

TP C10.2 – MICROMOTEUR

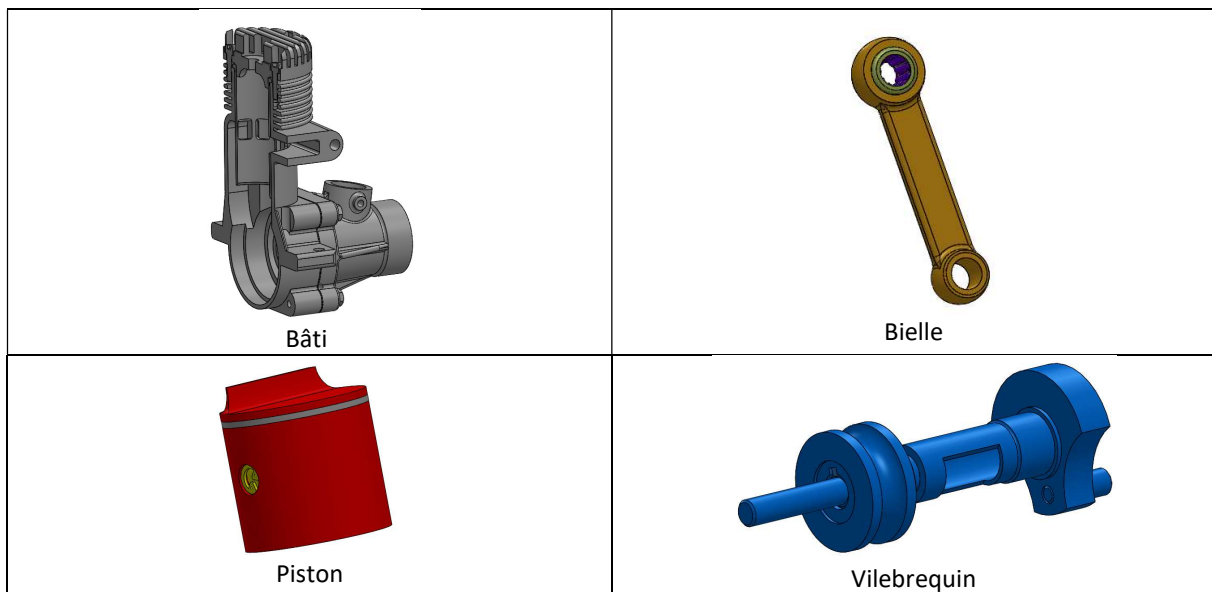
OBJECTIF

L'objectif de ce TP est de se familiariser avec l'utilisation de Solidworks et méca3D dans le but de faire de la simulation numérique. A la fin de ce TP il faudra savoir

1. Créer un assemblage et insérer les groupes cinématiques.
2. Utiliser les contraintes pour assembler le mécanisme.
3. Sous Méca3D, créer les liaisons.
4. Rentrer les paramètres de la simulation.
5. Récupérer les résultats de la simulation, les importer Python pour comparer à l'expérimentale et/ou au théorique.

SUPPORT




C'est le micromoteur du TD. Il est constitué de 4 groupes cinématique, ici ce sont des assemblages.




ASSEMBLAGE DES COMPOSANTS

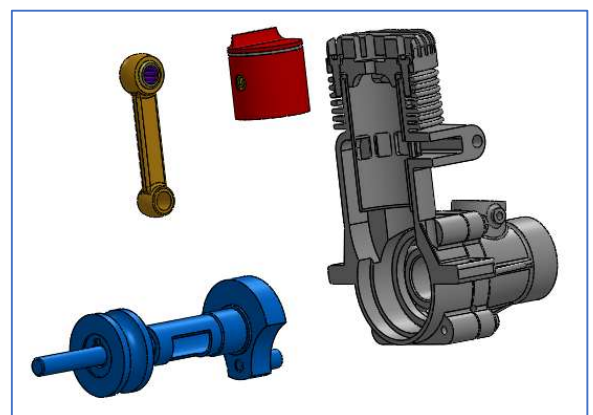
Vous savez faire avec le TP sur le Sinusmatic.


Quelques rappels :

-  Ouvrir Solidworks.
-  Créer un nouvel **assemblage**.
-  Insérer le bâti en **faisant attention de le placer au centre du repère de l'assemblage**.



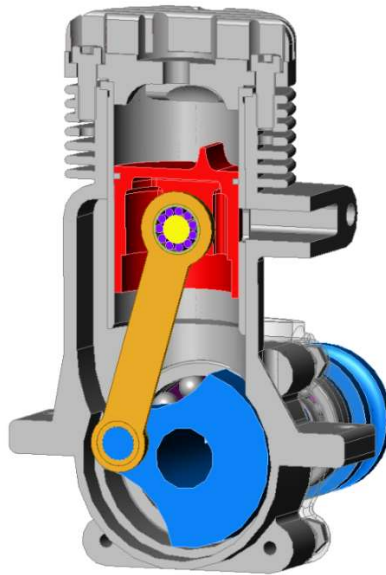
-  En cliquant sur insérer des composant : insérer n'importe où les 3 autres groupes cinématiques. Vous obtenez à peu près l'image ci-contre.



-  Pour lier deux composants, vous utilisez des contraintes géométriques. Attention de bien **prendre des surfaces** et pas des arrêtes.

Remarque : La contrainte de coaxialité entre la bielle et le piston suffit, ne pas ajouter une contrainte plan-plan.

Vous devez obtenir ceci



ANIMATION SOUS MECA3D

CREATION DES LIAISONS

L'objectif de cette partie est d'animer le système sous Méca3D. Vous l'avez également fait dans le TP précédent.



Cliquer sur l'icône de méca3D :



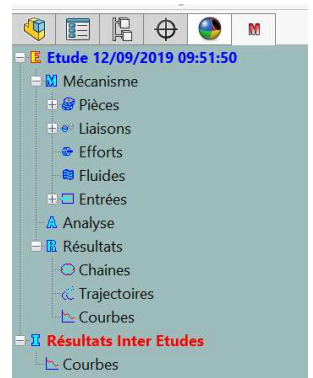
Dans la partie gauche, vous voyez que l'on doit créer des pièces (ie groupes cinématiques) et des liaisons. Lorsque tout se passe bien, nous pouvons laisser Méca3D se débrouiller en utilisant les contraintes géométriques que l'on a appliqué sous Solidworks.

Pour cela :



Faire un clic droit sur **Mécanisme** puis cliquer sur **Construction automatique**.

Si vos contraintes sont propres, Méca3D à créer 4 pièces (Bâti, Bielle, Piston, Vilebrequin) et 4 liaisons (2 pivots et 2 pivots glissants).



ANIMATION



Faire un clic droit sur **Analyse** puis cliquer sur **Calcul mécanique**.



Valider la fenêtre Analyse du mécanisme.

Nous faisons une étude **cinématique**



Fixer une vitesse de rotation à liaison entre le bâti et l'arbre d'entrée puis donner la durée de la simulation (faites en sorte de ne faire qu'un tour de vilebrequin) et le nombre de points de calculs... **Il y a un peu de réflexion.**



Lancer le calcul.

RESULTATS



Faire un clic droit sur **Résultats** puis **courbe**.

On cherche à avoir **le déplacement du piston par rapport au bâti** en fonction de **l'angle de rotation moteur par rapport au bâti**.

On a trois possibilités de courbes :

1. La **courbe simple** : c'est l'évolution d'un paramètre par rapport au temps.
2. La **courbe paramétrée** : c'est l'évolution d'un paramètre par rapport à un autre.
3. La **courbe multiple** : c'est l'évolution de plusieurs paramètres par rapport au temps.

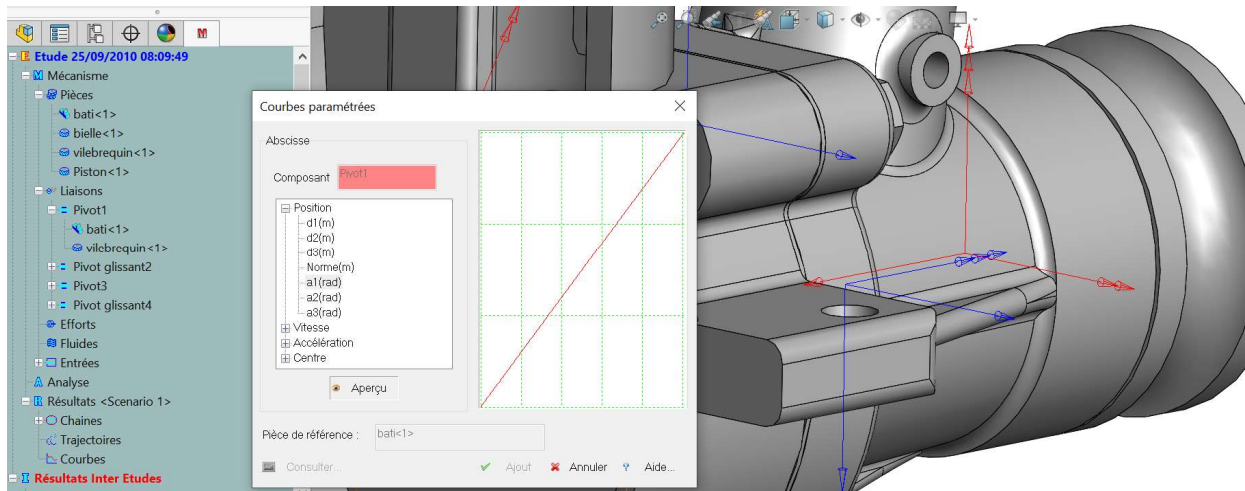
A noter que l'on peut choisir le référentiel, ce n'est pas obligatoirement le bâti.



On prend la courbe paramétrée

Il faut maintenant choisir l'abscisse et l'ordonnée.

En abscisse, nous voulons l'angle moteur. C'est le paramètre de position de la liaison pivot entre le bâti et le vilebrequin. Chez moi c'est la pivot 1... c'est donc celle que je prends comme composant.



Dans position, j'ai 7 possibilités :

$d1, d2, d3, norme, a1, a2, a3$

Bon, la **norme** je pense que vous comprenez !

La signification est **d** pour distance et **a** pour angle.

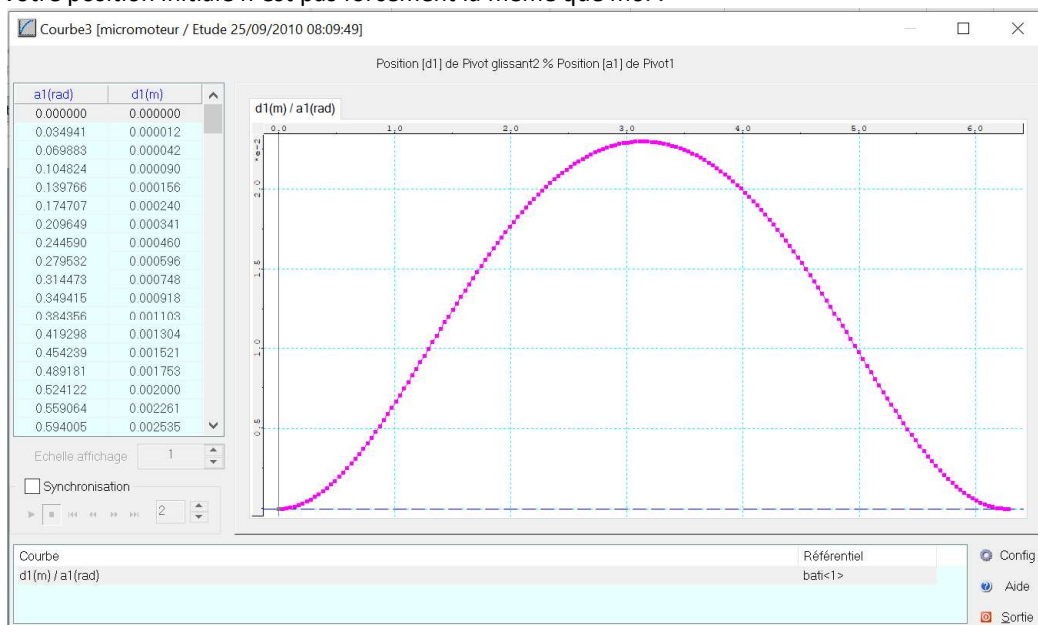
Sur le dessin 3D, les axes d'une liaison apparaissent en rouge. Le numéro du vecteur correspond au nombre de cônes sur les flèches.




Ici, je veux l'angle de rotation autour de l'axe qui ne possède qu'un cône donc je choisis **a1**.

A vous de faire pour l'axe des ordonnées !!!

Et finissez par **AJOUT**.

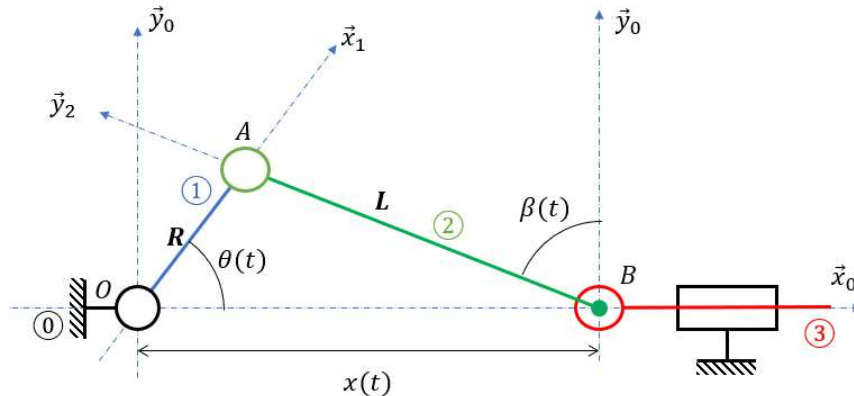
J'obtiens ceci ... votre position initiale n'est pas forcément la même que moi :



-  Faire un clic droit dans la partie des données à gauche de la courbe.
-  Puis enregistrer les données au format texte.
-  Ouvrir le fichier et supprimer les 3 dernières lignes... comme pour le TP précédent.

ANALYSE THEORIQUE

On donne le schéma cinématique du système avec le paramétrage



On a




$$\overrightarrow{OA} = R\vec{x}_1; \overrightarrow{OB} = x(t)\vec{x}_0; \overrightarrow{AB} = -L\vec{y}_2$$

En fonction du groupe, nous avons fait, ou nous ferons, l'étude en TD et :

$$x(t) = R \cdot \cos \theta + \sqrt{L^2 - R^2 \sin^2 \theta}$$


EXPLOITATION




L'idée est de comparer les résultats obtenus avec Solidworks et l'analyse théorique.

-  Ouvrir le fichier **Micromoteur-Etudiant.py**.
-  Ligne 13, **vérifier** le nom de votre fichier texte issue de Solidworks.
-  **Exécuter** le programme.


Pour récupérer les données Solidworks, j'ai créé la fonction **donneesSlw(NomFichier)** qui prend en argument le nom d'un fichier et renvoie les listes correspondants à 2 colonnes : ici aux θ et aux position x du piston. J'utilise cette fonction en ligne 51 et je trace la courbe x en fonction de θ (issue de solidworks) en ligne 62.

Vous allez maintenant ajouter la courbe issue de l'analyse analytique

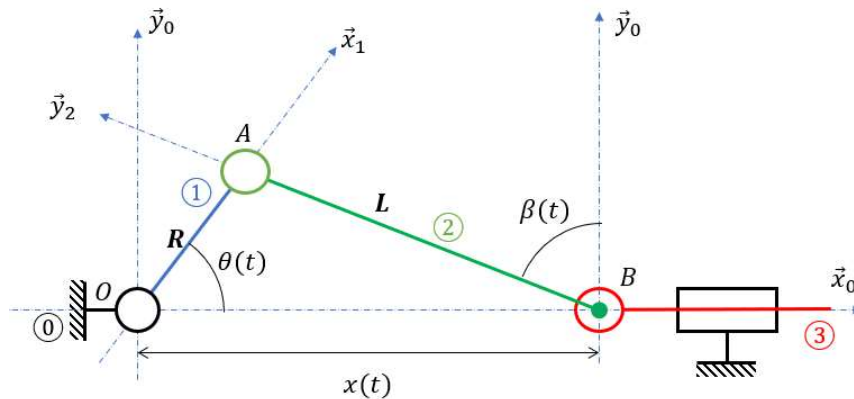
-  **Compléter** la ligne 21 qui crée la liste **ThetaAnalytique** des angles θ en utilisant la fonction **linspace** de la bibliothèque **Numpy**....

$$\text{np.linspace(depart, fin, nb de points)}$$
-  **Compléter** la ligne 46 qui calcul x pour un angle θ donné. On a alors la fonction **CalculPosition(theta)**.
-  En ligne 55 et 56, **Compléter** les 2 lignes permettant, à l'aide d'une boucle **for** de remplir la liste **XAnalytique** avec la valeur de x pour chaque θ de la liste créée en ligne 21.
-  Décommenter la ligne 63 et exécuter le programme.

Ça marche ??

-  Si les courbes ne sont pas confondues, identifier le problème et le résoudre 😊 !

RETOUR SUR LA RESOLUTION ANALYTIQUE



Pour établir l'expression de $x(\theta)$, nous sommes passés par une fermeture géométrique et par projection sur la base (\vec{x}_0, \vec{y}_0) nous avons obtenu le système d'équation qui peut s'écrire sous la forme suivante :

$$\begin{cases} x - L \cdot \sin \beta - R \cdot \cos \theta = 0 \\ L \cdot \cos \beta - R \cdot \sin \theta = 0 \end{cases}$$

Où x, β et θ sont des variables.

Nous cherchons les variables x et β pour un θ donné (c'est un argument)

Supposons que nous ne sachions pas résoudre « à la main » ce système. Comment avoir une résolution numérique ?

FSOLVE

Pour résoudre numériquement le système, nous utilisons la fonction ***fsolve*** du module ***optimize*** de la bibliothèque ***scipy***. Ainsi, en entête de notre programme, on écrira :

```
from scipy.optimize import fsolve
```

La fonction ***fsolve*** est définie de la manière suivante :

```
fsolve(système,initialisation,arguments)
```



Renseignez-vous sur la fonction ***fsolve*** de sorte à résoudre ce système avec Python.