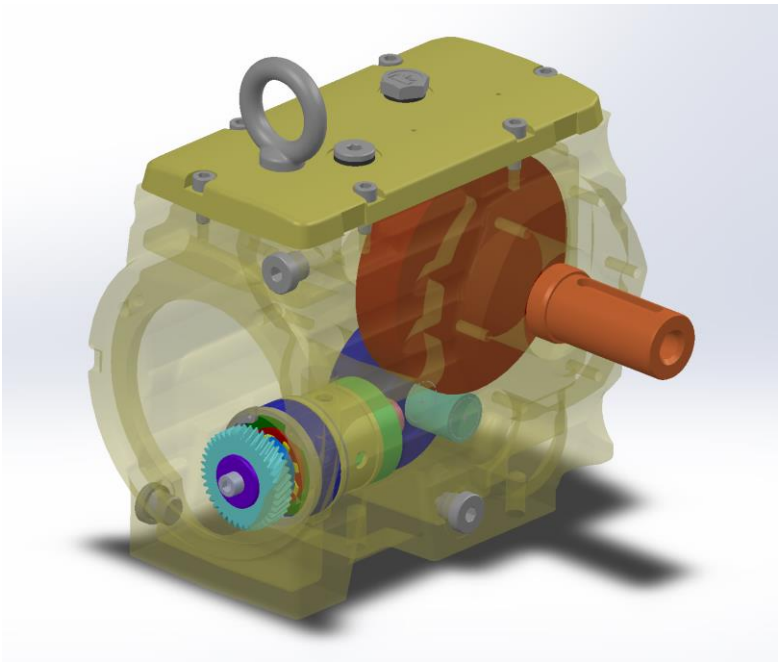
1. Choix du montage sur l'arbre 41-53
2. Choix du montage sur l'arbre 31-51
3. Mise en plan et éclaté du mécanisme

V - Conclusion

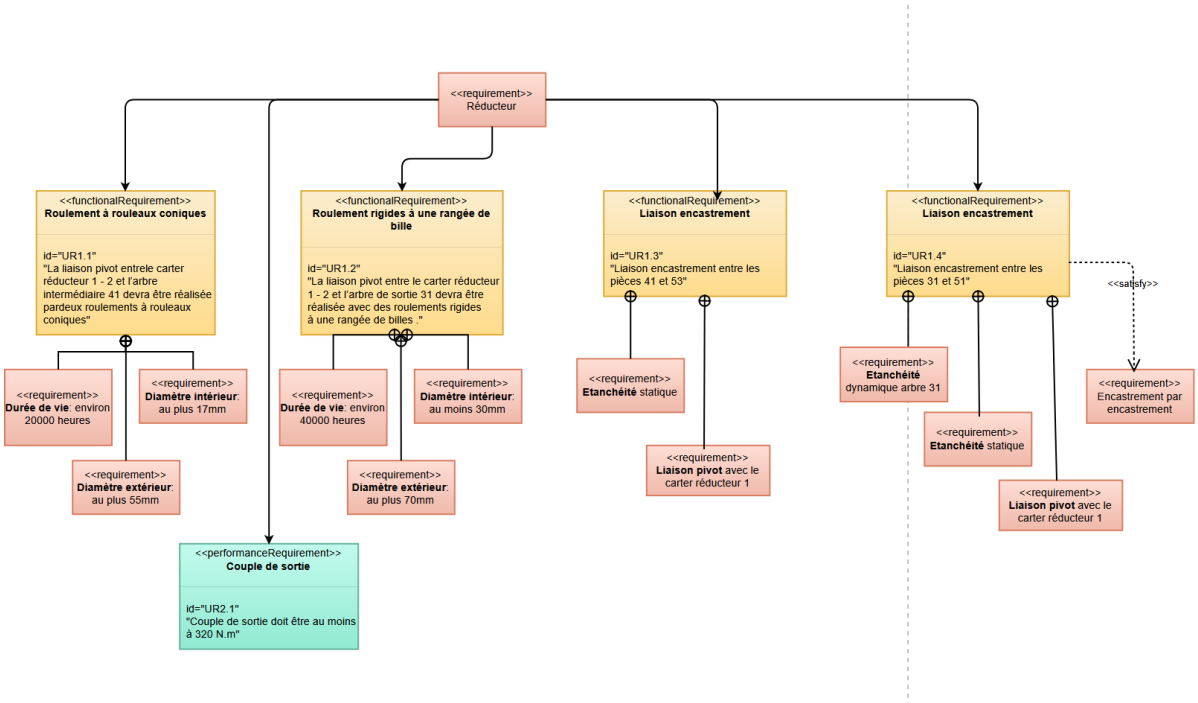
VI - Annexe et références bibliographiques

I - Présentation

L'objectif de projet est de concevoir une nouvelle gamme de réducteur à engrenages qui sera nommé FLORIC pour la société BO-RDEAU. A partir de l'étude préliminaire imposée, je doit construire un cahier des charges afin de réaliser la conception de ce réducteur.



La cahier est exprimé sur le diagramme d'exigence ci-contre:



II - Etude Statique

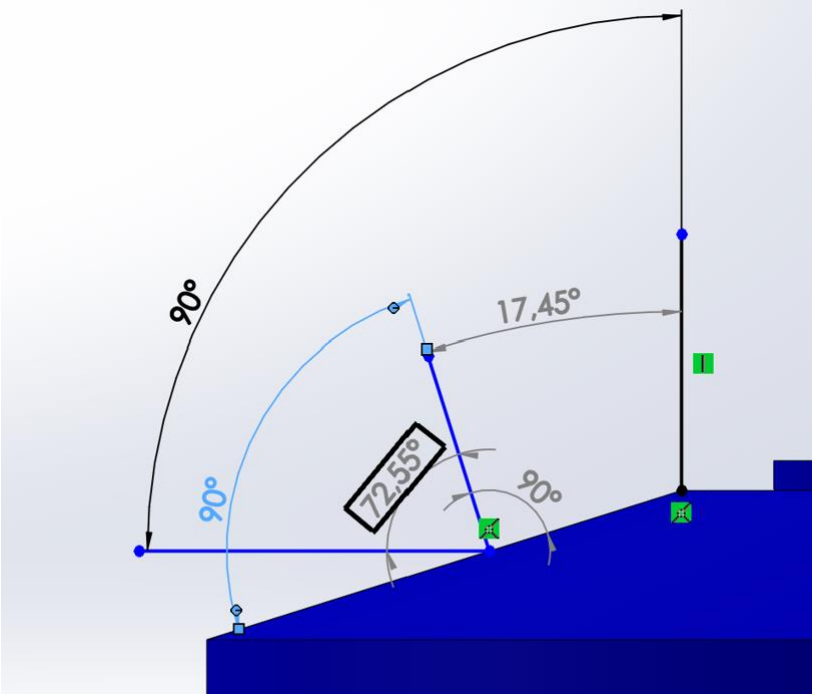
1. Statiques des engrenages

Pour déterminer les différents efforts dans les différents engrenages, j'ai tout d'abord calculé la norme (valeur numérique ou expression littérale) et ensuite déterminé son sens (sur quelle axe agis t'elle).

Pour les calculs des efforts, j'ai utilisé les formules d'efforts sur les dentures. Certains angles et rayons ne sont pas indiqués dans le sujet, on les mesure donc sur SolidWorks.

- Engrenages coniques à denture droite

  On mesure l'angle delta (δ) sur SolidWorks (voir image ci-contre) qui vaut 72.55°.



  Il faut également calculer les couples transmis sur cette arbre et les vitesses de rotations sur serons nécessaire dans la suite de l'étude (déterminer à partir des rapports de réduction).
  N4153 = N * (Z43/Z53)
  C4153 = C3151 * (Z41/Z51)

  Les efforts dans les engrenages sont :

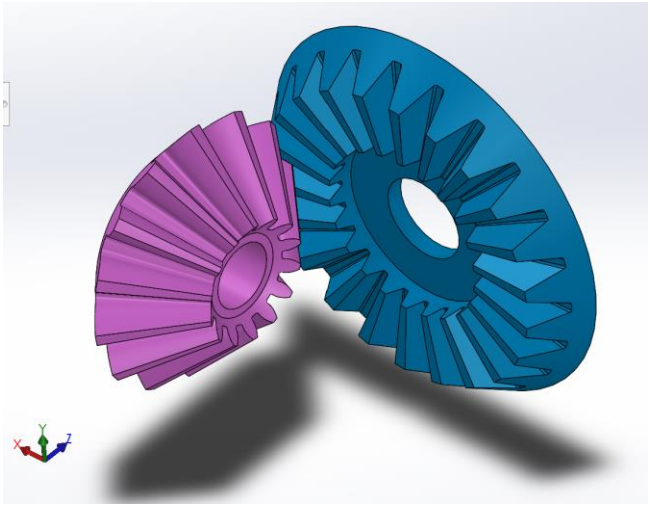
$$F_{T\_conique} = \frac{C_{4153}}{r_{m53}}$$

$$F_{A\_conique} = F_{T\_conique} \tan \propto_{53} \sin \delta$$

$$F_{R\_conique} = F_{T\_conique} \tan \propto_{53} \cos \delta$$

  Pour déterminer le sens des efforts et sur quelle axe ce dernier s'applique, une modélisation 3D des pignons sur SolidWorks peut m'aider. Je vais par conséquent

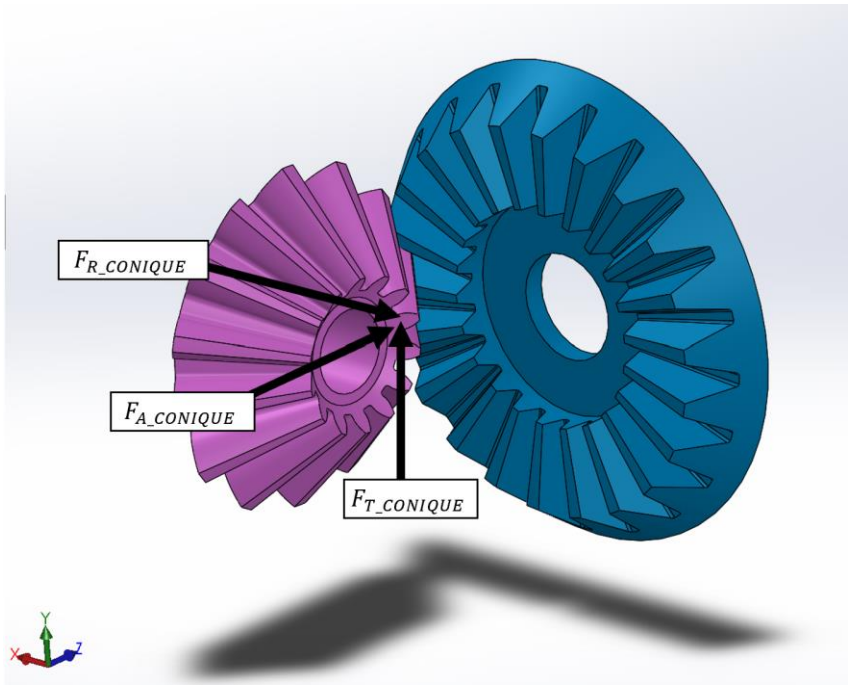
utiliser un fichier avec 2 engrenages à denture droite et raisonner avec les sens de rotation (avec les axes bien placés).



L'engrenage bleu correspond à la roue 53 et l'engrenage rose au pignon 43. (Les diamètres ne sont pas représentatifs)
En effet, l'arbre 41-53 tourne dans le sens trigo (anti-horaire), il en vient ainsi les efforts qu'exerce le pignon **43 sur 53** :

$$-\mathrm{F}_{\mathrm{R\_conique}}\vec{x}$$
$$\mathrm{F}_{\mathrm{T\_conique}}\vec{y}$$
$$\mathrm{F}_{\mathrm{A\_conique}}\vec{z}$$

(Voir image ci-contre pour les sens)

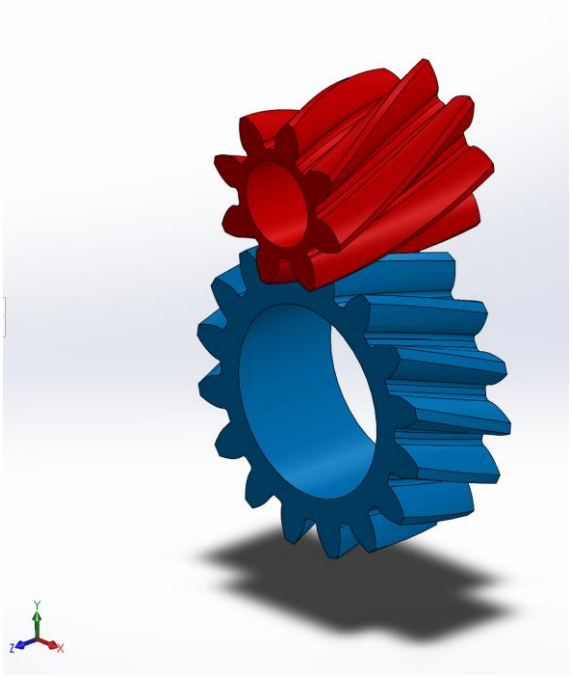


- Engrenages à denture hélicoïdales

  Les expressions des efforts sont :

  $$F_T = \frac{C_s}{r_{51}}$$
  $$F_A = F_T \tan\beta$$
  $$F_R = F_T \frac{\tan \propto}{\cos\beta}$$

  On raisonne de la même manière pour obtenir les sens, on utilise un fichier contenant deux engrenages hélicoïdaux (avec des diamètres non représentatifs)

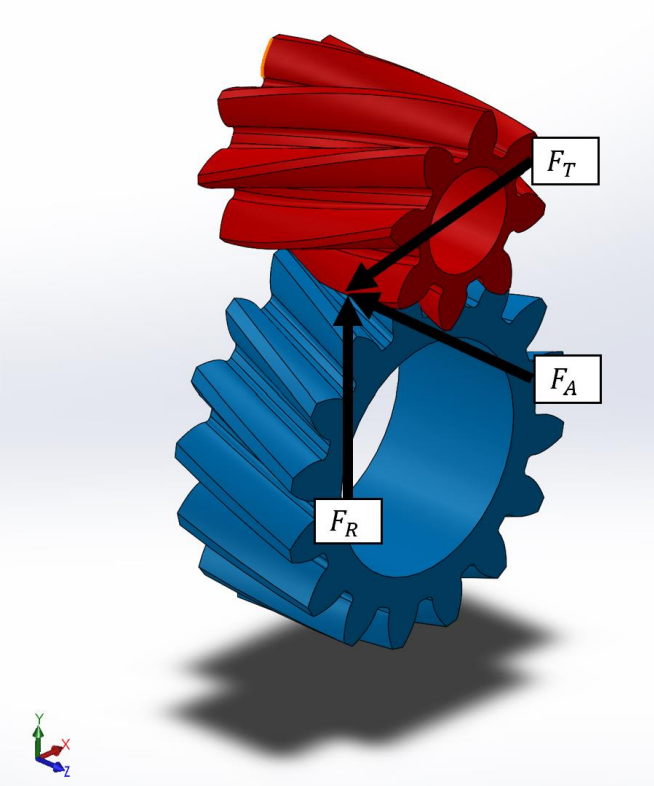


L'engrenage bleu correspond au pignon 41 et l'engrenage rouge à la roue 51.

En effet, l'arbre 51-31 tourne dans le sens horaire, il en vient ainsi les efforts qu'exerce le pignon **41 sur 51** (c'est-à-dire les efforts de l'engrenages 41 lorsqu'on isole l'arbre 31-51) :

$$-F_T\overrightarrow{x_1}$$
$$F_R\overrightarrow{y_1}$$
$$-F_A\overrightarrow{z_1}$$

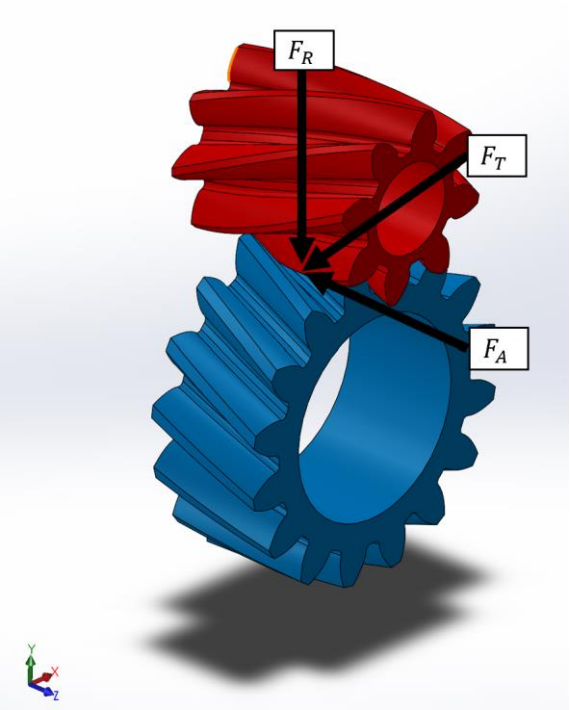
Voir image ci-contre



De même, il en vient ainsi les efforts qu'exerce le pignon **51 sur 41** (c'est-à-dire les efforts de l'engrenages 51 lorsqu'on isole l'arbre 41-53) :

$$-\mathrm{F_T}\overrightarrow{x_1}$$
$$-\mathrm{F_R}\overrightarrow{y_1}$$
$$-\mathrm{F_A}\overrightarrow{z_1}$$

Voir image ci-contre :



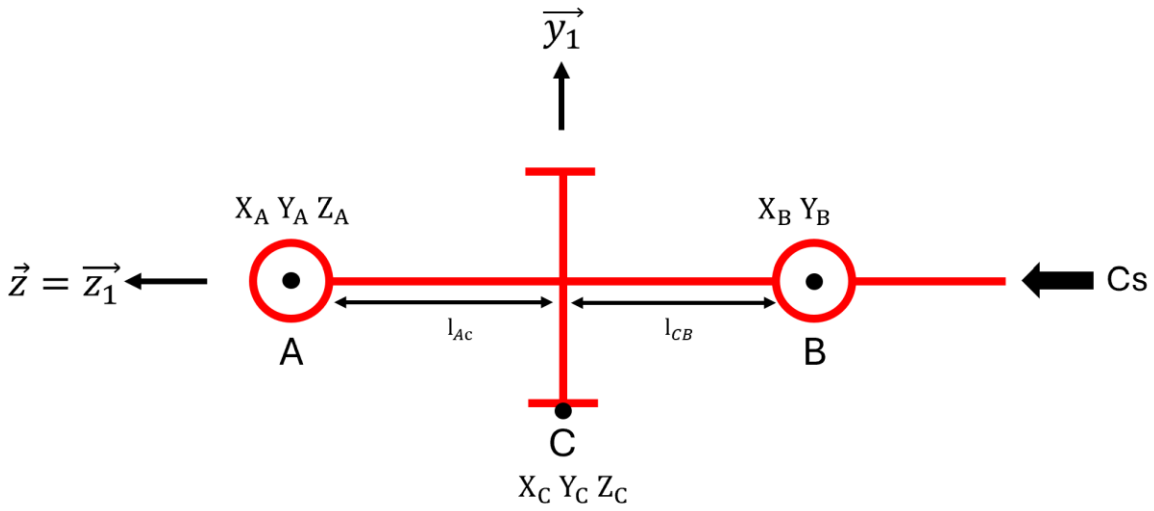
2. <u>Statique</u>

Hypothèse :

- Liaisons parfaites
- Poids négligeables

<u>Statique Arbre 51-31:</u>

On peut bloquer le roulement par un couvercle en A assez facilement, donc on effectuer 4 arrêts axiaux. Cela correspond donc à une liaison rotule en A. On verra par la suite si cette hypothèse est fonctionnelle.



D'après les calculs d'efforts précédents, $X_c = -F_t, Y_C = F_r \; et \; Z_C = -F_A$

On déplace les moments en A

$$\overrightarrow{M_{A,41\to51}} = \overrightarrow{M_{C,41\to51}} + \overrightarrow{AC} \wedge \overrightarrow{F_{41\to51}}$$

$$= \vec{0} + (-l_{ac}\overrightarrow{z_1} - r_{51}\overrightarrow{y_1} \wedge (-F_t\overrightarrow{x_1} + F_r\overrightarrow{y_1} - F_A\overrightarrow{z_1})$$

$$= l_{ac}F_t\overrightarrow{y_1} + l_{ac}F_r\overrightarrow{x_1} - r_{51}F_t\overrightarrow{z_1} + r_{51}F_A\overrightarrow{x_1}$$

$$\overrightarrow{M_{A,1\to51}} = \overrightarrow{M_{B,1\to51}} + \overrightarrow{AB} \wedge \overrightarrow{F_{1\to51}}$$

$$= \vec{0} + (-(l_{ac} + l_{cb})\overrightarrow{z_1}) \wedge (X_B\overrightarrow{x_1} + Y_B\overrightarrow{y_1})$$

$$= -(l_{ac} + l_{cb})X_B\overrightarrow{y_1} + (l_{ac} + l_{cb})Y_B\overrightarrow{x_1}$$

TMS en A:

$/\overrightarrow{x_1} \quad l_{ac}F_r + r_{51}F_A + (l_{ac} + l_{cb})Y_B = 0$

$/\overrightarrow{y_1} \quad l_{ac}F_t - (l_{ac} + l_{cb})X_B = 0$

$/\overrightarrow{z_1} \quad -C_S - r_{51}F_t = 0$

TRS

$X_A + X_B - F_T = 0$

$Y_A + Y_B + F_R = 0$

$Z_A - F_A = 0$

Donc d'après les équations précédentes, on a:

$$\mathbf{F_t = \frac{-C_S}{r_{51}}}$$

$$\mathbf{X_B = \frac{l_{ac}F_t}{(l_{ac} + l_{cb})}}$$

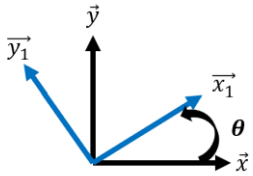
$$\mathbf{Y_B = \frac{l_{ac}F_r + r_{51}F_A}{-(l_{ac} + l_{cb})}}$$

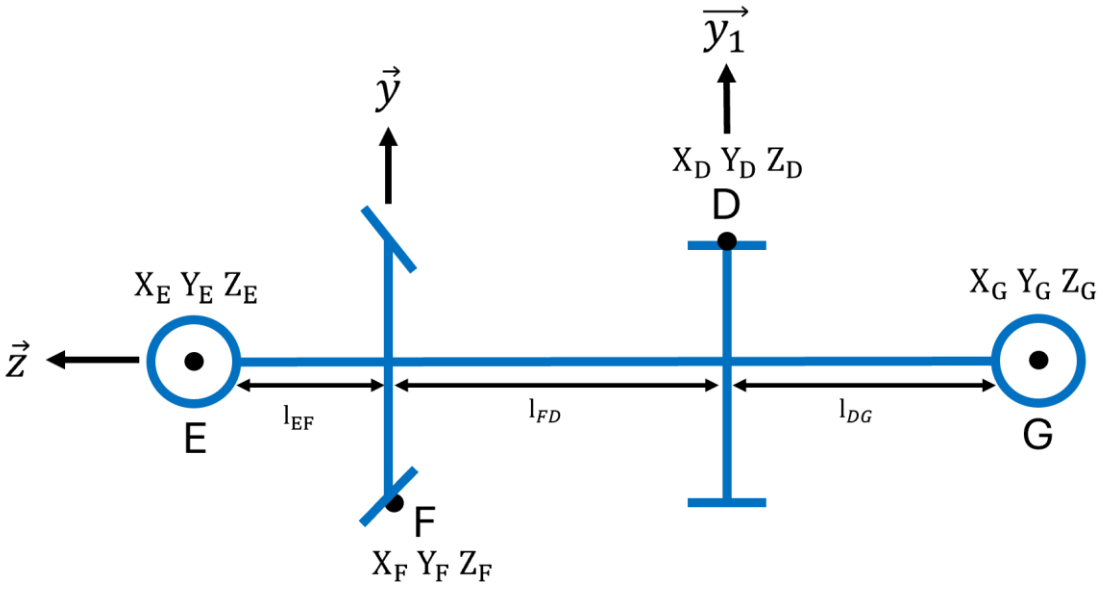
$$\mathbf{X_A = F_T - X_B}$$

$$\mathbf{Y_A = -Y_B - F_R}$$

<u>Statique Arbre 41-53:</u>

On remarque qu'on va avoir deux repères séparés d'un angle theta mesurée sur SolidWorks qui vaut 8.71°





Avec $X_F = -F_{R\_conique}$; $Y_F = F_{T\_conique}$; $Z_F = F_{A\_conique}$; $X_D = -F_T$; $Y_D = -F_R$; $Z_D = -F_A$

Les roulements à rouleaux coniques sont représentés par demi-rotule

$$\overrightarrow{M_{E,1\to41}} = \overrightarrow{M_{G,1\to41}} + \overrightarrow{EG} \wedge \overrightarrow{F_{1\to41}}$$

$$= \vec{0} - (l_{EF} + l_{FD} + l_{DG})\vec{z} \wedge (X_G\vec{x} + Y_G\vec{y} + Z_G\vec{z})$$

$$= -(l_{EF} + l_{FD} + l_{DG})X_G\vec{y} + (l_{EF} + l_{FD} + l_{DG})Y_G\vec{x}$$

$$\overrightarrow{M_{E,43\to53}} = \overrightarrow{M_{F,43\to53}} + \overrightarrow{EF} \wedge \overrightarrow{F_{43\to53}}$$

$$= \vec{0} + (-l_{EF}\vec{z} + r_{m53}\vec{x}) \wedge \left(-F_{R\_conique}\vec{x} + F_{T\_conique}\vec{y} + F_{A\_conique}\vec{z}\right)$$

$$= l_{EF}F_{R\_conique}\vec{y} + l_{EF}F_{T\_conique}\vec{x} + r_{m53}F_{T\_conique}\vec{z} - r_{m53}F_{A\_conique}\vec{y}$$

$$\overrightarrow{M_{E,51\to41}} = \overrightarrow{M_{D,51\to41}} + \overrightarrow{ED} \wedge \overrightarrow{F_{51\to41}}$$

$$= \vec{0} + (-(l_{EF} + l_{FD})\vec{z} + r_{41}\overrightarrow{y_1}) \wedge (-F_T\overrightarrow{x_1} - F_R\overrightarrow{y_1} - F_A\vec{z})$$

$$= (l_{EF} + l_{FD})F_T\cos\theta\,\vec{y} - (l_{EF} + l_{FD})F_T\sin\theta\,\vec{x} - (l_{EF} + l_{FD})F_R\sin\theta\,\vec{y}$$
$$- (l_{EF} + l_{FD})F_R\cos\theta\,\vec{x} + r_{41}F_T\vec{z} - r_{41}F_A\sin\theta\,\vec{y} - r_{41}F_A\cos\theta\,\vec{x}$$

$TMS\ en\ E$:

$/\vec{x}$ $\quad (l_{EF} + l_{FD} + l_{DG})Y_G + l_{EF}F_{T\_conique} - (l_{EF} + l_{FD})(F_T \sin\theta + F_R \cos\theta) - r_{41}F_A \cos\theta = 0$

$/\vec{y}$ $\quad -(l_{EF} + l_{FD} + l_{DG})X_G + l_{EF}F_{R\_conique} - r_{m53}F_{A\_conique} + (l_{EF} + l_{FD})(F_T \cos\theta - F_R \sin\theta) - r_{41}F_A \sin\theta = 0$

$/\vec{z}$ $\quad r_{m53}F_{T\_conique} + r_{41}F_T = 0$

$Donc$:

$$\boldsymbol{Y_G = \frac{r_{41}F_A \cos\theta - l_{EF}F_{T\_conique} + (l_{EF} + l_{FD})(F_T \sin\theta + F_R \cos\theta)}{(l_{EF} + l_{FD} + l_{DG})}}$$

$$\boldsymbol{X_G = \frac{l_{EF}F_{R\_conique} - r_{m53}F_{A\_conique} + (l_{EF} + l_{FD})(F_T \cos\theta - F_R \sin\theta) - r_{41}F_A \sin\theta}{(l_{EF} + l_{FD} + l_{DG})}}$$

$$\boldsymbol{X_E = F_{R\_conique} + F_T \cos\theta - F_R \sin\theta - X_G}$$

$$\boldsymbol{Y_E = F_T \sin\theta + F_R \cos\theta - F_{T_{conique}} - Y_G}$$

3. La Statique sur Python

Les calculs et les applications numériques sont calculées à l'aide de python et affichées.

```
Efforts dans les engrenages entre 41 et 51 (engrenages à dentures hélicoîdalles)
Ft:  5636.363636363636
Fa:  1984.8769634483235
Fr:  2174.956862890313
Efforts dans les engrenages entre 43 et 53 (engrenages coniques à denture droite
Ft:  2183.977455716586
Fa:  758.3205673557769
Fr:  238.37020916355624
Statique arbre 51-31
Xa =  2123.847167325428 Ya =  235.216573303725 Za =  1984.8769634483235
Xb =  3512.516469038208 Yb =  -2410.173436194038
Statique arbre 41-53
Xg =  3017.5036048762277 Yg =  1552.2912834160234
Xe =  2462.8677299272504 Ye =  -732.8612417147565
```

J'ai également calculé l'effort radial qui correspond à la racine de la somme des carré des efforts sur X et Y. Voir ci-contre :

```
Fre = ((Xe**2)+(Ye**2))**(1/2)
Frg = ((Xg**2)+(Yg**2))**(1/2)
```

III - Calcul de durée de vie

1. Roulements rigides à une rangée de billes

Pour les calculs de roulements rigides à une rangée de billes, il faut choisir des roulements ayant les dimensions (longueur max et largeur max), et ayant Nadm > N. A partir du rapport Fa/C0, on détermine e,Y. Ensuite on calcule la rapport Fa/Fr pour déterminer la charge P appliquée au roulement. On finit par calculer les durées de vies.

Pour déterminer e et Y, j'ai créé une fonction :

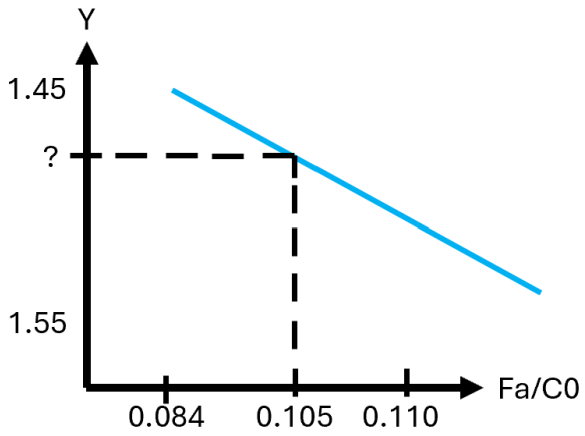
```
liste_rapport_Fa_sur_Co = [0.014,0.028,0.056,0.084,0.110,0.170,0.280,0.420,0.560]
liste_e = [0.19,0.22,0.26,0.28,0.30,0.34,0.38,0.42,0.44]
liste_Y = [2.30,1.99,1.71,1.55,1.45,1.31,1.15,1.04,1.00]

def calcul_e_et_Y(rapport_FaCo):

    e = 0
    Y = 0
    #CAS si le rapport est déjà dans la liste
    for k in range(len(liste_rapport_Fa_sur_Co)):
        if rapport_FaCo == liste_rapport_Fa_sur_Co[k]:
            e = liste_e[k]
            Y = liste_Y[k]
            return e, Y


    #CAS si le rapport est entre deux valeurs dans la liste
    for i in range(len(liste_rapport_Fa_sur_Co)-1):
        if liste_rapport_Fa_sur_Co[i] < rapport_FaCo < liste_rapport_Fa_sur_Co[i+1]:
            #Calcul de la droite y = a*x + b pour e
            a = (liste_e[i+1] - liste_e[i])/(liste_rapport_Fa_sur_Co[i+1] - liste_rapport_Fa_sur_Co[i])
            b = liste_e[i] - (a*liste_rapport_Fa_sur_Co[i])
            e = (a * rapport_FaCo) + b
            #Calcul de la droite y = c*x + d pour Y
            c = (liste_Y[i+1] - liste_Y[i])/(liste_rapport_Fa_sur_Co[i+1] - liste_rapport_Fa_sur_Co[i])
            d = liste_Y[i] - (c*liste_rapport_Fa_sur_Co[i])
            Y = (c * rapport_FaCo) + d
            return e,Y
    else:
        if rapport_FaCo > 0.560:
            return 0.5,1
        else:
            return 0,0
```

Dans un premier, je vérifie si le rapport Fa/C0 est directement égale à l'une de valeur de référence, c'est-à-dire à une valeur dans liste_rapport_Fa_sur_C0. Si ce n'est pas le cas, si le rapport est bornée entre deux valeurs de la liste, à partir d'un calcul de droite, on détermine précisément la valeur de de e et Y comme le montre le schéma ci-contre.



Enfin si le rapport calculée est supérieur à la valeur maximal constructeur qui est 0.560, on renvoie e = 0.5 et Y = 1 et si le rapport est plus petit que 0.014, on renvoie e = 0 et Y = 0.

Pour le calcul de durée de vie, j'ai également codé une fonction :

```
def duree_de_vie (Roulement, C, Co, Fa, Fr, N, e, Y):

    #Iniatialisation
    n = 0
    X = 0
    P = 0
    L10 = 0
    L10h = 0

    if Roulement == "billes": #cas roulement à bille
        n = 3
        X = X_roulement_rigides_a_une_rangee_de_bille
        #Rapport Fa/Co et calcul de e et Y
        rapport_FaCo = abs(Fa/(Co*(10**3)))
        rapport_FaCo = round(rapport_FaCo, 3) #on arrondit le rapport à 3 décimales

        e, Y = 0,0
        e, Y = calcul_e_et_Y(rapport_FaCo)
        #Calcul de la charge P
        if abs(Fa/Fr) <= e:
            P = Fr
        else:
            P = (X*Fr)+(Y*Fa)

        #Calcul de la durée de vie nominale du roulement
        L10 = (((C*(10**3))/P)**n)*(10**6)
        L10h = L10/(60*N)

        return L10h


    else: #cas roulement à rouleaux
        n = 10/3
        X = X_roulement_a_rouleaux_coniques

        if abs(Fa/Fr) <=e:
            P = Fr
        else:
            P = (X*Fr) + (Y*Fa)
        L10 = (((C*(10**3))/P)**n)*(10**6)
        L10h = L10/(60*N)
        return L10h
```

On commence par faire une disjonction de cas sur le type de roulement puisque le X va être différent selon le roulement. On calcule ensuite le rapport Fa/Fr pour déterminer P et par conséquent, déterminer la durée de vie du roulement en heures.

2. Roulements à rouleaux coniques

Enfin, pour les roulements coniques, j'ai créé une fonction qui permet de déterminer les efforts axiaux en fonction des montages. On calcule donc les efforts induits, et on prend la norme de l'effort axial, ou la valeur algébrique de cette effort.

```python
def roulements_coniques(Fra, Frb, Y, Fae):

    #on prend la valeur abs
    Fae = abs(Fae)

    Fia = (0.5 * Fra)/Y
    Fib = (0.5 * Frb)/Y


    liste_coniques = [[],[]]
    #Montage en O
    if Fia <= Fae + Fib:
        Faa = Fae + Fib
        Fab = Fib
    else:
        Faa = Fia
        Fab = Fia - Fae
    liste_coniques[0] = [Faa, Fab]
    #Montage en X
    if Fia + Fae >= Fib:
        Fab = Fae + Fia
        Faa = Fia
    else:
        Fab = Fia
        Faa = Fia - Fae
    liste_coniques[1] = [Faa, Fab]

    return liste_coniques
```

On va distinguer les 2 cas, pour le montage en O et montage en X. On procède par les calculs vue dans le cours. Fae correspond à la force axiale exercée sur l'axe de rotation de l'arbre. Cette dernière vaut Fa_conique - Fa5141 et se note effort_axiale dans le script python.

3. <u>Affichage sur python des calculs et choix des roulements</u>

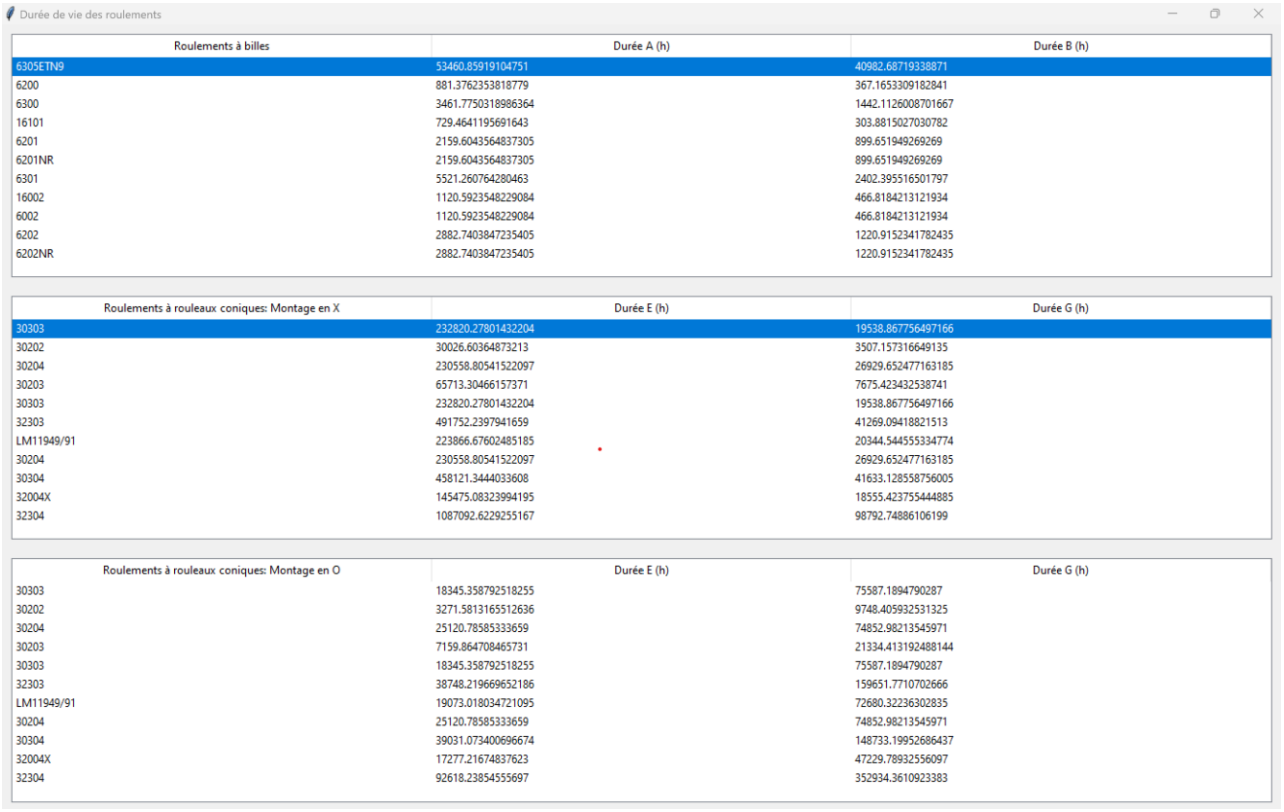
J'ai créé deux fichiers contenant tout les différents roulements possibles trouvé sur le site d'SKF. J'ai déjà procéder par un prétrie, j'ai pris des roulements avec les diamètres cohérents. C'est à partir de ces fichiers qu'on va tout calculés.

Pour les roulements à rouleaux coniques. On procède par faire un montage en X pour plusieurs raisons. Tout d'abord, la charge est mobile par rapport à l'arbre, puis les roulements se sont pas si proches donc on a pas de risques d'avoir les centres coïncidents. On remarque que un montage en O permet d'avoir des meilleurs durée de vie mais il n'est pas à privilégier dans notre situation. J'ai choisi le roulement 30303 et pour faciliter la lecture ses durées de vie sont les premières affichés sur le tableau. En effet, le pignon 41 impose un diamètre d'arbre de 17 mm pour que ce soit usinable donc ce roulement correspondait.

Pour les roulements à billes, il suffisait de trouver un roulement qui respectaient les diamètres imposées et des durées de vies élevées.

Ainsi pour les choix des roulements, on doit respecter le cahier des charges. En effet, les roulements à billes 6305ETN9 et les roulements à rouleaux coniques 30303 respectent les demandes.

Concernant l'affichage finale, j'ai choisi la mise en forme la tableau à l'aide de tkinter. Cela m'a permis de voir plus claire les durées de vie associées à chaque roulement, et présente de manière efficace les calculs. Je me suis inspiré de programme déjà existant (voir annexe). L'affichage donne :


Durée de vie des roulements

Roulements à billes	Durée A (h)	Durée B (h)
6305ETN9	53460.85919104751	40982.68719338871
6200	881.3762353818779	367.1653309182841
6300	3461.7750318986364	1442.1126008701667
16101	729.4641195691643	303.881502703078
6201	2159.6043564837305	899.651949269269
6201NR	2159.6043564837305	899.651949269269
6301	5521.260764280463	2402.395516501797
16002	1120.5923548229084	466.8184213121934
6002	1120.5923548229084	466.8184213121934
6202	2882.7403847235405	1220.9152341782435
6202NR	2882.7403847235405	1220.9152341782435

Roulements à rouleaux coniques: Montage en X	Durée E (h)	Durée G (h)
30303	232820.27801432204	19538.867756497166
30202	30026.60364873213	3507.157316649135
30204	230558.80541522097	26929.652477163185
30203	65713.30466157371	7675.423432538741
30303	232820.27801432204	19538.867756497166
32303	491752.2397941659	41269.09418821513
LM11949/91	223866.67602485185	20344.544555334774
30204	230558.80541522097	26929.652477163185
30304	458121.3444033608	41633.128558756005
32004X	145475.08323994195	18555.423755444885
32304	1087092.6229255167	98792.74886106199

Roulements à rouleaux coniques: Montage en O	Durée E (h)	Durée G (h)
30303	18345.358792518255	75587.1894790287
30202	3271.5813165512636	9748.405932531325
30204	25120.78585333659	74852.98213545971
30203	7159.864708465731	21334.413192488144
30303	18345.358792518255	75587.1894790287
32303	38748.219669652186	159651.7710702666
LM11949/91	19073.018034721095	72680.32236302835
30204	25120.78585333659	74852.98213545971
30304	39031.073400696674	148733.19952686437
32004X	17277.21674837623	47229.78932556097
32304	92618.23854555697	352934.3610923383

Remarque : On peut aussi imprimé des listes des durées de vie pour chaque point et chaque roulement, il suffit d'enlever le # devant les print correspondants.

Une fois les roulements trouvés et la maquette SolidWorks finit, j'ai repris les valeurs des distances pour vérifier la cohérence des valeurs de durée de vie. En effet, lors des calculs généraux, j'ai pris la moyenne de chaque longueur possible. Dans le cas des distance par rapport aux roulements, j'ai souhaité faire une valeur approchée, un ordre de grandeur qui me permettra d'avoir plus ou moins les mêmes valeurs. J'ai par ailleurs laissé sur le python ces ordres de grandeurs. J'ai donc vérifier la cohérence et effectivement dans le cas de nos roulements choisis, c'est nouvelle distance impose une différence d'une centaine de Newton, l'ordre de grandeur était bien choisi. On remarque aussi que l'hypothèse d'une rotule de A fonctionne.

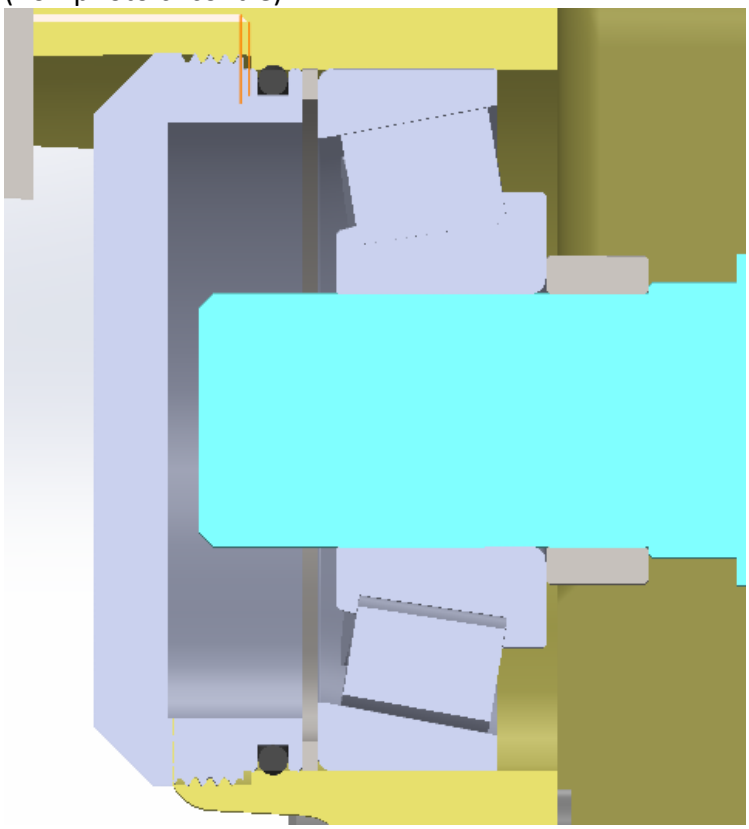
IV - Solidworks

1. Choix du montage sur l'arbre 41-53

Pour faciliter le montage de l'arbre, j'ai considéré l'arbre et le pignon 41 comme une pièce. En effet, pour que le pignon soit usinable, il faut que l'arbre ne dépasse pas les 17mm de diamètres pour usiner les dents du pignon. J'ai alors choisi de faire une liaison encastrement par clavette entre l'arbre-pignon 41 et la roue 53.

Concernant le montage en X, on monte en G l'arbre l'entretoise puis le bas du roulement a rouleaux 30303 en ajustement serré.

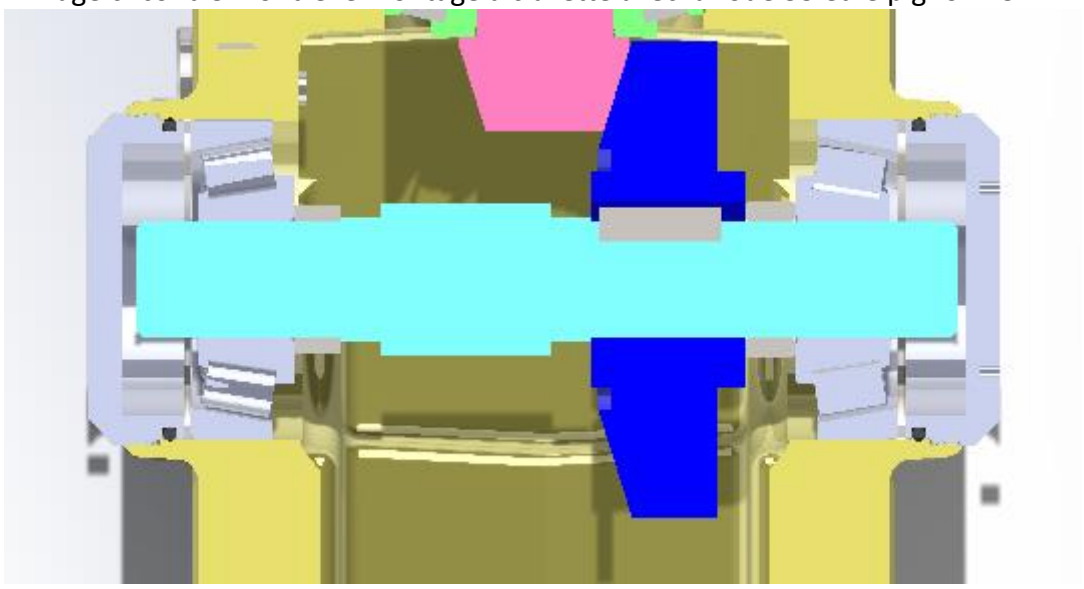
On vient y ajouter le haut du roulement dans le carter ainsi qu'une cale de réglage nommé séparateur sur le SolidWorks puis un bouchon équipé d'un joint torique pour assurer l'étanchéité statique de ce coté. Ce couvercle va exercer une précharge sur le roulement, . (Voir photo ci-contre)



En effet, on remarque l'utilisation du même couvercle pour faciliter la fabrication à grande échelle du carter.

Sur le coté au point E, on vient mettre l'engrenage conique 30303 en buté sur l'arbre et avec une clavette. On vient y mettre ensuite l'entretoise droite, le bas du roulement à rouleaux coniques, puis on vient ensuite mettre une cale de réglage nommé Séparateur sur Solidworks puis le bouchon équipé d'un joint torique pour l'étanchéité. Ce couvercle permet de maintenir en position et de mettre une précontrainte nécessaire au bon fonctionnement du roulement.

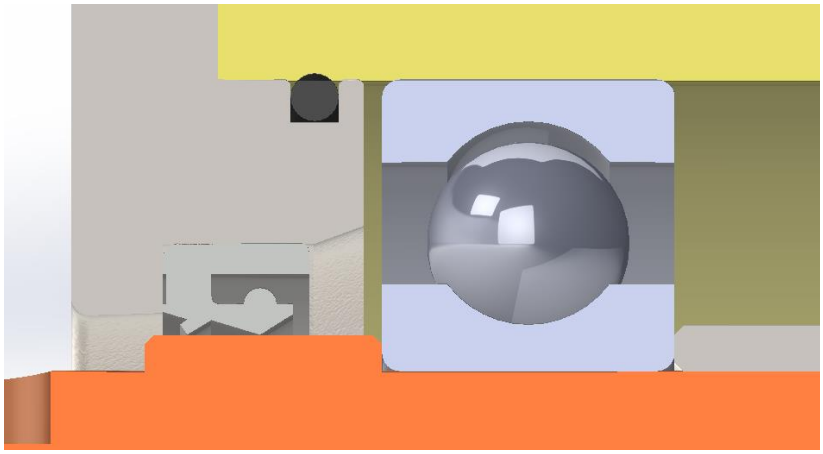
L’image ci-contre montre le montage à clavette avec la roue 53 et le pignon 43.



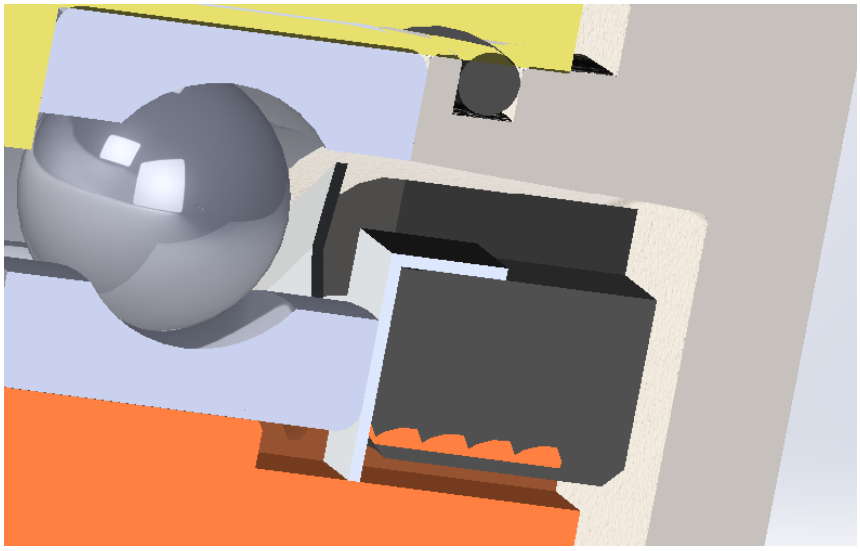
Les deux bouchon sont vissé à l’aide d’un outil spécialement conçu pour ce système.

2. <u>Choix du montage sur l’arbre 31-51</u>

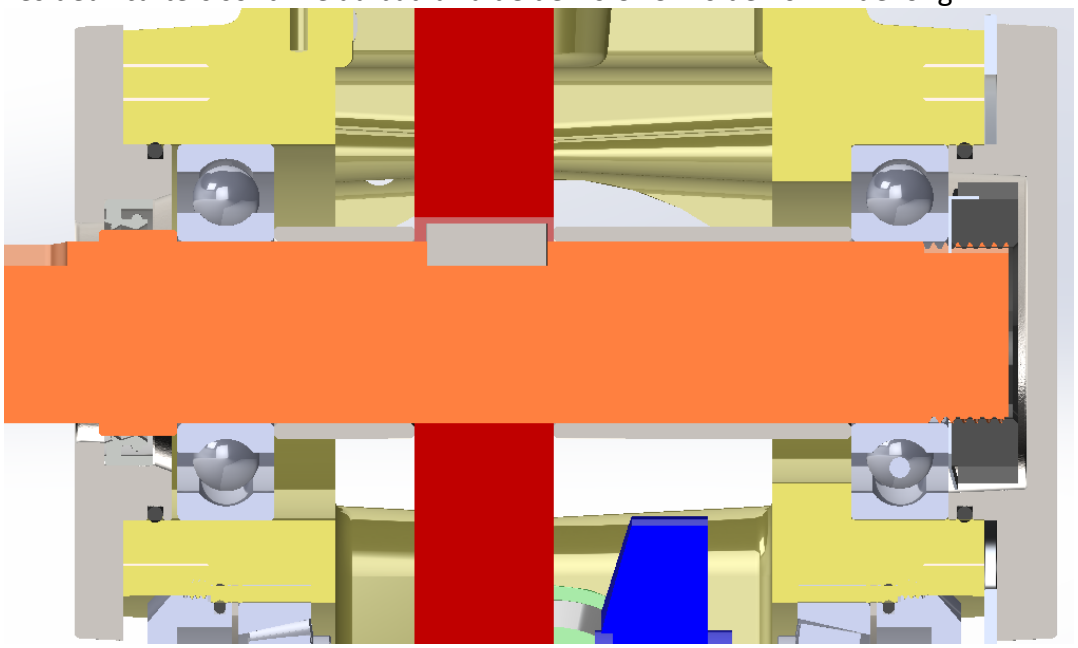
On vas venir monter sur l’arbre le roulement à bille 6206_ETN9 en ajustement serré sur l’arbre et en butée, puis l’entretoise droite comme nommé dans SolidWorks.



On monte ensuite dans le carter la roue dans l’arbre puis l’entretoise à côté du point B et le deuxième roulement à bille 6206_ETN9 en ajustement glissant sur l’arbre pour permettre le montage. Puis on fixe le tout avec un écrou à encoche au bout de l’arbre.

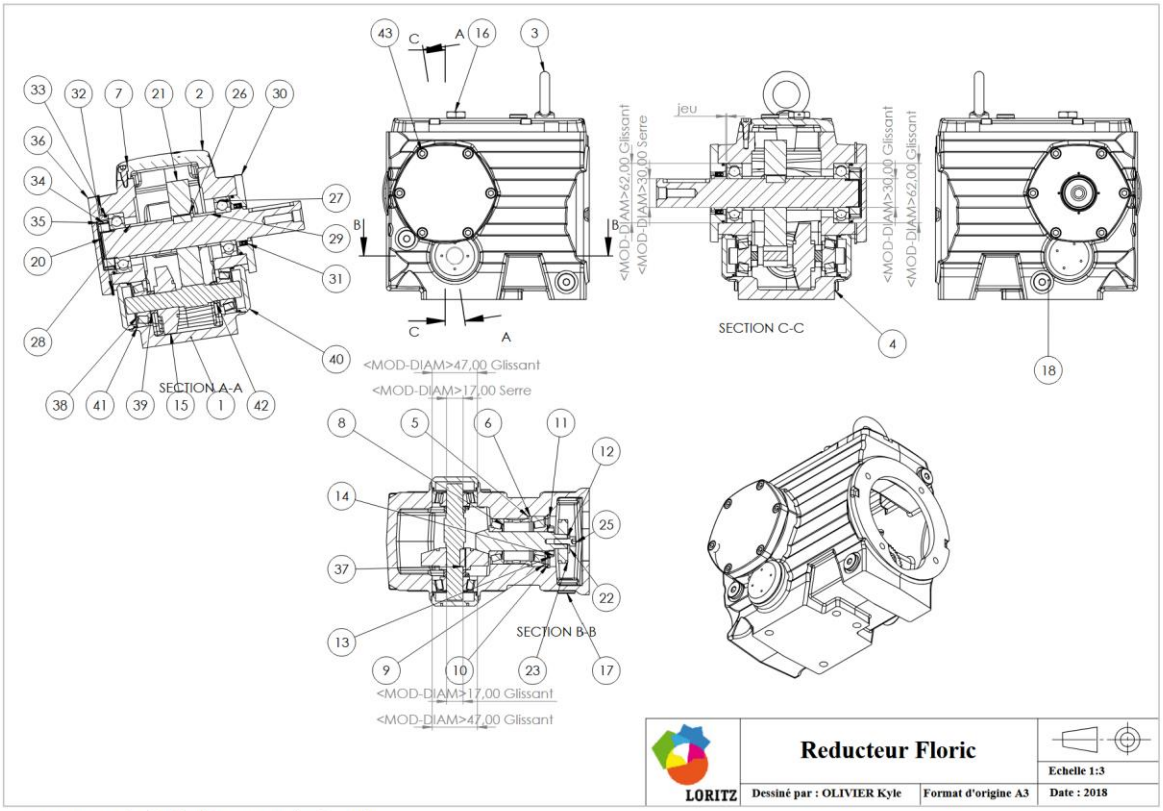


On vient fermer le tout avec deux couvercles :
-le carter droit à droite équipé d'un joint à lèvre avec lèvre antipollution et un joint torique afin d'assurer l'étanchéité et d'empêcher les impuretés de rentrer dans le système. Ce carter a un jeu présent entre lui et le roulement à bille.
-le carter gauche à gauche avec un joint torique pour l'étanchéité.
Les deux carters sont fixé au bâti à l'aide de vis CHC M6 de 16mm de long.



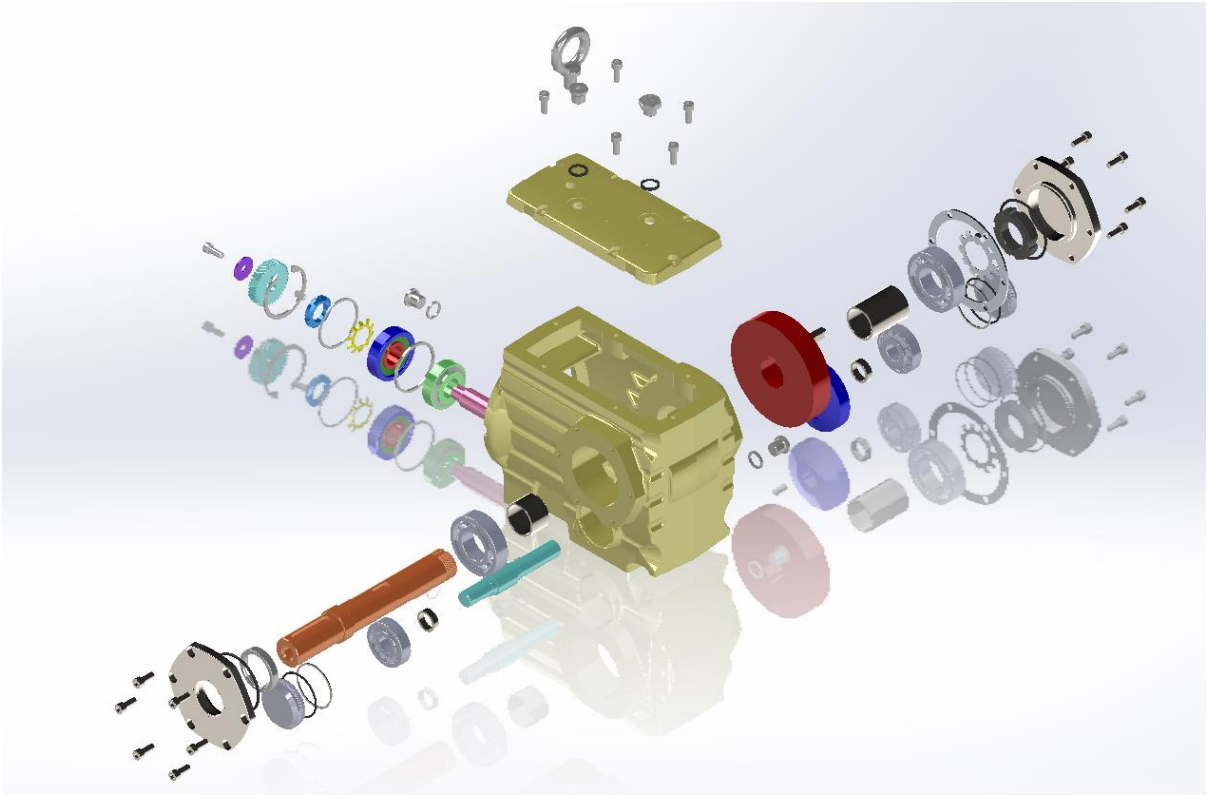
3. <u>Mise en plan et éclaté du mécanisme</u>

Mise en plan :



J'ai remarqué que le symbole diamètre ne s'affiche pas dans la mise en plan, j'ai par conséquent laisser le terme MOD-DIAM pour désigner le diamètre.

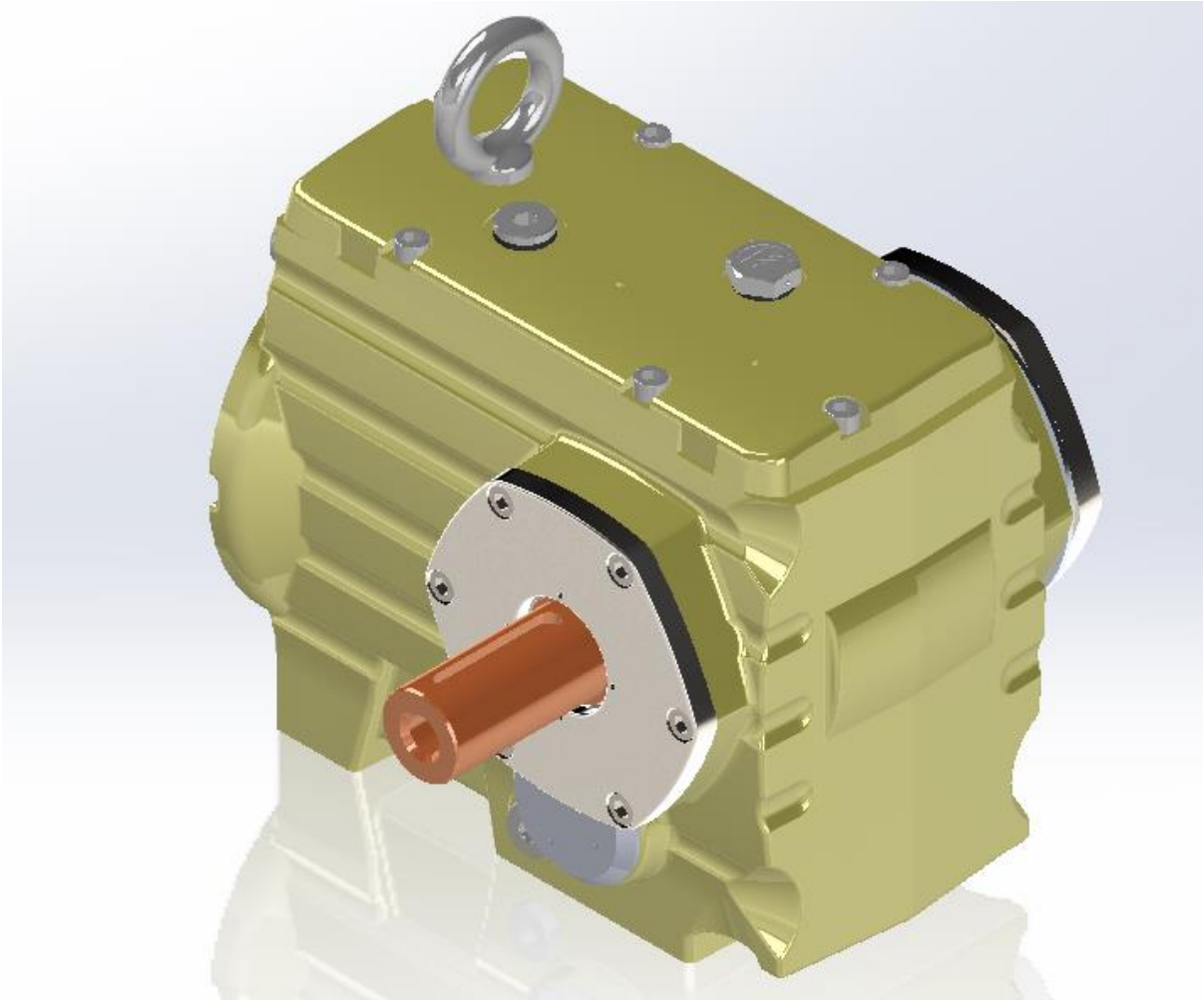
Eclaté du mécanisme :



La nomenclature est sous forme de tableau dans le fichier Nomenclature.pdf

V – Conclusion

Pour conclure, ce projet m'a permis de mettre à jour mes compétences SolidWorks, de s'améliorer en calcul statique et d'acquérir des nouvelles connaissances en python. J'ai réussi à concevoir un réducteur qui respectent le cahier des charges requis.



VI - Annexe et références bibliographiques

Interface tkinter tutoriel, https://python.doctor/page-tkinter-interface-graphique-python-tutoriel

Code python :

```
import numpy as np
import tkinter as tk
from tkinter import ttk

# ----------------------------------
# Donnéés
# ----------------------------------

#Couple de l'arbre de sortie 31
Cs = 310 #N.m
#Vitesse de rotation du moteur
N = 1450 #tr/min
#Pignon 43
Z43 = 11 #dents
rm43 = 9.05 * (10**-3) #m rayon moyen du pignon 43
#Roue 53
Z53 = 35 #dents
Delta = 72.55 #degre, l'angle de demi-sommet du cône mésurée à l'aide de SW
Alpha53 = 20 #degre, l'angle de pression
rm53 = 28.8 * (10**-3) #m le rayon moyen de la roue 53
#Pignon 41
Z41 = 14 #dents
#----Roue 51----
Z51 = 69 #dents
Beta = 19.4 #angle d'hélice
Alpha = 20 #angle de pression
m51 = 1.5 #mm (module)

#longueur du pignon a A (ordre de grandeur)
#lca = (11.5 + (34.13/2)+36) * 10**(-3)
lca = 64.5 * (10**(-3))
#longueur du pignon a B
#lcb = ((35/2) + 13 + 11.5) * 10**(-3)
lcb = 39 *(10**(-3))
#diametre primitif de la roue 51
#d51 = m51 * Z51 * (10**-3) #m
d51 = 110 * 10**(-3)
#diametre primitif du pignon 41
d41 = m51 * Z41 * (10**-3) #m
r41 = d41/2 #rayon du pignon 41

#longueur du roulement E à la roue 53 (à F)
#lef = 39.19 * (10**(-3)) #ordre de grandeur de la distance
lef = 22.7 * (10**(-3))#m
#longueur de la roue 53 (de F) au pignon 41 (à D)
lfd = 21.31 * (10**(-3)) #m
#longueur du pignon 41 (de D) au roulement G
#ldg = 37.5 * (10**(-3)) #ordre de grandeur de la distance
ldg = 23.85 * (10**(-3))
```

```
#Arbre 31-51
#Couple de cette arbre
C3151 =  Cs
#Vitesse de rotation
N3151 = N * (Z43/Z53) * (Z41/Z51)

#Arbre 41-53
#Couple de cette arbre
C4153 = C3151 * (Z41/Z51)
#Vitesse de rotation
N4153 = N * (Z43/Z53)


# ------------------------------------
### STATIQUE
# ------------------------------------

# Efforts dans les engrenages entre 51 et 41
Ft5141 = C3151/(d51/2) #effort tangentiel
Fa5141 = Ft5141 *(np.tan(np.deg2rad(Beta))) #effort axial
Fr5141 = Ft5141 * (np.tan(np.deg2rad(Alpha))/np.cos(np.deg2rad(Beta))) #effort radial


print("Efforts dans les engrenages entre 41 et 51 (engrenages à dentures hélicoïdalles)")
print("Ft: ", Ft5141)
print("Fa: ", Fa5141)
print("Fr: ", Fr5141)

# Effort dans les engrenages coniques entre 43 et 53
Ft_conique = C4153/rm53 #effort tangentiel
Fa_conique = Ft_conique * np.tan(np.deg2rad(Alpha53)) * np.sin(np.deg2rad(Delta)) #effort axial
Fr_conique = Ft_conique * np.tan(np.deg2rad(Alpha53)) * np.cos(np.deg2rad(Delta)) #effort radial

print("Efforts dans les engrenages entre 43 et 53 (engrenages coniques à denture droite")
print("Ft: ", Ft_conique)
print("Fa: ", Fa_conique)
print("Fr: ", Fr_conique)


## STATIQUE ARBRE 51-31
Xc = - Ft5141
Yc = Fr5141
Zc = - Fa5141

#TMS
#Zb = 0 car sphère cylindre en B et rotule en A
Yb = ((lca * Fr5141) + ((d51/2)*Fa5141))/(-(lca + lcb))
Xb = (lca * Ft5141)/(lca + lcb)

#TRS
Xa = Ft5141 - Xb
Ya = - Yb - Fr5141
Za = Fa5141

print("Statique arbre 51-31")
print("Xa = ",Xa,"Ya = ",Ya,"Za = ",Za)
print("Xb = ",Xb,"Yb = ",Yb)

## STATIQUE ARBRE 53-41

#Angle du changement de base
theta = 8.71 #degré

# TMS

Yg = ((r41*np.cos(np.deg2rad(theta))*Fa5141) - (lef*Ft_conique) + ((lef+lfd)*((Ft5141*np.sin(np.deg2rad(theta)))+(Fr5141*np.cos(np.deg2rad(theta))))))/(lef+lfd+ldg)
Xg = ((lef*Fr_conique)-(rm53*Fa_conique)+((lef+lfd)*((Ft5141*np.cos(np.deg2rad(theta)))-(Fr5141*np.sin(np.deg2rad(theta)))))-(r41*np.sin(np.deg2rad(theta))*Fa5141))/((l

#TRS

Xe = Fr_conique + (Ft5141*np.cos(np.deg2rad(theta))) - (Fr5141*np.sin(np.deg2rad(theta))) - Xg
Ye = (Fr5141*np.cos(np.deg2rad(theta))) + (Ft5141*np.sin(np.deg2rad(theta))) - Ft_conique - Yg
```

```
print("Statique arbre 41-53")
print("Xg = ",Xg,"Yg = ",Yg)
print("Xe = ",Xe,"Ye = ",Ye)

Fre = ((Xe**2)+(Ye**2))**(1/2)
Frg = ((Xg**2)+(Yg**2))**(1/2)

print("Force radiale au point E: ",Fre)
print("Force radiale au point G: ",Frg)

Fr_roulements_coniques = [Fre,Frg]

#Effort Fr et Fa sur les roulements A et B

Fab = 0
Frb = ((Xb**2) + (Yb**2))**(1/2)
Faa = Za
Fra = ((Xa**2) + (Ya**2))**(1/2)

print("Force radial au point A: ", Fra, "Force axiale au point A: ", Faa)
print("Force radial au point B: ", Frb, "Force axiale au point B: ", Fab)

Fa_total_arbre5131 = [Faa,Fab]
Fr_total_arbre5131 = [Fra,Frb]

# -----------------------------------
## Calcul de durée de vie et dimensions
# -----------------------------------

X_roulement_rigides_a_une_rangee_de_bille = 0.56
X_roulement_a_rouleaux_coniques = 0.4

liste_rapport_Fa_sur_Co = [0.014,0.028,0.056,0.084,0.110,0.170,0.280,0.420,0.560]
liste_e = [0.19,0.22,0.26,0.28,0.30,0.34,0.38,0.42,0.44]
liste_Y = [2.30,1.99,1.71,1.55,1.45,1.31,1.15,1.04,1.00]


def calcul_e_et_Y(rapport_FaCo):

    e = 0
    Y = 0
    #CAS si le rapport est déjà dans la liste
    for k in range(len(liste_rapport_Fa_sur_Co)):
        if rapport_FaCo == liste_rapport_Fa_sur_Co[k]:
            e = liste_e[k]
            Y = liste_Y[k]
            return e, Y


    #CAS si le rapport est entre deux valeurs dans la liste
    for i in range(len(liste_rapport_Fa_sur_Co)-1):
        if liste_rapport_Fa_sur_Co[i] < rapport_FaCo < liste_rapport_Fa_sur_Co[i+1]:
            #Calcul de la droite y = a*x + b pour e
            a = (liste_e[i+1] - liste_e[i])/(liste_rapport_Fa_sur_Co[i+1] - liste_rapport_Fa_sur_Co[i])
            b = liste_e[i] - (a*liste_rapport_Fa_sur_Co[i])
            e = (a * rapport_FaCo) + b
            #Calcul de la droite y = c*x + d pour Y
            c = (liste_Y[i+1] - liste_Y[i])/(liste_rapport_Fa_sur_Co[i+1] - liste_rapport_Fa_sur_Co[i])
            d = liste_Y[i] - (c*liste_rapport_Fa_sur_Co[i])
            Y = (c * rapport_FaCo) + d
            return e,Y
    else:
        if rapport_FaCo > 0.560:
            return 0.5,1
        else:
            return 0,0
```

```
def duree_de_vie (Roulement, C, Co, Fa, Fr, N, e, Y):

    #Iniatialisation
    n = 0
    X = 0
    P = 0
    L10 = 0
    L10h = 0

    if Roulement == "billes": #cas roulement à bille
        n = 3
        X = X_roulement_rigides_a_une_rangee_de_bille
        #Rapport Fa/Co et calcul de e et Y
        rapport_FaCo = abs(Fa/(Co*(10**3)))
        rapport_FaCo = round(rapport_FaCo, 3) #on arrondit le rapport à 3 décimales

        e, Y = 0,0
        e, Y = calcul_e_et_Y(rapport_FaCo)
        #Calcul de la charge P
        if abs(Fa/Fr) <= e:
            P = Fr
        else:
            P = (X*Fr)+(Y*Fa)

        #Calcul de la durée de vie nominale du roulement
        L10 = (((C*(10**3))/P)**n)*(10**6)
        L10h = L10/(60*N)

        return L10h


    else: #cas roulement à rouleaux
        n = 10/3
        X = X_roulement_a_rouleaux_coniques

        if abs(Fa/Fr) <=e:
            P = Fr
        else:
            P = (X*Fr) + (Y*Fa)
        L10 = (((C*(10**3))/P)**n)*(10**6)
        L10h = L10/(60*N)
        return L10h
```

```
## Choix du roulement

#Roulement à billes/ Arbre 51-31

#Roulement à billes SKF
dtype1 = [('désignation', 'U10'), ('d[mm]', 'f8'), ('D[mm]', 'f8'), ('B[mm]', 'f8'),('C_kN', 'f8'), ('C0_kN', 'f8'), ('r/min', 'f8'), ('r/min2', 'f8')]
donnees_roulement_billes = np.genfromtxt("roulement_a_billes.txt",delimiter=";", skip_header=1,encoding="utf-8",dtype = dtype1)
# Extraction des colonnes utiles
vec_nom_billes = donnees_roulement_billes["désignation"]
vec_C_roulement_bille = donnees_roulement_billes["C_kN"]
vec_Co_roulement_bille = donnees_roulement_billes["C0_kN"]


duree_roulement_a = []
duree_roulement_b = []
nom_billes =[]

for k in range(len(Fa_total_arbre5131)):
    #print(vec_C_rab[k])
    for i in range (len(vec_C_roulement_bille)):
        duree_roulement = duree_de_vie("billes",vec_C_roulement_bille[i],vec_Co_roulement_bille[i],Fa_total_arbre5131[k],Fr_total_arbre5131[k],N3151,0,0)
        if k == 0:
            duree_roulement_a.append(duree_roulement)
        else:
            duree_roulement_b.append(duree_roulement)

for i in range (len(vec_nom_billes)):
    nom_billes.append(vec_nom_billes[i])

#print("Rouelement à billes, arbre 31-51")
#print("Duree roulement a: ",duree_roulement_a)
#print("Duree roulement b: ",duree_roulement_b)
```

```
## Roulement coniques

def roulements_coniques(Fra, Frb, Y, Fae):

    #on prend la valeur abs
    Fae = abs(Fae)

    Fia = (0.5 * Fra)/Y
    Fib = (0.5 * Frb)/Y


    liste_coniques = [[],[]]
    #Montage en O
    if Fia <= Fae + Fib:
        Faa = Fae + Fib
        Fab = Fib
    else:
        Faa = Fia
        Fab = Fia - Fae
    liste_coniques[0] = [Faa, Fab]
    #Montage en X
    if Fia + Fae >= Fib:
        Fab = Fae + Fia
        Faa = Fia
    else:
        Fab = Fia
        Faa = Fia - Fae
    liste_coniques[1] = [Faa, Fab]

    return liste_coniques

## Choix du roulement
#Roulement à billes SKF
dtype2 = [('désignation', 'U10'), ('d[mm]', 'f8'), ('D[mm]', 'f8'), ('B[mm]', 'f8'),('C_kN', 'f8'), ('C0_kN', 'f8'), ('r/min', 'f8'), ('r/min2', 'f8'),('e','f8'),('Y','
donnees_roulement_coniques = np.genfromtxt("roulement_a_rouleaux_coniques.txt",delimiter=";", skip_header=1,encoding="utf-8",dtype = dtype2)
# Extraction des colonnes utiles
vec_nom_rouleaux = donnees_roulement_coniques["désignation"]
vec_C_roulement_conique = donnees_roulement_coniques["C_kN"]
vec_Co_roulement_conique = donnees_roulement_coniques["C0_kN"]
vec_e_roulement_conique = donnees_roulement_coniques["e"]
vec_Y_roulement_conique = donnees_roulement_coniques["Y"]


duree_roulement_e_O = []
duree_roulement_g_O = []

duree_roulement_e_X = []
duree_roulement_g_X = []

effort_axiale = Fa_conique - Fa5141
```

```
for i in range (len(vec_C_roulement_conique)):
    liste_coniques = roulements_coniques(Fre,Frg,vec_Y_roulement_conique[i],effort_axiale)
    Fae, Fag = liste_coniques[0][0],liste_coniques[0][1]
    duree_roulement_e = duree_de_vie("rouleaux",vec_C_roulement_conique[i],vec_Co_roulement_conique[i],Fae,Fre,N4153,vec_e_roulement_conique[i],vec_Y_roulement_conique[
    duree_roulement_g = duree_de_vie("rouleaux",vec_C_roulement_conique[i],vec_Co_roulement_conique[i],Fag,Frg,N4153,vec_e_roulement_conique[i],vec_Y_roulement_conique[
    duree_roulement_e_O.append(duree_roulement_e)
    duree_roulement_g_O.append(duree_roulement_g)

#print("Montage en O")
#print("Duree roulement e: ",duree_roulement_e_O)
#print("Duree roulement g: ",duree_roulement_g_O)

for i in range (len(vec_C_roulement_conique)):
    liste_coniques = roulements_coniques(Fre,Frg,vec_Y_roulement_conique[i],effort_axiale)
    Fae, Fag = liste_coniques[1][0],liste_coniques[1][1]
    duree_roulement_e = duree_de_vie("rouleaux",vec_C_roulement_conique[i],vec_Co_roulement_conique[i],Fae,Fre,N4153,vec_e_roulement_conique[i],vec_Y_roulement_conique[
    duree_roulement_g = duree_de_vie("rouleaux",vec_C_roulement_conique[i],vec_Co_roulement_conique[i],Fag,Frg,N4153,vec_e_roulement_conique[i],vec_Y_roulement_conique[
    duree_roulement_e_X.append(duree_roulement_e)
    duree_roulement_g_X.append(duree_roulement_g)

#print("Montage en X")
#print("Duree roulement e: ",duree_roulement_e_X)
#print("Duree roulement g: ",duree_roulement_g_X)


# ------------------------------------------------------------
## Affichage à l'aide de la bibliothèque tkinter
# ------------------------------------------------------------

# Création de la fenêtre principale
fenetre_principale = tk.Tk()
fenetre_principale.title("Durée de vie des roulements")

# ---- Tableau 1 : Roulements à billes ----
# Créer un cadre pour les roulements à billes
cadre_billes = tk.Frame(fenetre_principale)
cadre_billes.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

# Création du tableau  pour les roulements à billes
columns_roulement_bille = ("Roulement à billes", "Durée roulement en A (h)", "Durée roulement en B (h)")
arbre_roulement_bille = ttk.Treeview(cadre_billes, columns=columns_roulement_bille, show="headings")

# Définir les en-têtes des colonnes
arbre_roulement_bille.heading("Roulement à billes", text="Roulements à billes")
arbre_roulement_bille.heading("Durée roulement en A (h)", text="Durée A (h)")
arbre_roulement_bille.heading("Durée roulement en B (h)", text="Durée B (h)")

# Ajuster la largeur des colonnes
arbre_roulement_bille.column("Roulement à billes", width=150)
arbre_roulement_bille.column("Durée roulement en A (h)", width=150)
arbre_roulement_bille.column("Durée roulement en B (h)", width=150)
```

```
# Positionner le tableau dans le cadre
arbre_roulement_bille.pack(fill=tk.BOTH, expand=True)

# ---- Tableau 2 : Roulements à rouleaux coniques montage en X ----
# Créer un cadre pour les roulements à rouleaux
cadre_rouleaux = tk.Frame(fenetre_principale)
cadre_rouleaux.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

# Création du tableau pour les roulements à rouleaux coniques (E et G)
columns_roulement_rouleaux = ("Roulement à rouleaux coniques", "Durée roulement en E (h)", "Durée roulement en G (h)")
arbre_roulement_rouleaux = ttk.Treeview(cadre_rouleaux, columns=columns_roulement_rouleaux, show="headings")

# Définir les en-têtes des colonnes
arbre_roulement_rouleaux.heading("Roulement à rouleaux coniques", text="Roulements à rouleaux coniques: Montage en X")
arbre_roulement_rouleaux.heading("Durée roulement en E (h)", text="Durée E (h)")
arbre_roulement_rouleaux.heading("Durée roulement en G (h)", text="Durée G (h)")

# Ajuster la largeur des colonnes
arbre_roulement_rouleaux.column("Roulement à rouleaux coniques", width=150)
arbre_roulement_rouleaux.column("Durée roulement en E (h)", width=150)
arbre_roulement_rouleaux.column("Durée roulement en G (h)", width=150)

# Insérer les données dans le tableau des roulements à rouleaux coniques (E et G)
for j in range(len(vec_nom_rouleaux)):
    arbre_roulement_rouleaux.insert("", tk.END, values=(vec_nom_rouleaux[j], duree_roulement_e_X[j], duree_roulement_g_X[j]))

# Positionner le tableau dans le cadre
arbre_roulement_rouleaux.pack(fill=tk.BOTH, expand=True)

# ---- Tableau 3 : Roulements à rouleaux coniques montage en O ----
# Créer un cadre pour le troisième tableau
cadre_rouleaux_O = tk.Frame(fenetre_principale)
cadre_rouleaux_O.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

# Création du tableau pour les roulements à rouleaux coniques (E et G)
columns_roulement_rouleaux_O = ("Roulement à rouleaux coniques (O)", "Durée roulement en E (h)", "Durée roulement en G (h)")
arbre_roulement_rouleaux_O = ttk.Treeview(cadre_rouleaux_O, columns=columns_roulement_rouleaux_O, show="headings")

# Définir les en-têtes des colonnes
arbre_roulement_rouleaux_O.heading("Roulement à rouleaux coniques (O)", text="Roulements à rouleaux coniques: Montage en O")
arbre_roulement_rouleaux_O.heading("Durée roulement en E (h)", text="Durée E (h)")
arbre_roulement_rouleaux_O.heading("Durée roulement en G (h)", text="Durée G (h)")

# Ajuster la largeur des colonnes
arbre_roulement_rouleaux_O.column("Roulement à rouleaux coniques (O)", width=150)
arbre_roulement_rouleaux_O.column("Durée roulement en E (h)", width=150)
arbre_roulement_rouleaux_O.column("Durée roulement en G (h)", width=150)


# Insérer les données dans le tableau des roulements à rouleaux coniques (E et G avec O)
for j in range(len(vec_nom_rouleaux)):
    arbre_roulement_rouleaux_O.insert("", tk.END, values=(vec_nom_rouleaux[j], duree_roulement_e_O[j], duree_roulement_g_O[j]))

# Positionner le tableau dans le cadre
arbre_roulement_rouleaux_O.pack(fill=tk.BOTH, expand=True)

# Lancer l'interface graphique
fenetre_principale.mainloop()
```