UCLouvain

epl

École polytechnique de Louvain

De la Conception Assistée par Ordinateur à la Modélisation Multicorps

Solidworks vers Robotran

Auteur: **Louis Ridelle**
Promoteurs: **Nicolas Docquier, Paul Fisette**
Lecteurs: **Vincent Legat, Louis Devillez**
Année académique 2023–2024
Master [120] : ingénieur civil mécanicien

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce mémoire. Sans leur soutien et leur expertise, ce travail n'aurait pas pu voir le jour.

Je souhaite tout d'abord remercier chaleureusement mes promoteurs, Monsieur Nicolas Docquier et Monsieur Paul Fisette, pour leur suivi et leur encadrement tout au long de ce projet. Leur expertise et leurs conseils m'ont permis d'avancer dans la bonne direction pour l'aboutissement de ce travail.

Je tiens également à adresser mes remerciements à Monsieur Olivier Lantsoght et Monsieur Louis Devillez qui on été mes principaux encadrants dans ce travail. Leur disponibilité, leur patience et leur capacité à répondre à chacune de mes questions ont été d'une aide précieuse. Dès le début de ce projet, ils m'ont guidé avec une clarté et une direction qui ont permis de mener à bien ce travail.

Je souhaiterais également exprimer ma reconnaissance envers l'ensemble de l'équipe encadrante des TFE sous la direction de Monsieur Fisette et Monsieur Docquier. Que ce soit pour les réunions d'avancement régulières tout au long de l'année ainsi que leur volonté de nous guider dans nos travaux respectifs.

Enfin, je tiens à remercier chaleureusement l'ensemble de mon entourage, qui m'a beaucoup soutenu durant les derniers instants précédant la remise de ce travail. Leur soutien moral a été d'une grande valeur et m'a permis de rester concentré et motivé jusqu'au bout.

Résumé

Dans ce travail, nous avons développé une méthode automatisée pour la création de modèles multicorps dans le logiciel *Robotran* à partir d'assemblages réalisés dans *SolidWorks*, un logiciel de Conception Assistée par Ordinateur (CAO) largement utilisé dans l'industrie. L'objectif principal de cette méthode est de faciliter et d'accélérer la conversion des modèles CAO en modèles dynamiques exploitables pour le logiciel *Robotran*.

La méthode proposée s'articule autour de deux modules principaux : le premier, nommé *pyswtools*, est dédié au pré-traitement des données extraites de *SolidWorks*, telles que les géométries, les propriétés de masses et les configurations d'assemblage. Ces données sont ensuite structurées dans un format spécifique à travers un fichier JSON. Le second module, *sw2robotran*, utilise ces données pour générer automatiquement le modèle multicorps (MBS), définissant les différents corps du système, leurs propriétes et les interactions avec les autres composants du système. Ce modèle est ensuite prêt pour une simulation dynamique dans *Robotran*.

Les résultats obtenus démontrent la faisabilité et l'efficacité de cette approche pour la création de modèles multicorps complexes, ouvrant la voie à une automatisation accrue dans le processus de modélisation. Les perspectives d'amélioration incluent l'intégration d'une dimension physique plus approfondie au module, l'élargissement de la compatibilité avec d'autres logiciels de CAO ainsi que l'adaptation à une plus grande variété de configurations mécaniques. Ces améliorations visent à enrichir l'expérience utilisateur et à étendre les capacités de la méthode, en la rendant plus polyvalente et adaptée aux besoins des applications industrielles.

En conclusion, ce travail constitue une avancée significative dans le domaine de la simulation dynamique en simplifiant le processus de modélisation des systèmes multicorps et en augmentant l'accessibilité des outils de simulation pour les ingénieurs et chercheurs. À terme, la possibilité d'intégrer directement de cette méthode au logiciel *Robotran* pourrait offrir une solution complète pour l'analyse et l'optimisation des systèmes mécaniques complexes.

Abstract

In this work, we have developed an automated method for creating multibody models in the *Robotran* software from assemblies designed in *SolidWorks*, a Computer-Aided Design (CAD) software widely used in industry. The main objective of this method is to facilitate and accelerate the conversion of CAD models into dynamic models that can be utilized in *Robotran*.

The proposed method revolves around two main modules : the first, named *pyswtools*, is dedicated to preprocessing data extracted from *SolidWorks*, such as geometries, mass properties, and assembly configurations. This data is then structured into a specific format through a JSON file. The second module, *sw2robotran*, uses this data to automatically generate the multibody system (MBS) model, defining the different bodies of the system, their properties, and interactions with other components. This model is then ready for dynamic simulation in *Robotran*.

The results obtained demonstrate the feasibility and efficiency of this approach for creating complex multibody models, paving the way for increased automation in the modeling process. The identified areas for improvement include the integration of a deeper physical dimension into the module, expanding compatibility with other CAD software, and adapting to a wider variety of mechanical configurations. These improvements aim to enrich the user experience and extend the method's capabilities, making it more versatile and suited to the needs of industrial applications.

In conclusion, this work represents a significant advance in the field of dynamic simulation by simplifying the modeling process for multibody systems and increasing the accessibility of simulation tools for engineers and researchers. Ultimately, the possibility of directly integrating this method into the *Robotran* software could offer a comprehensive solution for the analysis and optimization of complex mechanical systems.

Table des matières

Liste des figures

1 Introduction

1.1 Contexte

La Conception Assistée par Ordinateur (CAO) est une composante essentielle de l'ingénierie moderne, largement utilisée dans les secteurs industriels pour la conception et la modélisation de produits. *SolidWorks*[1] est l'un des logiciels de CAO les plus populaires, offrant des outils robustes pour la modélisation 3D, la création de dessins techniques et la gestion de données produit. Il est particulièrement apprécié pour sa capacité à produire des modèles précis et détaillés de pièces et d'assemblages complexes, ce qui en fait un choix privilégié dans divers secteurs industriels, tels que l'automobile, l'aéronautique, le ferroviaire et la robotique.

Cependant, la modélisation ne suffit pas à elle seule. Dans de nombreux cas, une analyse dynamique détaillée des systèmes mécaniques est nécessaire pour évaluer leur comportement sous diverses conditions dynamiques. C'est ici qu'intervient *Robotran*[2], un logiciel spécialisé dans l'analyse dynamique de systèmes multi-corps. *Robotran* permet de simuler les mouvements, les forces et les interactions au sein de systèmes complexes, fournissant des informations cruciales pour l'optimisation et la validation des conceptions mécaniques.

La première étape dans l'analyse d'un système dynamique avec *Robotran* passe par la création d'un modèle dynamique exploitable par le logiciel, appelé "MBsysPad" ou "Pad"[3]. La conception de ce modèle via un éditeur en 2D met en lumière toutes les interactions entre les différents éléments du système. Le processus actuel de création de ce modèle à partir de fichiers *SolidWorks* est souvent manuel et laborieux, nécessitant une intervention considérable pour convertir les données de modélisation en données utilisables pour la simulation dynamique, ce qui limite l'efficacité et la rapidité du processus de conception et de validation.

1.2 Objectif

L'objectif de ce mémoire est de proposer une méthode pour automatiser la conversion des données de *SolidWorks* en modèles dynamiques exploitables par *Robotran*. Cette automatisation présente plusieurs avantages significatifs.

Premièrement, elle permet de gagner un temps précieux. La conversion manuelle des données peut être longue et fastidieuse, particulièrement pour des systèmes mécaniques complexes. En automatisant cette étape, nous pouvons offrir une base de travail aux chercheurs qui pourront se concentrer davantage sur l'analyse du système. De plus, cette automatisation permet de réduire les erreurs humaines en transférant les informations avec précision.

Deuxièmement, une méthode automatisée fournit une première compréhension du système dynamique dès la phase de modélisation. En créant automatiquement un modèle *Robotran* à partir des données de *SolidWorks*, il est possible d'obtenir rapidement une vue d'ensemble des interactions et des comportements du système mécanique, facilitant ainsi les ajustements et les optimisations nécessaires. Cette automatisation contribue donc aussi à enrichir l'outil *Robotran*.

Pour atteindre ces objectifs, le développement est structuré autour de deux modules principaux :

1. **Pré-traitement des données** : nommé *pyswtools*, ce module prend en charge l'extraction et la conversion des données de modélisation de *SolidWorks*. Il s'agit de récupérer les informations pertinentes telles que les géométries, les propriétés des matériaux et les configurations d'assemblage pour les préparer à être intégrées dans *Robotran*. En sortie, ce module rassemble toutes ces informations sous la forme d'un fichier de données au format .json.
2. **Création du modèle dynamique** : nommé *sw2robotran*, ce module utilise les données pré-traitées pour générer automatiquement le modèle *Robotran*. Il s'agit de créer les composants du modèle dynamique et de définir les interactions entre les différents corps. Ce module doit être capable de gérer les différentes contraintes et interactions complexes présentes dans les systèmes mécaniques.

En intégrant ces deux modules, le processus de création de modèles *Robotran* à partir de fichiers *SolidWorks* est entièrement automatisé, allant de l'extraction des données à la génération du modèle dynamique. La Figure 1.1 montre de quelle manière ceux-ci s'ajoutent dans le flux de travail déjà existant avec *Robotran*. Dans les sections 3.2 et 3.3 nous verrons l'implication de chaque module plus en détails.

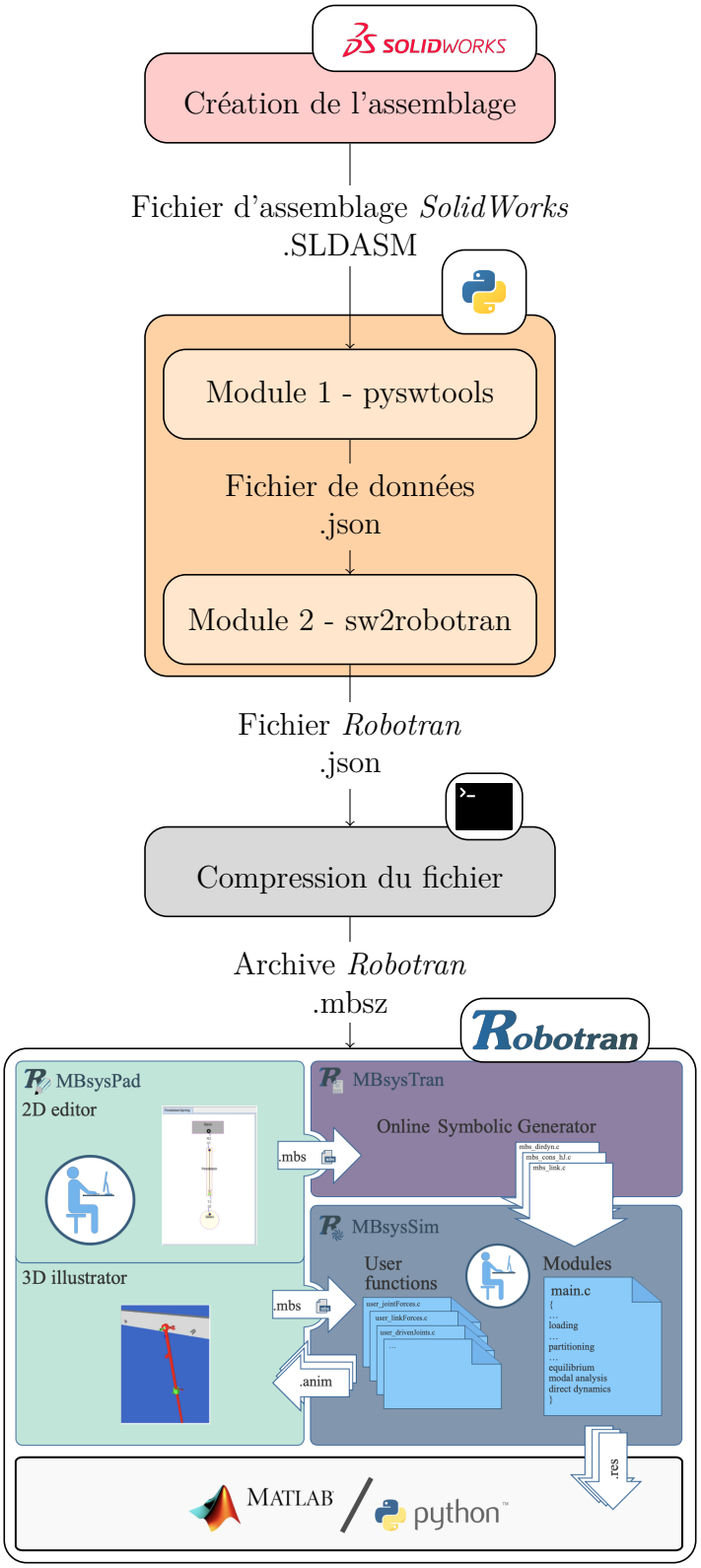


FIGURE 1.1 – Schéma de flux du processus complet

2 Fondamentaux

2.1 SolidWorks

Dans cette section, nous développons tous les éléments nécessaires à la bonne compréhension du logiciel *SolidWorks* et l'utilisation qui en est faite dans ce mémoire. Nous revenons sur les bases de la conception assistée par ordinateur ce qui permet de comprendre pourquoi ce logiciel a été choisi. Nous expliquons le vocabulaire lié au logiciel et à ses outils et nous ne faisons qu'un bref aperçu de son interface. Cela permettra de bien comprendre les notions en lien avec *SolidWorks*. Ensuite, nous expliquons plus précisément l'utilisation de l'API du logiciel, cette compréhension étant notamment importante pour la section 3.2.

2.1.1 Logiciels de Conception Assistée par Ordinateur

La Conception Assistée par Ordinateur est une technologie qui permet aux ingénieurs, architectes et autres professionnels de créer des conceptions précises et détaillées de produits en utilisant des logiciels informatiques. Cette technologie a révolutionné l'industrie en offrant des outils puissants et variés, utilisables tout au long du développement d'un produit, depuis sa modélisation jusqu'au processus de sa fabrication.

Historiquement[4], les premières grandes avancées dans la CAO remontent aux années 1950, lorsque les premières tentatives de conception numérique étaient réalisées avec des macroordinateurs, tant la demande en calcul était grande. Ivan Sutherland[5] est souvent crédité de l'invention de la CAO moderne avec son programme *Sketchpad*, qu'il a développé en 1963 dans le cadre de sa thèse de doctorat au MIT[6]. *Sketchpad* a introduit des concepts tels que l'interaction graphique en temps réel avec un ordinateur, ouvrant la voie à des applications plus sophistiquées.

Dans les années 1970 et 1980, la CAO a commencé à se développer rapidement avec l'avènement de logiciels comme *AutoCAD*, développé par Autodesk en 1982. Grâce à l'introduction des premiers ordinateurs personnels et à la capacité de ce

logiciel à fonctionner dessus, *AutoCAD* est devenu l'un des premiers logiciels de CAO largement adopté, rendant la technologie accessible à un plus grand nombre de professionnels. Il est devenu une référence dans plusieurs domaines de l'industrie, tels que la mécanique et l'architecture. Avec les progrès informatiques, *AutoCAD* a offert de plus en plus d'avantages, représentant un gain de productivité pour les entreprises, notamment grâce à la possibilité de modifier certains aspects de la modélisation sans en changer l'ensemble, ainsi qu'à la capacité de zoomer sur le projet, un changement majeur par rapport à la conception sur papier.
Durant les années qui ont suivi, d'autres logiciels de CAO ont fait leur apparition, chacun offrant des fonctionnalités spécifiques adaptées à divers secteurs industriels. Voici une liste de différents logiciels qui ont vu le jour durant cette période : *MicroStation*, *Cadwork*, *CSG*, *Euclid*, *Pro/Engineer*. Ce dernier marquera un tournant dans la CAO avec l'introduction de la modélisation paramétrique des solides, un type de modélisation qui est maintenant largement utilisé.

Avec le développement de la puissance de calcul et des graphiques en trois dimensions (3D) dans les années 1990, la CAO a franchi une nouvelle étape. En 1994, le logiciel *SolidWorks95* est lancé, permettant de concevoir des dessins techniques de précision ainsi que des ensembles de sous-ensembles 3D de plus en plus complexes. De plus, des outils de fabrication mécanique et de simulation de résistance des matériaux y sont intégrés, en faisant un outil incontournable. En 1997, la firme *Dassault Systèmes*, déjà propriétaire du logiciel concurrent *CATIA*, rachète *SolidWorks*.

Depuis les années 2000, la CAO a évolué grâce à l'amélioration des logiciels et du matériel[7]. Le portage des grands logiciels de 3D sur Windows NT a contribué à leur démocratisation, permettant la création de bibliothèques 3D gratuites en ligne. En 2008, *CATIA V6* a révolutionné le domaine avec sa maquette numérique fonctionnelle, tandis que l'open source a popularisé *Blender*. Par la suite, l'avènement du cloud a permis le travail collaboratif à distance et des logiciels comme *Onshape* ou encore *Fusion 360* ont intégré cette technologie à leur environnement. Parallèlement à cela, l'impression 3D devient de plus en plus accessible, et avec elle, les techniques de prototypage évoluent, rendant les outils CAO indispensables pour transformer la production de prototypes en une activité plus rapide et précise. L'avenir de la CAO pourrait être marqué par l'informatique quantique et la réalité virtuelle augmentée, offrant des possibilités de conception encore plus avancées, permettant aux ingénieurs et designers de créer des produits complexes avec une efficacité et une précision inédites.

Bien que les logiciels de *CATIA* ou *Fusion 360* deviennent de plus en plus populaires, *SolidWorks* reste donc encore aujourd'hui une référence, largement

utilisée dans une grande variété de domaines industriels comme l'aéronautique, l'automobile, la construction, l'architecture et l'électronique. Il est notamment apprécié pour ses fonctionnalités robustes et sa capacité à s'adapter aux besoins complexes de la conception moderne.

2.1.2 Utilisation du logiciel

Le but ici n'est pas de rentrer dans les détails de ce qu'il est capable de faire grâce à *SolidWorks* mais bien de fournir toutes les informations nécessaires à la bonne compréhension de son usage.

Tout d'abord, ce logiciel possède 3 types de fichiers propriétaires :

- Les fichiers de dessins techniques (ou "drawings" en anglais) (`.SLDDRW`) sont utilisés pour créer des représentations 2D des pièces et/ou assemblages. Cela permet de concevoir rapidement et précisément des documents avec les spécificités techniques requises.
- Les fichiers de pièces (ou "part" en anglais) (`.SLDPRT`) contiennent toute la géométrie et les propriétés d'un objet 3D individuel. Ces fichiers sont fondamentaux dans le processus de conception tant ils facilitent la conception, modification et analyse d'une pièce 3D.
- Les fichiers d'assemblage (ou "assembly" en anglais) (`.SLDASM`) reprennent des combinaisons de fichiers `.SLDPRT` ou d'autres fichiers `.SLDASM` afin de créer un modèle complexe. Ces fichiers permettent de définir les relations de position et d'assemblage entre les pièces et les sous-assemblages.

Dans ce travail, nous parlons principalement des fichiers d'assemblage `.SLDASM`. Dans ceux-ci, nous retrouvons donc différentes pièces (`.SLDPRT`) qui sont organisées et liées ensemble à l'aide de ce que l'on appelle des contraintes (ou "mates" en anglais).

Les **contraintes** sont des relations qui régissent la manière dont certaines pièces interagissent entre elles, elles imposent des restrictions sur la position, l'orientation ou le mouvement relatif d'une pièce. Il existe une grande variété de contraintes dans *SolidWorks* mais nous ne traitons que les contraintes dites standard qui sont catégorisées par un **type** spécifique qui déterminera la nature de la relation entre les pièces : coïncidence, coaxialité, parallélisme, perpendiculaire, distance, angle, tangence et blocage [8].

Une contrainte est également associé à deux **entités** (ou "entities" en anglais). Une entité peut être une surface, une arête, un point, un plan ou encore un axe de révolution et est souvent rattachée à une pièce. Chaque contrainte agit donc entre deux entités en fonction du type de relation souhaité.

La figure suivante nous montre l'interface du logiciel lors de l'ouverture d'un fichier d'assemblage. Nous pouvons voir au milieu, la zone graphique avec une vue 3D sur l'assemblage ainsi que les différentes pièces et sur la gauche l'arbre de création (ou " FeatureManager" [9]) qui offre une vue d'ensemble sur les caractéristiques de l'assemblage, les pièces et les contraintes qui lient ces pièces.

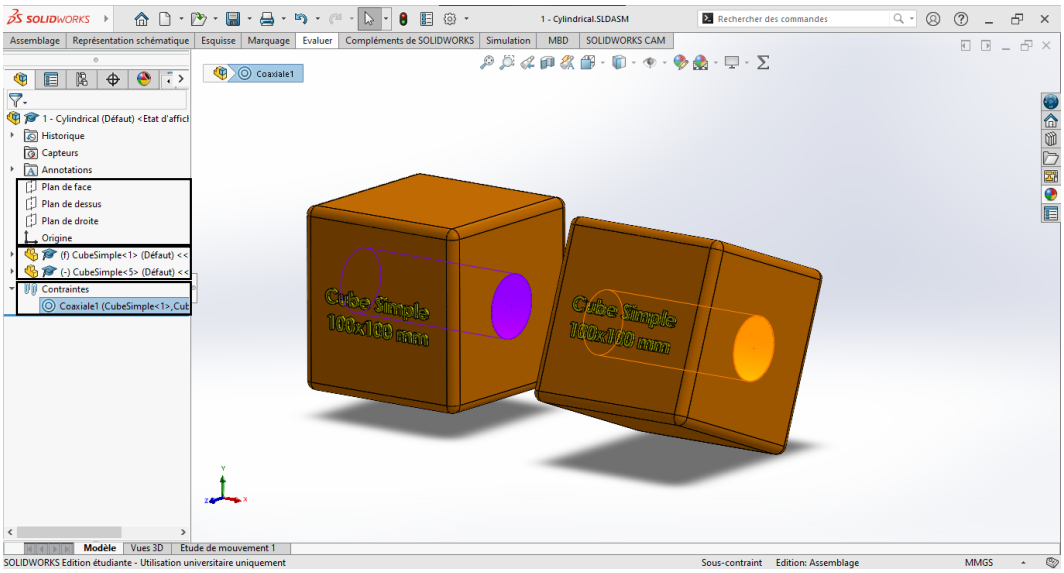


FIGURE 2.1 – Mise en évidence des informations utiles dans l'interface utilisateur d'un assemblage *SolidWorks*

Pour résumé, cette figure nous montre donc un **assemblage** qui contient deux **pièces**. Ces pièces sont mises en relation avec une **contrainte** de coaxialité entre les deux **entités** que sont les surfaces cylindrique misent en évidence.

2.1.3 API SolidWorks

Une API (Application Programming Interface) est un ensemble d'outils et de protocoles, souvent liés à un logiciel, qui permettent à un programme de communiquer avec un autre[10, 11].

Dans le contexte de *SolidWorks*, l'API permet aux utilisateurs de contrôler le logiciel grâce à des lignes de code, en accédant à certaines fonctionnalités comme si l'utilisateur interagissait directement avec l'interface graphique. Cela représente une bibliothèque d'outils puissants qui permettent d'automatiser certaines tâches ; ce qui est particulièrement utile pour des opérations répétitives ou lorsque nous souhaitons extraire des données spécifiques d'un modèle 3D.

L'*API SolidWorks* est large et complexe mais l'usage que nous en faisons réside principalement autour des **Interfaces** et **Members**.
Une **Interface** peut être considérée comme un objet, tel que défini dans les langages de programmation orientés objet là où un **Member** reprend les propriétés (**Properties**) et les méthodes (**Methods**) de ces objets.

Dans ce mémoire, nous nous intéressons principalement aux "interfaces" et "members" des fichiers d'assemblage[12, 13], des pièces[14, 15], des contraintes[16, 17], des entités[18, 19] et les propriétés de masses[20, 21]. Nous utilisons tous ces outils afin d'extraire des données d'assemblage de manière automatisée, ce qui permet de traiter efficacement des modèles complexes. Cette utilisation sera détaillée dans la section 3.2.

2.2 Robotran

Robotran est un logiciel qui a été développé à l'UCLouvain par le MEED[2]. Il permet de modéliser, de simuler et d'analyser des systèmes multicorps (MBS). Il est capable de générer les équations du mouvement associées à une large variété de systèmes dynamiques dans des domaines variés comme la robotique, l'automobile, la biomécanique, etc.

Pour générer ces équations souvent complexes, *Robotran* utilise une approche symbolique qui permet de simplifier les expressions mathématiques. Sa conception est basée sur 3 grandes étapes, comme présenté sur la Figure 2.2, avec d'abord un éditeur graphique 2D du MBS (MBsysPad), ensuite un traducteur symbolique qui permet de générer les équations du mouvement (MBsysTrans) et enfin une simulation 3D qui interprète ces relations (MBsysSim)[22].

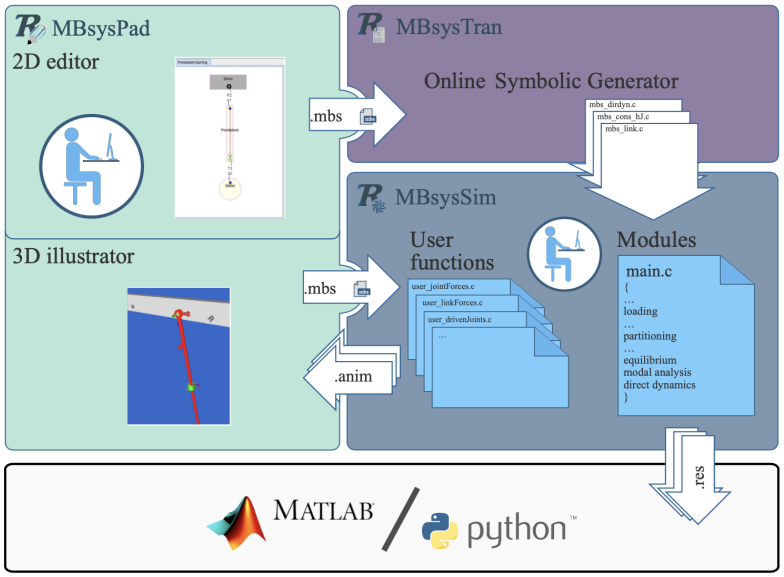


FIGURE 2.2 – Schéma d'utilisation de *Robotran*. Source : [2].

Il est donc évident que l'étape initiale dans l'analyse d'un MBS avec *Robotran* est la conception topologique de ce système dans l'interface 2D, MBsysPad. Cette étape fastidieuse demande beaucoup de recherches et de documentations sur le système à analyser afin de pouvoir judicieusement intégrer toutes les informations au MBS selon les conventions de *Robotran*.

2.2.1 Système Multicorps dans *Robotran*

Pour bien comprendre ce qu'est un système multicorps (MBS) dans *Robotran*, il est essentiel de maîtriser les conventions et le vocabulaire qui lui sont associés. Un MBS est une collection de corps rigides interconnectés, pouvant se mouvoir les uns par rapport aux autres. Dans *Robotran*, ces corps sont supposés être rigides, ce qui signifie qu'ils ne se déforment pas sous l'effet de forces extérieures. La modélisation d'un MBS repose sur une représentation topologique précise, suivie par une définition détaillée de ses composants, notamment les corps, les articulations, et les points d'ancrage.

Tout d'abord, il est important de définir la structure générale d'un MBS dans *Robotran*, à savoir sa **topologie**. La topologie d'un MBS décrit la manière dont les corps sont connectés entre eux. Dans *Robotran*, la structure topologique est représentée sous forme d'arbre, c'est-à-dire une configuration dans laquelle il n'y a pas de boucles de corps. Cette représentation en arbre est simple et efficace pour de nombreux systèmes, mais elle devient insuffisante pour modéliser un grand

nombres d'autres systèmes, qui incluent des boucles fermées, aussi appelées **boucles cinématiques**.

Les boucles cinématiques se produisent lorsqu'une série de corps et d'articulations forment un chemin fermé. Pour traiter ces boucles, *Robotran* emploie une technique appelée **coupure**, qui consiste à éliminer temporairement certaines contraintes pour transformer la structure en une configuration en arbre tout en préservant les interactions physiques initiales du système. Cette procédure de coupure est cruciale pour permettre une analyse dynamique correcte du MBS, notamment lorsque l'on utilise des méthodes numériques pour résoudre les équations de mouvement. La Figure 2.3 nous donne un aperçu visuel de ces deux types de structure. A titre d'exemple, un mécanisme bien connu, le quatre-barres, est un système qui contient une boucle fermée.

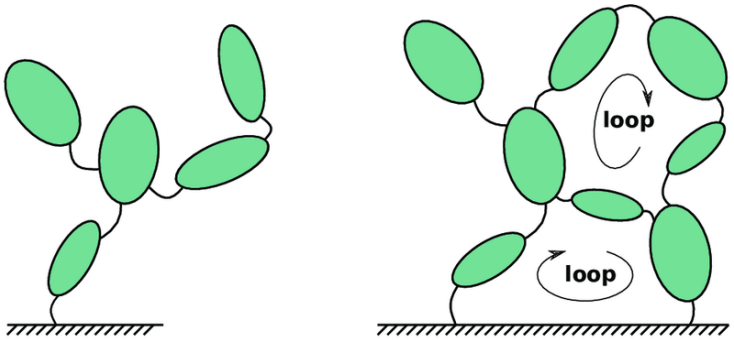


FIGURE 2.3 – Topologies MBS en arbre (à gauche) et avec boucles fermées (à droite). Source : [23].

Un MBS dans *Robotran* est constitué de plusieurs éléments fondamentaux qui interagissent pour décrire le mouvement et les forces au sein du système. Les **corps** sont les entités physiques principales du système. Chaque corps, i, est défini par une masse (m^i), la position d'un centre de masse (CM^i), et une matrice d'inertie ($\mathbf{I}^i$). Chaque corps possède son propre repère, $\{\hat{\mathbf{X}}^\mathbf{i}\}$, qui est défini par rapport à l'élément parent du corps, généralement une articulation.

Les corps sont connectés entre eux par des **articulations**, qui sont les éléments de liaisons permettant des mouvements relatifs dans le MBS. Une articulation est décrite par des coordonnées relatives, notées q. Elle possède un unique degré de liberté (DDL), que ce soit en translation ("T") ou en rotation ("R"), selon les trois axes du repère de l'élément parent ($\hat{\mathbf{e}}^i$, $\hat{\mathbf{e}}^j$, et $\hat{\mathbf{e}}^k$). La force des articulations à 1 DDL réside dans le fait qu'un agencement judicieux de celles-ci permet d'obtenir tout type d'interaction entre les corps. Une articulation peut être dite contrôlée, c'est-à-dire qu'elle n'est pas libre mais obéit à des règles spécifiques qui lui sont imposées.

En complément de ces corps et de ces articulations, les **points d'ancrage** jouent également un rôle important dans la modélisation d'un MBS. Un point d'ancrage est un point défini par une position fixe dans un corps. Chaque point, noté j, est représenté par un vecteur $\mathbf{d}^j$ dans le repère local du corps auquel il est attaché. Grâce au caractère rigide des corps, ces points restent fixes par rapport au corps, même lorsque celui-ci se déplace dans l'espace. Les points d'ancrage sont souvent utilisés pour indiquer les emplacements précis où les articulations ou d'autres éléments du système sont connectés au corps, assurant ainsi une représentation précise des interactions au sein du MBS.

La procédure de coupure précédemment évoquée est une étape indispensable pour traiter les boucles cinématiques dans un MBS. Elle consiste à modifier la topologie du système en supprimant certaines articulations dans MBsysPad pour récupérer une structure en arbre. Ces coupures suppriment donc des interactions entre les corps. Mais afin de maintenir les interactions initiales du système ,des contraintes de boucles ont été ajoutées. Ces contraintes permettent de maintenir les interactions initiales du système et d'un point de vue mathématique elles sont notées, $h_l(q)$. Cette transformation est indispensable pour le calcul des équations du mouvement.

Dans *Robotran*, il existe trois types de coupures, chacune adaptée à des situations particulières :

- **Coupure d'un corps (Solid Cut)** : cette méthode consiste à diviser un corps en deux parties distinctes. Ce type de coupure est utilisée lorsque la boucle cinématique est formée principalement autour d'un corps unique. Ce corps est divisé en deux parties souvent définies comme "l'original" et "l'ombre" où la deuxième représente en réalité un corps fictif.
- **Coupure en boule (Ball Cut)** : c'est une articulation spécifique qui est supprimée pour rompre la boucle cinématique. Cette coupure représente une articulation sphérique entre deux points spécifiques et impose que ceux-ci coïncident en tout temps. Cette articulation sphérique est considérée comme idéale, il n'y a donc pas de transmission de couple.
- **Coupure en bielle (Rod Cut)** : cette coupure s'applique sur un corps afin d'éliminer la boucle cinématique. Elle représente un corps de longueur fixe, considéré sans masse ni inertie, qui est lié à ses extrémités par deux articulations sphériques idéales. Ce type de coupure permet principalement de représenter des barres de connections comme des bielles.

La création d'un modèle multicorps cohérent dans *Robotran* nécessite une compréhension approfondie de la topologie des systèmes, des éléments constitutifs tels que les corps et les articulations, ainsi que de la gestion des boucles cinématiques.

L'utilisation de coupures constitue une approche essentielle pour aborder les défis associés aux boucles fermées en modifiant la structure du système, ce qui simplifie les calculs dynamiques. Grâce à ces concepts et outils, *Robotran* permet la modélisation de modèle MBS robuste en vue de simulations plus complexes, facilitant ainsi l'analyse de leur comportement dynamique dans des contextes scientifiques et industriels variés.

2.3 Cahier des charges

Afin de définir ce modèle selon les objectifs que nous nous sommes fixés dans la section 1.2, il était important d'établir un cahier des charges qui reprenait la portée fonctionnelle du module. D'abord, nous avons défini la portée fonctionnelle du module, en précisant clairement les cas que celui-ci était capable de gérer. Cela nous a permis d'aboutir à une table de correspondance à utiliser comme base dans la conception d'un assemblage. Ensuite, étant donné la complexité et la diversité d'utilisation de *SolidWorks*, nous avons fixé des restrictions sur la conception des assemblages via ce que nous avons appelé les règles de conception (sect. 2.3.2).

2.3.1 Portée fonctionnelle

Afin de définir cette portée nous avons suivi la démarche du module "*MATLAB - Simscape Multibody Link*" [24] qui consiste à donner l'ensemble des relations cinématiques (contraintes) interprétables par le modèle. Celles-ci sont représentées sous la forme d'une table de validités, table 2.2, qui peut servir de référence pour déterminer si un assemblage *SolidWorks* est interprétable ou non par le module. Cette table fait le lien entre les contraintes du côté *SolidWorks* et les articulations élémentaires de translation et rotation du côté *Robotran*.

Comme précédemment expliqué à la section 2.1.2, une contrainte *SolidWorks* est définie par un **type** ainsi que **deux entités** auxquelles elle se rapporte. La table suivante reprend à gauche les contraintes et à droite les entités qui sont correctement interprétables par le modèle.

Contraintes	Entités
• Coïncidence • Coaxiale • Parallèle	• Point • Axe • Plan • Cylindre • Cercle

TABLE 2.1 – Ensemble des contraintes et entités reconnues par le modèle

On peut maintenant définir la table 2.2 qui reprend les contraintes valides selon les paires d'entités.

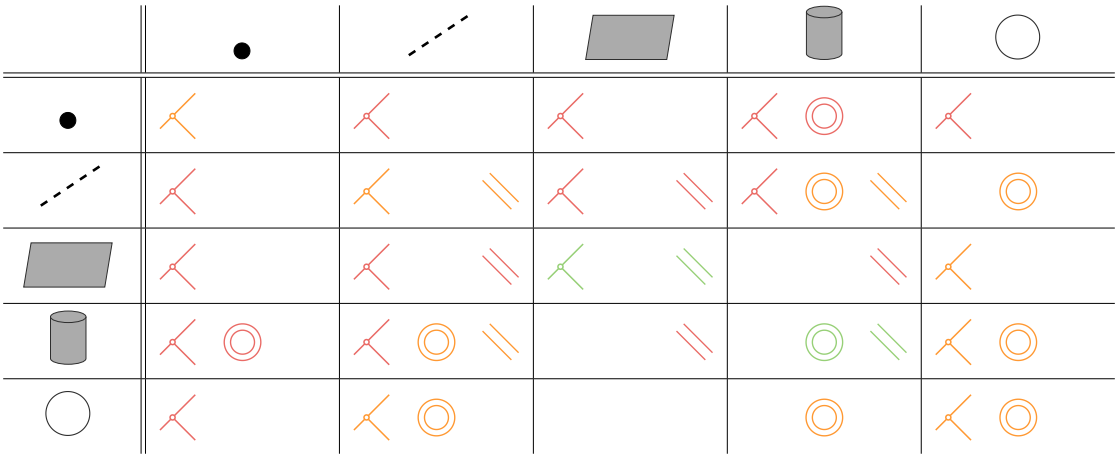


TABLE 2.2 – Table représentant en vert les contraintes validées par le modèle, en orange les contraintes prises en charge par le modèle et en rouge les contraintes qui ne sont pas prises en charge par le modèle mais existantes dans *SolidWorks*

La création de cette table est basée sur la documentation de *SolidWorks* [25, 26, 27]. Nous verrons plus tard (sect. 3.3.5) comment passer de ces contraintes vers des articulations pour *Robotran*.

2.3.2 Règles de conception

Étant donné que l'extraction des données du fichier d'assemblage *SolidWorks* est le point de départ du module, il est important que celui-ci soit bien conçu en suivant certaines règles. Dans le cas contraire, le module ne sera pas en mesure de fournir les informations correctes à la sortie. Voici donc une série de règles à respecter dans la conception de ce fichier CAO :

- Le module n'est pas conçu pour traiter des assemblages qui comportent eux-mêmes des sous-assemblages.
- Deux corps sont liés par un maximum de 2 contraintes.
- Aucun nom de contraintes ou de corps ne doit être identique ou comporter d'espace.
- Les données de masse, de centre de masse et d'inertie d'un corps doivent être remplies à la main dans *SolidWorks* via l'onglet Outils > Évaluer > Propriétés de masse.

3 Méthodologie

3.1 Introduction

Le but de ce chapitre est de parcourir étape par étape le processus qui a mené à la conception des deux modules.
Le premier, appelé *pyswtools*, reprend l'ensemble des interactions directes avec le logiciel *SolidWorks*. Cela se fait à travers l'utilisation de l'*API SolidWorks* afin de ressortir toutes les données importantes de l'assemblage. Nous parlons ici des données de masse, centre de masse et matrice inertie pour chaque corps mais aussi et surtout les différentes relations entre les corps appelées contraintes qui pourront ensuite être interprétées et comprises par *Robotran*. Ce module prend donc en entrée un chemin vers un fichier d'assemblage *Solidworks* (`.SLDASM`) et il renvoie en sortie une fichier de données au format json qui reprend les données extraites. Le second, appelé sw2robotran, récupère les informations dans le fichier de données venant du module 1 pour les traiter et produire un fichier qui représente le système multicorps dans *Robotran*. C'est ce module qui est vraisemblablement le coeur processus. La stratégie ici est d'abord de créer une structure de données qui reprend toutes les informations nécessaires à la conversion vers *Robotran*. Nous voyons comment y interpréter les données de *SolidWorks* afin de les convertir en données nécessaires à *Robotran* et petit à petit raffiner cette structure pour ne garder que les informations essentielles au modèle. A la fin de ce module, nous récupérons donc un fichier json qui contient l'ensemble du système multicorps.

Il est important de préciser que les deux modules nous donnent des fichiers au format identique (json) mais en réalité bien différents. Le fichier en sortie du module 1 permet de répertorier les informations de l'assemblage *SolidWorks* dans un format facile à manipuler, nous l'appelons le fichier de données. En revanche, le fichier de sortie du module 2, représente le système multicorps et est écrit de manière à respecter le format lisible par *Robotran*, nous l'appelons simplement le fichier *Robotran*.

Dans les sections suivantes, nous développons les différentes étapes de ce module dans l'ordre effectué par le code. La figure 3.1 montre les différentes étapes.

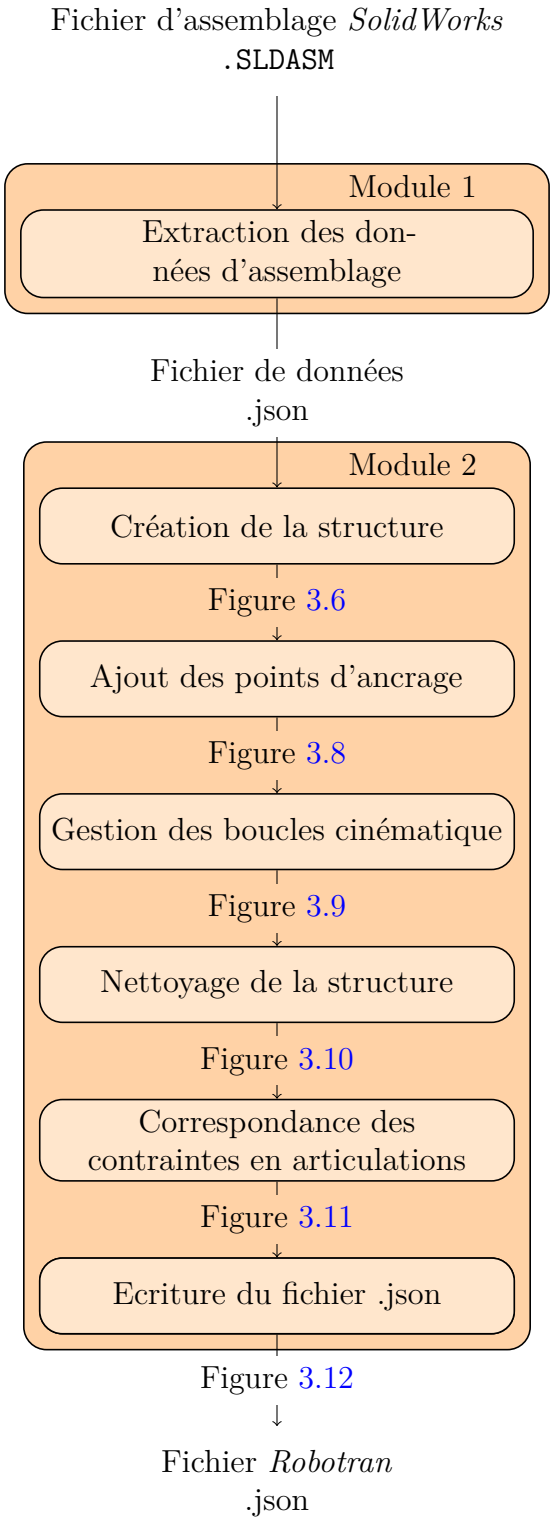


FIGURE 3.1 – Schéma de flux du processus dans les modules 1 et 2

3.2 Module 1 - pyswtools

Dans cette section nous allons parcourir les différentes étapes du module 1. Pour rappel, ce module reçoit en entrée le chemin vers un fichier de type assemblage *SolidWorks*. Avant de pouvoir extraire des données de l'assemblage, il faut lire ce fichier. Étant donné que celui-ci est dans un format propriétaire au logiciel *SolidWork* (`.SLDASM`), il faut le lire avec le logiciel lui-même. Pour ce faire nous avons utilisé une librairie python qui permet d'exécuter une application dans le système[28, 29]. C'est vraiment à partir de ce moment que nous avons utilisé l'*API SolidWorks* pour extraire les données souhaitées.

La première étape consistait à parcourir l'ensemble des corps dans l'ordre de l'arborescence du *Feature Manager*. Pour obtenir cet ordre spécifique nous avons dû faire appel à une méthode particulière qui récupèrait un élément grâce à son indice unique[30]. Cet ordre était important pour le traitement des données dans le module 2 car il nous assurait que le corps fixe au sommet de l'arborescence était bien le premier corps traité. Nous avons pu ainsi parcourir l'ensemble des corps de l'assemblage et pour chacun d'eux en extraire les données importantes.

Les premières informations que nous avons obtenues étaient toutes celles qui étaient liées aux propriétés de masse de la pièce. Parmi elles, nous avions bien évidemment la masse du corps mais aussi son centre de masse et sa matrice d'inertie. Dans *Robotran*, les coordonnées du centre de masse sont exprimées dans le repère du corps et la matrice d'inertie est calculée par rapport au centre de masse et alignée avec le repère du corps. Il est important de tenir compte de ces alignements et points de référence pour directement extraire les bonnes informations grâce à l'*API SolidWorks*[31, 32, 33].

Un autre aspect important est de pouvoir situer chaque corps dans l'espace et plus précisément situer l'origine du repère de chaque corps par rapport à l'origine du repère de l'assemblage. Cela nous a été utile notamment pour exprimer les coordonnées des points d'ancrages dans le repère d'un corps plutôt que par rapport à l'origine de l'assemblage. Une fonction pré-existante nous permet d'extraire la matrice de transformation homogène qui aligne le repère de l'assemblage avec le repère d'un corps [34]. Ce type de transformation est beaucoup utilisée en robotique et permet de calculer un changement de repère en une seule opération matricielle. Ces matrices de transformation homogène (**H**) sont de taille 4x4 et contiennent, une matrice de rotation(**R**) qui lie les deux bases des repères et un vecteur de translation (**x**) qui lie les deux origines des repères. Voici à quoi ressemble ce type de matrice en toute généralité :

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{x} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & x_1 \\ R_{21} & R_{22} & R_{23} & x_2 \\ R_{31} & R_{32} & R_{33} & x_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

Les dernières données que nous avons dû récupérer étaient toutes celles qui nous fournissaient des informations sur les interactions entre les corps, à savoir les contraintes. Pour chaque corps parcouru, nous avons regardé les contraintes qui lui étaient liées pour en extraire des informations sur le type de contraintes[35] et sur les entités qui étaient liées par cette contrainte. Une entité est également caractérisée par un type mais aussi et surtout par des paramètres[36]. Ceux-ci nous ont été très utiles par la suite pour interpréter les contraintes en articulations.

A la fin de ce processus, l'ensemble de ces données ont été écrites selon un certain format dans le fichier de données json qui a été notre base de travail pour le module 2.

3.3 Module 2 - sw2robotran

C'est ce module qui fait le lien entre les données venant de *SolidWorks* et permet de les rendre cohérentes au regard de *Robotran*.

C'est donc dans cette partie que nous voyons la conception du système multicorps notamment via l'interprétation des relations cinématiques entre les corps ainsi que la gestion des boucles cinématiques qui représentent deux aspects importants du développement.

3.3.1 Choix d'une structure de données

Avant de pouvoir travailler sur les données extraites de *SolidWorks* grâce au module 1, il était important de réfléchir à la structure de données qui nous a permis de stocker ces informations. Celle-ci était notre base de travail tout au long du processus, il était donc capital de la construire intelligemment pour n'oublier aucune information afin de faciliter l'accès à celle-ci.

Pour le choix de cette structure, nous nous sommes dirigés vers une structure de type graphe pour deux raisons principales.

D'une part, *SolidWorks* et surtout *Robotran* représentent tous deux des systèmes d'assemblage et multicorps via une arborescence de corps liés par des contraintes

ou des joints. Ce type de structure peut s'apparenter à des graphes. Par conséquent, le maintien d'une structure en graphe nous évite des étapes intermédiaire de restructuration de l'information lors du passage d'un logiciel à l'autre.

D'autre part, ce type de structure permet de simplifier grandement une étape importante du module : la détection des boucles cinématiques. En effet, la suppression de ces boucles est fondamentale dans *Robotran*, l'avantage des graphes est qu'ils offrent une grande variété d'algorithmes selon l'application voulue. La détection des boucles est devenue dès lors une tâche presque triviale grâce à un algorithme bien connu appelé "Depth First Search" (DFS)[37]. Nous aurons l'occasion de développer cela dans la section 3.3.4, dédié aux boucles cinématiques.

Concrètement, ce module nous a permis de faire le passage d'un fichier de données brutes à un graphe dirigé acyclique (DAG) [38].

Pour la conception de cette structure de données, nous avons exploité les avantages du langage python et plus précisément du caractère orienté objet de ce langage. Cela nous a permis de créer des classes où chacune des instances, appelée objet, a pu stocker des informations propres à l'objet en question ; ce sont les variables de classe.

Nous avons créé les classes suivantes : corps (Body), contraintes (Mate), entités (Entity), articulation (Joint), point (Point) et coupures (Cut). Celles-ci vont être expliquées dans les paragraphes qui suivent.

Par souci de compréhension avec le code, nous parlons d'un objet corps, contrainte, entité, articulation, point ou coupure pour faire référence à la classe ou une instance de la classe Body, Mate, Entity, Joint, Point ou Cut respectivement.

Classe Body

Cette classe comprend les corps présents dans l'assemblage et par la même occasion les éléments principaux de notre structure de données. Un corps est caractérisé par un nom et un parent, comme expliqué dans la section 2.3.2, ce nom ne peut pas être identique d'un corps à l'autre et il ne peut pas contenir d'espace. Chaque instance de cette classe contient des informations de masse, de centre de masse et d'inertie du corps en question mais aussi la matrice de rotation et le vecteur de translation entre le repère du corps et le repère origine de l'assemblage. Nous avons aussi différentes listes qui reprennent les contraintes, les points, les joints et les voisins associés à ce corps. Cette dernière liste est très importante car c'est grâce à elle que nous avons pu parcourir l'ensemble de la structure par la suite.

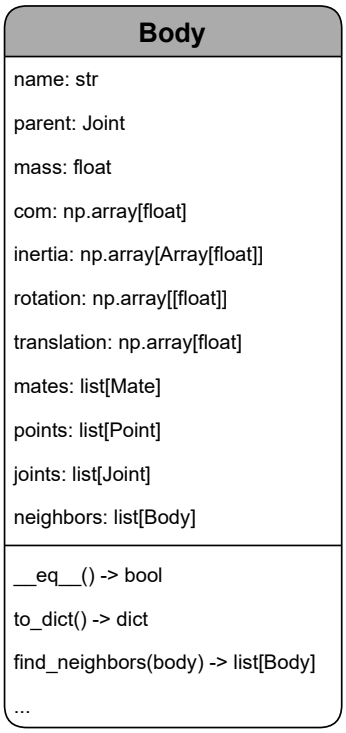


FIGURE 3.2 – Diagramme de structure de la classe `Body`

Classe Mate et Entity

Ces deux classes reprennent les données de relation entre les corps qui viennent de *SolidWorks*.
D'un côté, nous avons les contraintes. Une contrainte est caractérisée par un nom, un type, deux entités auxquelles elle se rattache ainsi qu'un lien vers un corps enfant. Au même titre qu'un corps, le nom d'une contrainte doit respecter certaines règles établie dans la section 2.3.2. Un champ important est celui du `type` qui nous permet de savoir si une contrainte est soit de coïncidence, de coaxialité ou parallèle [39].
De l'autre côté, nous avons les entités qui font référence aux éléments sur lesquels une contrainte est applicable. Tout comme les contraintes, le champ le `type` est important et permet de savoir s'il s'agit d'un point, une ligne, un plan, un cylindre ou un cercle[40].

De plus les tableaux `point`, `vector` et `radius` nous ont donné des informations chiffrées sur l'entité [36]. Ce sont informations qui nous ont permis de savoir selon quel(s) axe(s) une contrainte agi(ssen)t pour pouvoir intelligemment la convertir en articulations dans *Robotran*.

J'attire l'attention du lecteur sur les champs `type` de chacune de ces deux

classes qui définissent à eux deux les limites de notre modèle et donc la diversité de la table de correspondances définie dans la section 2.2.

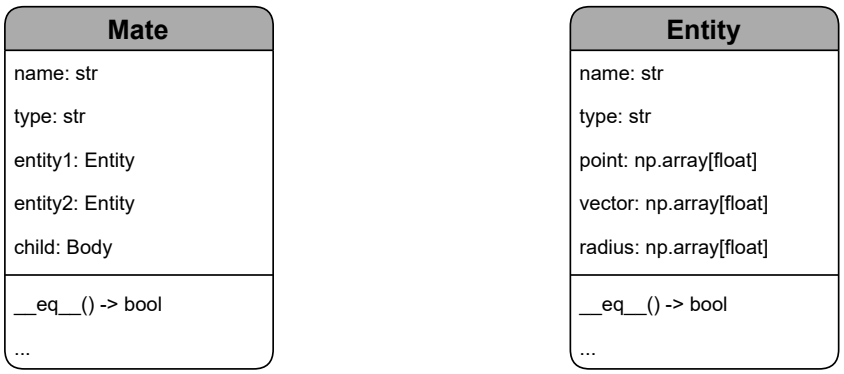


FIGURE 3.3 – Diagramme de structure des classes `Mate` et `Entity`

Classe Joint, Point et Cut

Voici les trois classes qui font référence aux éléments de *Robotran*.

La première classe (Joint), concerne les articulations qui font référence aux transformations élémentaires de rotation et de translation selon les trois axes dans *Robotran*, cet axe qui est fixé par la variable `axis`. Ces objets sont donc principalement caractérisés par un axe et une nature selon que l'articulation soit indépendante ou contrôlée. La variable parent peut faire référence à une autre articulation ou à un corps.

La seconde classe concerne les points. Celle-ci est plutôt simple car elle ne contient que le nom et les coordonnées du point en question.

La troisième classe représente les coupures des objets qui sont utilisées pour couper les boucles cinématiques. Ces dernières seront abordées plus en détails dans la section 3.3.4. Elles possèdent trois natures différentes : solide, boule et bielle. Cette dernière demande également de spécifier une longueur. De plus, cette coupure s'applique à deux points qui sont eux-mêmes rattachés à deux corps.

Il est important de noter que les variables de chacune de ces trois classes réfèrent à un champ obligatoire dans le fichier de sortie json pour *Robotran*. De plus, nous avons pu voir que chacune de ces classes possède une fonction `to_dict()` qui convertit toutes les données de l'objet en un dictionnaire. Cela nous a permis de facilement mettre en page le fichier json final en sélectionnant les champs souhaités.

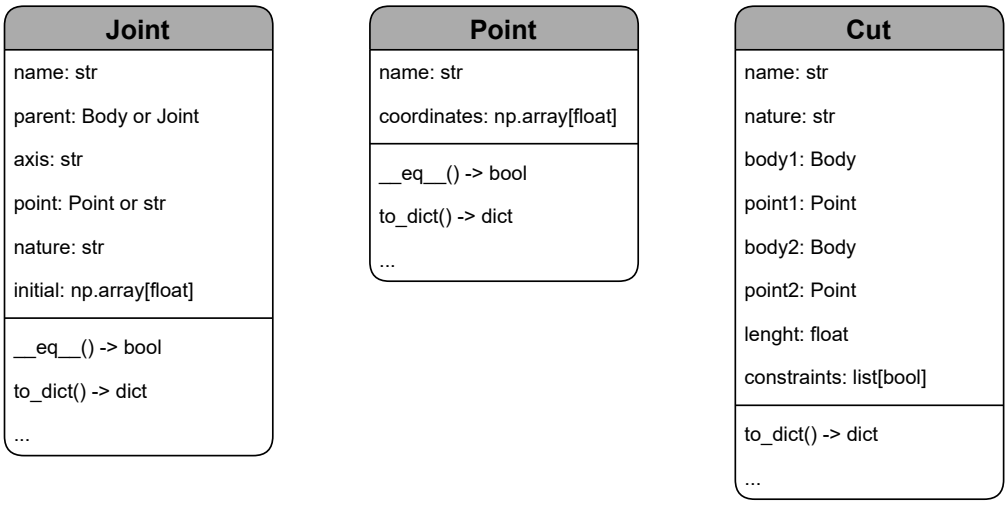


FIGURE 3.4 – Diagramme de structure des classes `Joint`, `Point` et `Cut`

Classe TreeBody

Enfin, voici la classe reprenant l'ensemble de la structure globale qui est l'élément central de notre graphe.

Pour les variables de classe, nous avons une variable `root` qui pointe vers le premier corps de la structure, un nom de projet et une liste reprenant l'ensemble des coupures présentes dans la structure.
Pour les fonctions de classe, d'abord, cette classe implémente les deux fonctions nécessaires au protocole *Iterable* : `__iter__()` et `__next__()`. Cela a été grandement utile pour parcourir facilement l'intégralité de notre structure. Ensuite, nous avions `list_body()` qui renvoyait simplement à une liste des corps de la structure et `find_cycle()` qui a servi plus tard dans la détection des boucles cinématiques, un sujet abordé dans la section 3.3.4. Enfin, les deux dernières fonctions, `remove_redundant_neigh()` et `remove_unused_points()`, ont permis de supprimer des informations inutiles ou redondantes qui ont été créés durant le processus de création de la structure.

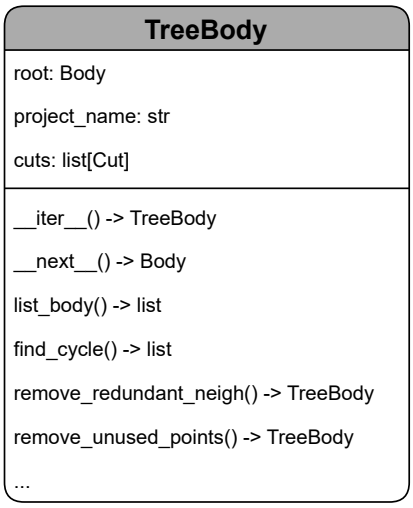


FIGURE 3.5 – Diagramme de structure de la classe `TreeBody`

3.3.2 Lecture du fichier de données

Dés lors que nous avions clairement défini notre structure, nous avons pu commencer à y stocker de l'information. L'idée ici était de lire le fichier produit par le module 1 et d'intégrer à notre structure toutes les caractéristiques des corps ainsi que des contraintes qui les liaient entre eux.

Initialisation des corps

Pour commencer nous avons créé les corps avec leurs propriétés de masse, de centre de masse et d'inertie. C'est également ici que nous avons intégré un vecteur de translation et une matrice de rotation. Ce vecteur et cette matrice nous ont permis de situer le repère du corps par rapport au repère de l'assemblage lui-même. Nous aurons besoin de cela plus tard pour situer les points d'ancrage ainsi que les transformations dans le repère du corps.

Il est important de noter que nous avons fait le choix de toujours créer un corps "Base" qui serait lié par les 6 DDL au premier corps de l'assemblage. Ce choix est lié à une pratique courante dans *Robotran*. De cette manière le corps "Base" peut servir à représenter plusieurs aspects du modèle dynamique comme une trajectoire ou des imperfections sur une route. L'utilisation ou non de ce corps "Base" est laissé à l'utilisateur. Par souci de clarté nous n'avons pas représenté ce corps dans les schémas qui suivent.

Initialisation des contraintes

L'ajout de l'ensemble des contraintes se fait de manière similaire. Nous avons

récupéré pour chaque corps les contraintes qui lui sont propres et nous avons pris en compte qu'une contrainte est bien constituée de deux entités, comme nous l'avons définie dans les section 2.1.2. De plus, sachant qu'une contrainte lie deux corps, nous avons ajouté le corps dit enfant parmi les voisins du corps initial.
Il est important de remarquer que nous avons créé une redondance dans les liens entre les corps. En effet, prenons l'exemple d'une contrainte appliquée entre le corps "C1" et le corps "C2". Pour "C1" cette contrainte est dirigée vers "C2" et pour "C2" elle est dirigée vers "C1" de telle sorte à pouvoir circuler dans les deux sens entre les corps. Ceci était indispensable à ce stade car nous ne connaissions pas la disposition et le sens final des relations entre les corps.

Prenons un exemple composé de 4 corps, la figure 3.6 décrit la forme que aurait pu prendre notre structure à ce stade de développement. Elle est constituée de l'ensemble des corps avec leurs propriétés de masse ainsi que de toutes les contraintes. Les corps qui doivent l'être sont liés entre eux de manière à obtenir un graphe bidirectionnel multiple[41] comme expliqué ci-dessous.

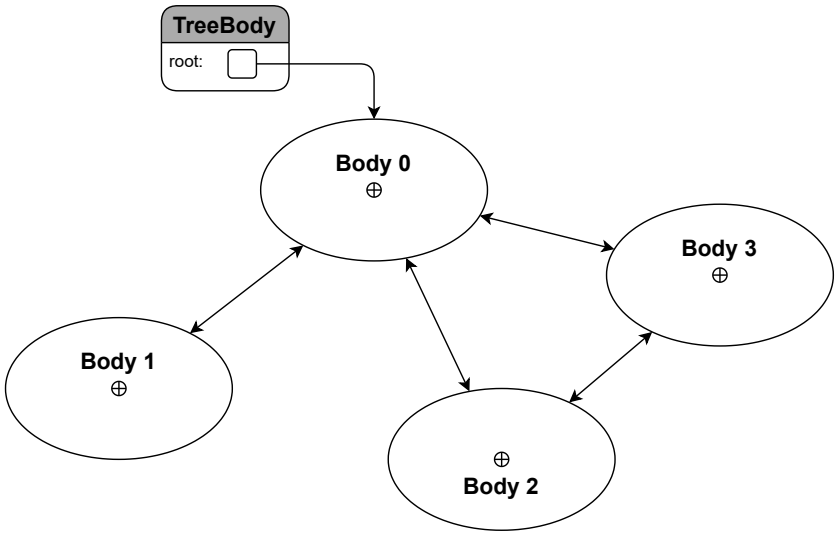


FIGURE 3.6 – Création du système avec les corps et leurs voisins

3.3.3 Ajout des points d'ancrage

Nous avons commencé à convertir les données stockées vers des données utiles dans *Robotran*. C'est ainsi que nous avons pu déterminer les points d'ancrage.
La difficulté ici était d'utiliser l'ensemble des paramètres qui étaient à notre disposition et de définir les coordonnées du point dans le repère du corps. Le schéma suivant permet de mieux représenter les vecteurs qui sont en jeu :

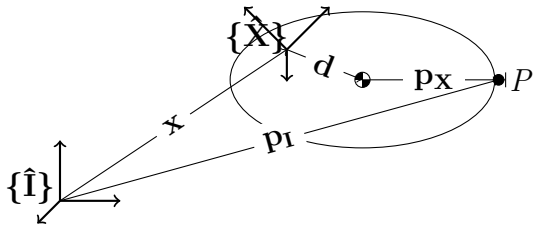


FIGURE 3.7 – Localisations des éléments d'un corps dans le système

Nous avons cherché à obtenir les coordonnées du point P dans la base $[\hat{X}]$, ce qui revenait à trouver $\mathbf{p_X}$. Pour cela nous avons dû exprimer l'ensemble des vecteurs disponibles dans la base $[\hat{X}]$ et judicieusement les sommer entre eux. C'est ici qu'intervient la matrice de rotation du corps telle que $[\hat{X}] = R_X^I[\hat{I}]$.
Nous pouvons donc aboutir à la formule suivante :

$$\mathbf{p_X} = \mathbf{R}_X^I(\mathbf{p_I} - \mathbf{x}) - \mathbf{d} \tag{3.2}$$

Il est important de souligner que nous avons généré des points pour chaque interaction avec un autre corps. Bien que cela ait conduit à un nombre élevé de points, cette approche est bénéfique pour l'étape suivante. Voici donc l'apparence de notre exemple à l'issue de cette étape :

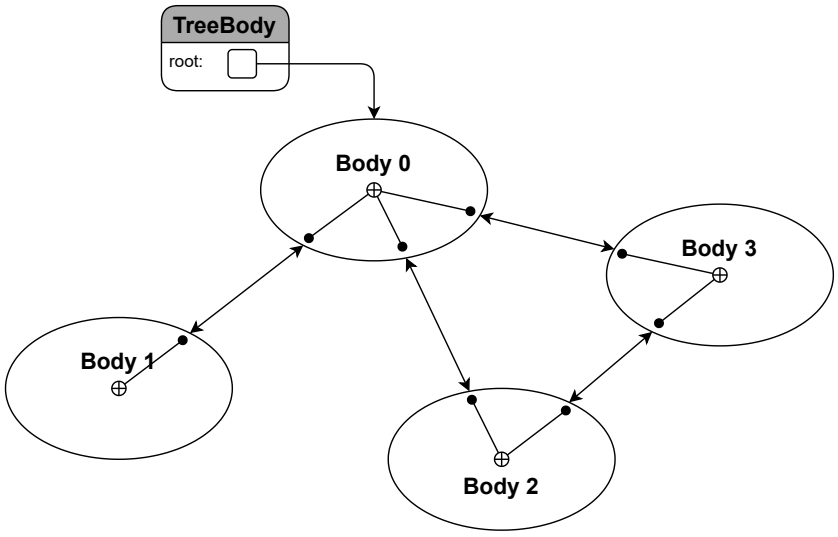


FIGURE 3.8 – Ajout des points d'ancrage au système

3.3.4 Gestion des boucles cinématiques

La suppression des boucles cinématiques est une étape indispensable pour permettre la résolution du système d'équations du modèle *Robotran*. Dans cette

phase, nous avons identifié et traité les cycles présents dans le graphe représentant l'assemblage.

Algorithme Depth First Search Nous allons brièvement expliquer l'algorithme que nous avons utilisé afin de détecter ces boucles. Dans la théorie des graphes, ces boucles sont appelées des cycles et, les noeuds et les arrêtes font référence aux corps et relations entre les corps dans notre cas.
Pour détecter ces cycles, nous avons utilisé l'algorithme de recherche en profondeur (DFS)[37, 42]. Cet algorithme a parcouru le graphe de manière récursive du noeud racine jusqu'à avoir examiné l'ensemble de la structure. Une liste a gardé en mémoire les noeuds déjà visités et lorsque nous avons tenté de revenir à un noeud qui était présent dans cette liste, c'est que nous avions détecté une boucle. Dans ce cas, une coupure devait être créée pour rompre celle-ci. C'est ici qu'intervint l'utilisateur pour déterminer le type de coupure (boule, solide ou bielle) et l'endroit où celle-ci doit être faite dans la boucle.

La figure ci-dessous nous montre qu'un cycle a été trouvé entre les corps 0-2-3 et la boucle a été supprimée via la coupure nommée "Cut1" entre les corps 2 et 3.

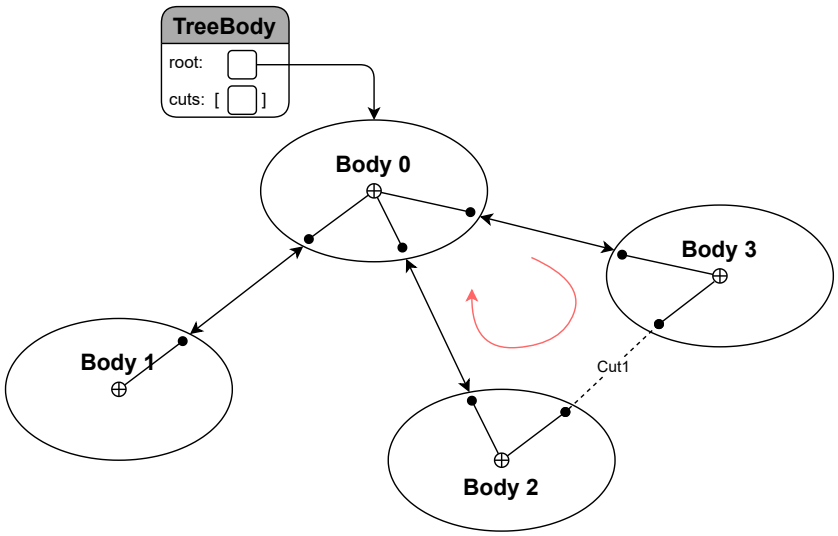


FIGURE 3.9 – Coupure des boucles cinématiques du système

Graphe orienté acyclique

Maintenant que notre graphe ne contenait plus de boucle, il prenait la forme d'un graphe dit *bidirectionnel acyclique*. Il ne nous restait plus qu'à déterminer le sens des relations parent-enfant pour obtenir un *graphe orienté acyclique* aussi connu sous le nom de *DAG*[38]. Cela se fit simplement en parcourant toute la

structure et en retirant les voisins redondants de manière à n'obtenir que des liens unidirectionnels.

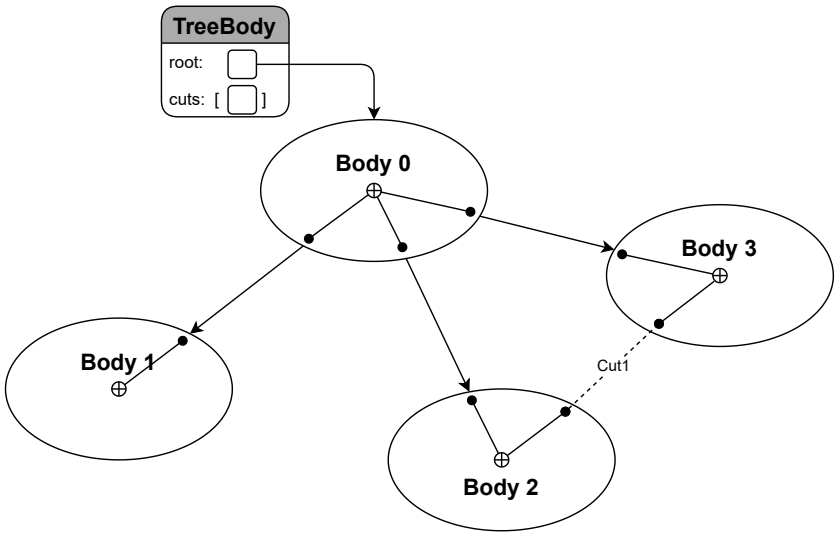


FIGURE 3.10 – Suppression des liens redondants du système

3.3.5 Correspondance contraintes/articulations

C'est durant cette étape qu'intervint l'interprétation dynamique du système. A partir des contraintes entres deux corps, nous avons déterminé la chaîne d'articulations équivalente dans *Robotran*.
Nous avons adopté la même stratégie pour chacun des corps. Initialement, nous supposions qu'un corps était libre par rapport à l'autre, ce qui revenait à dire qu'il est lié par 6 DDL. Ensuite, pour chaque contrainte entre ces deux corps, certains degrés de liberté étaient bloqués pour réduire cette liste. Ce qui nous a permis de finalement obtenir les interactions réelles entre les corps.

Afin de déterminer quels DDL devaient être bloqués, nous devions regarder l'axe référence [1] d'une contrainte. Si nous prenions l'exemple de deux cubes, la liste d'articulations initiale était bien $[T1, T2, T3, R1, R2, R3]$, les 6 degrés de liberté. Si nous ajoutions une contrainte de coïncidence entre deux plans ayant la même normale $\hat{\mathbf{n}}_\mathbf{c} = (1, 0, 0)$, cette contrainte bloquait la translation dans le sens de la normale et les rotations selon les deux axes perpendiculaires à cette normale. Et on obtenait donc la liste $[T2, T3, R1]$.

1. Cet axe s'interprète différemment selon le type d'entité : pour un plan il représente la normale, pour un cylindre il fait référence à l'axe principal de celui-ci [36].

Si nous ajoutions encore un parallélisme entre deux autres plans de normale $\hat{\mathbf{n}}_\mathbf{p} = (0, 1, 0)$, nous bloquions ici uniquement les deux rotations perpendiculaires à la normale. Ce qui donnait donc la liste finale suivante $[T2, T3]$.
Voici comment nous avons été capable de passer d'une interaction à 6 degrés de liberté pour finalement obtenir uniquement 2 degrés de liberté en translation entre ces deux cubes.

Nous avons pu noter une particularité où l'axe de référence n'était pas aligné avec le repère. Dans ce cas, nous avons fixé un repère à cet axe et avons déterminé ensuite la matrice de changement de base entre ce nouveau repère et le repère du corps[43]. Nous avons ensuite décomposé cette matrice de changement de base en trois matrices de rotation distinctes selon les trois axes dont les angles correspondants sont connus sous le nom d'angles de *Tait-Bryan*[44, 45] en cinématique.

Nous avons donc obtenu après cette étape un graphe dirigé acyclique avec une chaîne d'articulations entre chaque corps (représenté en gris sur le schéma).

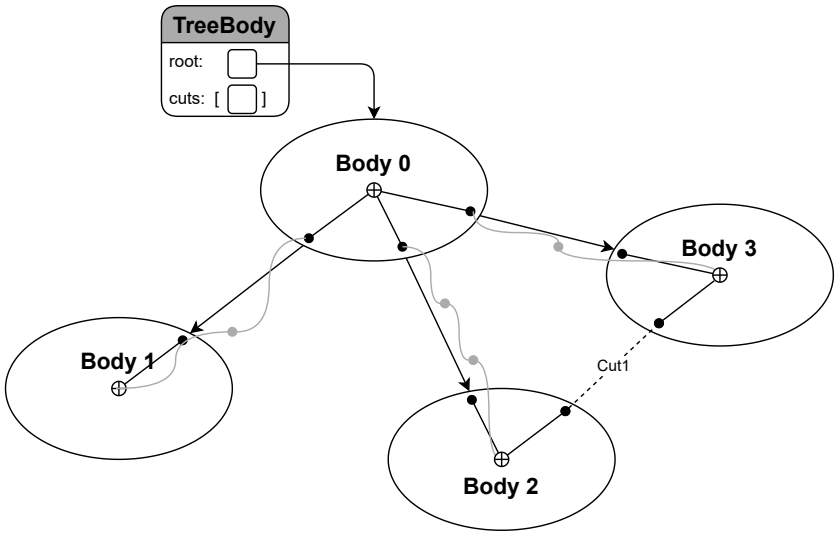


FIGURE 3.11 – Ajout des joints entre les corps du système

3.3.6 Suppression des points d'ancrage inutilisés et écriture du fichier json

Avant de pouvoir écrire le fichier json destiné à *Robotran*, il ne nous restait plus qu'à supprimer les points inutilisés dans la structure. Cette étape, bien que facile à réaliser, était importante car ces points inutiles ne devaient pas se retrouver dans le modèle *Robotran*.

La structure finale devenait donc la suivante :

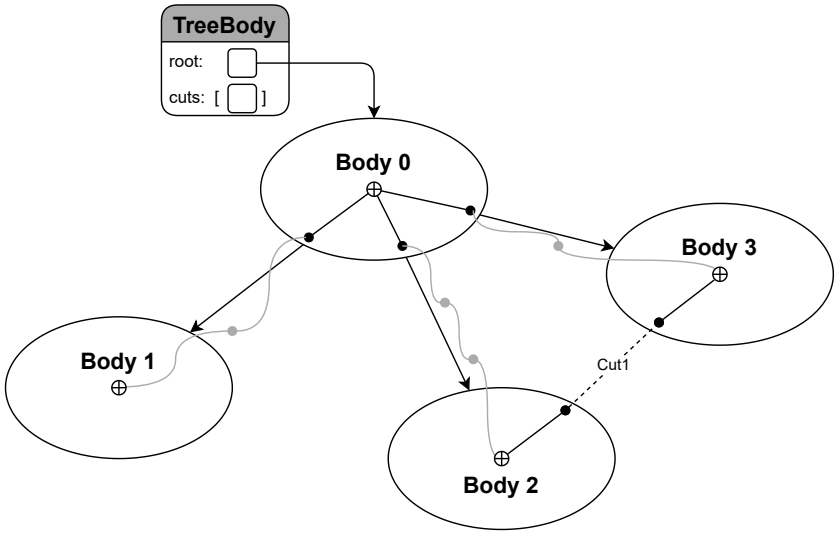


FIGURE 3.12 – Suppression des points d'ancrages redondants du système

Maintenant que la structure était complète nous pouvions reprendre l'ensemble des informations importantes pour le modèle *Robotran* et les rassembler toutes dans un fichier JSON, lequel constitue le fichier final du module 2.

4 Résultats et discussions

4.1 Introduction

Après avoir expliqué la conception du module, il était essentiel de tester ses fonctionnalités. Ce chapitre propose plusieurs modèles de test que le module est capable de prendre en charge, afin de démontrer ses capacités dans la pratique.

Nous avons évalué les performances de notre module à travers 3 benchmarks, chacun d'eux a permis de valider différentes caractéristiques et spécifications. D'un point de vue théorique, cela nous a donné l'occasion de valider plusieurs aspects importants dans la conception d'un modèle *Robotran*. Et d'un point de vue pratique, cela a nous montré la fonctionnalité du modèle avec des applications concrètes qui ont déjà pu être étudiées par l'iMMC.
Ces benchmarks sont présentés dans les sections suivantes, classés du plus simple au plus complexe.

4.2 Validation de géométries simples

4.2.1 Objectifs

Le premier benchmark a pour objectif de valider la capacité des modules à transmettre fidèlement les informations géométriques, massiques et cinématiques dans de scénarios simples. Il s'agissait ici de vérifier l'exactitude des interactions entre deux corps en utilisant des assemblages de base. Cela nous a permis de valider certains éléments de la table de correspondance, table 2.2, en testant plusieurs configurations d'assemblage avec deux cubes identiques. Un cube était fixe, tandis que l'autre était soumis à des contraintes limitant ses degrés de liberté.

La Figure 4.1 nous montre une vision 3D des cubes que nous avons utilisée dans ces assemblages. Ce cube contient un alésage cylindrique en son centre ce qui permet d'y appliquer toutes les types de contraintes, notamment des contraintes

de coaxialité.

FIGURE 4.1 – Cube contenant un alésage cylindrique

Sur base de ce cube, voici les six assemblages testés et les degrés de liberté qu'ils autorisent :

- Cylindrique : permet une translation et une rotation.
- Co-planaire : permet deux translations.
- Planaire : permet deux translations et une rotation.
- Rotation : permet une seule rotation.
- Prismatique 1 & 2 : permet une translation, avec deux configurations différentes.
- Sphérique : permet les trois rotations.

Une visualisation de l'ensemble de ces assemblages est reprise en annexe A où nous détaillons les relations de contraintes et entités.

4.2.2 Résultats

L'ensemble des résultats lié a ce benchmarck sont repris en annexe A mais voici ce que nous avons obtenu pour le premier assemblage. La Figure 4.2 nous montre l'assemblage cylindrique dans *SolidWorks* et met en évidence les DDL résultants de la contrainte de coaxialité entre les deux cylindres. La Figure 4.3 nous montre le modèle résultant dans l'interface web de *Robotran*. Nous pouvons voir comme attendu trois corps (la base et les deux cubes), liés entre eux par des chaînes d'articulations ainsi qu'un point d'ancrage.

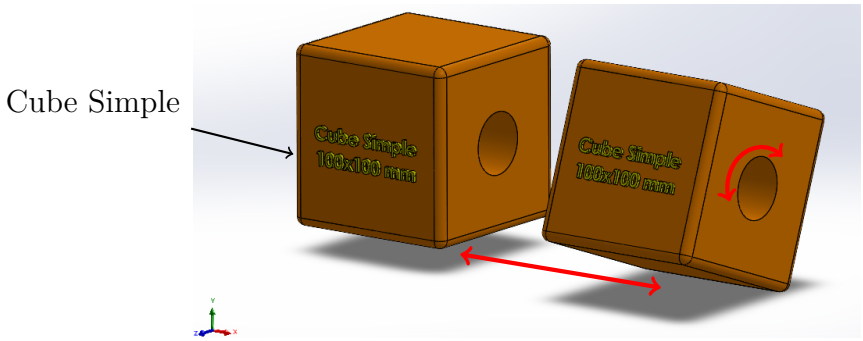


FIGURE 4.2 – Assemblage cylindrique entre deux cubes dans *SolidWorks* avec les DDL visualisé en rouge

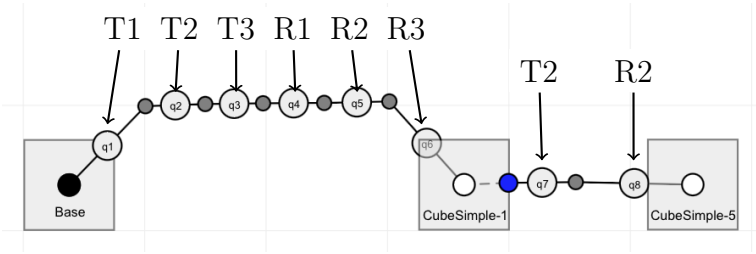


FIGURE 4.3 – MBS de l'assemblage cylindrique entre deux cubes dans *MBSysWeb*

4.2.3 Analyse

De ces différents tests nous confirmons plusieurs choses mais de manière générale nous avons pu voir que le module était capable de produire un modèle *Robotran* complet où chaque corps était présent avec les articulations associées ainsi que les propriétés de masses.

Tout d'abord, nous avons vu que le corps base a été ajouté au sommet du MBS, comme expliqué à la section 3.3.2. Celui-ci était libre par rapport au premier corps réel ; la chaîne d'articulations entre les deux représentait donc bien les 6 DDL. Pour rappel, la création de ce corps fictif "base" est un choix de conception et celui-ci permettrait de modéliser plusieurs choses dans *Robotran* selon le domaine, par exemple : dans le ferroviaire une trajectoire avec des défauts de voix, dans l'automobile un virage de rayon fixé, ...

Ensuite, les propriétés de masse, centre de masse et matrice d'inertie étaient correctement transférés dans *Robotran*. On a pu notamment observer qu'il y avait un changement d'orientation du repère origine de *SolidWorks* à *Robotran*.

Le dernier point d'attention était la validation de plusieurs contraintes et associations de contraintes.

La table 4.1 reprend les combinaisons de contraintes qui ont pu être validées suite à ce benchmark.

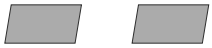
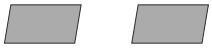
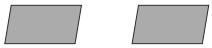
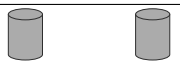
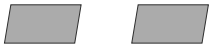
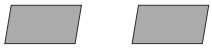
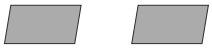
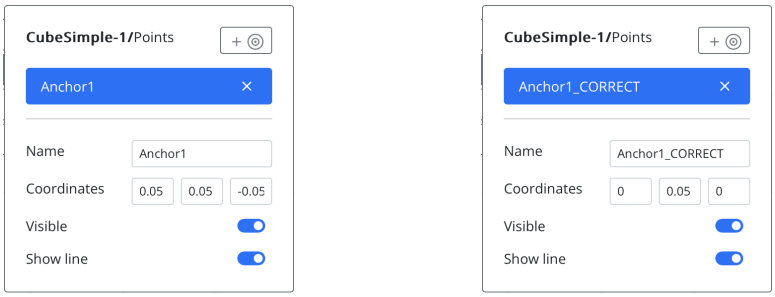
Assemblage	Contrainte 1	Paire d'entités 1	Contrainte 2	Paire d'entités 2
Cylindrique				
Co-planaire				
Planaire				
Prismatique 1				
Prismatique 2				
Sphérique				

TABLE 4.1 – Combinaisons des différentes contraintes validés par le benchmark 1

En revanche, nous avons noté le manque de précision sur les coordonnées des points d'ancrage. En effet, si nous regardons les résultats de l'assemblage "Co-planaire", les deux figures suivantes nous montrent les coordonnées que nous avons obtenues à gauche par rapport aux coordonnées à droite, correctes qu'il aurait fallu obtenir.



(a) Coordonnées obtenues par le module

(b) Coordonnées corrigées

FIGURE 4.4 – Détails des résultats obtenus sur les coordonnées des points d'ancrage

Le point d'ancrage aurait du être situé au milieu de la surface à laquelle il se rattachait mais on a vu que celui-ci était situé dans le coin inférieur gauche de cette surface ($(x, y, z) = [0.5, 0.5, -0.5]$). Cette erreur est due à l'extraction des données

de l'assemblage dans le module 1 et plus précisément à la fonction `EntityParams` de *l'API SolidWorks*. Comme expliqué dans la section 2.1.3, cette fonction nous a donné 8 paramètres chiffrés pour caractériser une entité. Parmi ces paramètres, nous avions une position (x, y, z) qui permettait de définir les coordonnées des points d'ancrage dans le module 2. La documentation de cette fonction[36] nous a appris que ces paramètres chiffrés représentaient différentes choses selon le type d'entité. Dans le cas d'un plan, cette position faisait référence à un point arbitraire dans le plan et donc ce n'était pas nécessairement le point précis où se rattachait la contrainte.
Pour solutionner ce problème, il n'était pas possible d'utiliser une fonction toute faite de *l'API SolidWorks*. En revanche, nous pouvions demander à l'utilisateur d'ajouter de l'information directement dans l'assemblage *SolidWorks* initial en créant des points de référence aux endroits souhaités. Le module 1 se chargerait de répertorier ces points avec leurs coordonnées et le module 2 pourrait ensuite les interpréter judicieusement.

4.3 Systèmes pendule/ressort et vélo

4.3.1 Objectifs

Après avoir validé plusieurs assemblages simples, nous avons appréhendé deux modèles théoriques plus complets. Ces modèles sont un système de pendule/ressort et un vélo. Ils ont été choisi pour leur capacité à tester différents aspects de l'automatisation de la génération de systèmes multicorps dans *Robotran* à partir de modèles conçus dans *SolidWorks*.

Le premier modèle, un système pendule/ressort inspiré du tutoriel de *Robotran* [3], permet d'évaluer la gestion des interactions entre plusieurs corps ainsi que la résolution des boucles cinématiques.

Le second modèle, un vélo basé sur des projets académiques [46, 47], nous permet de tester la robustesse des modules dans un contexte plus complexe, où les contraintes ne sont pas toujours alignées avec les repères des corps, nécessitant l'ajout d'articulations fixes pour une correction optimale.

A travers ces deux modèles, les objectifs spécifiques de ce benchmark étaient les suivants :

- Valider la gestion des interactions complexes entre plusieurs corps dans un système cinématique.
- Analyser la capacité du module à supprimer efficacement les boucles cinématiques en explorant différentes options de coupure.

- Tester la robustesse et la flexibilité des modules dans un modèle plus réaliste et complexe.
- Vérifier la précision dans la gestion des contraintes non alignées, en corrigeant ces dernières via des articulations fixes.

En somme, ces deux benchmarks permettent de tester la fiabilité et la flexibilité des modules, offrant ainsi une évaluation complète des capacités du système, à la fois dans des situations théoriques et dans des configurations plus réelles.

4.3.2 Résultats

Les résultats obtenus par le module pour les deux modèles définis précédemment sont présentés en annexes B et C.

La Figure 4.5 illustre le modèle *SolidWorks* utilisé comme base pour la création du système multicorps (MBS) par notre méthode, dont le résultat est présenté dans la Figure 19. Le modèle de référence du système multicorps se trouve en Annexe 18. Ce MBS est directement issu du tutoriel *Robotran* et sert de point de comparaison pour évaluer et critiquer nos résultats.

La Figure 4.8 offre un aperçu des résultats obtenus pour le modèle de vélo, basé sur le modèle *SolidWorks* illustré dans la Figure 4.7. Comme pour le modèle du pendule, nos résultats sont évalués en les comparant avec le modèle de référence tiré des travaux de Cyril Jánosi et Martin Servais, disponible en Annexe 22.

Modèle Pendule/Ressort

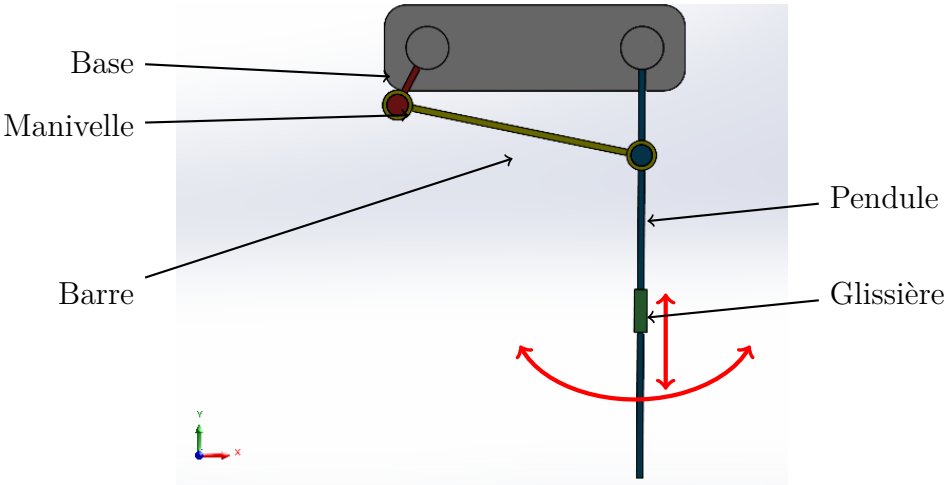


FIGURE 4.5 – Assemblage pendule/ressort dans *SolidWorks*

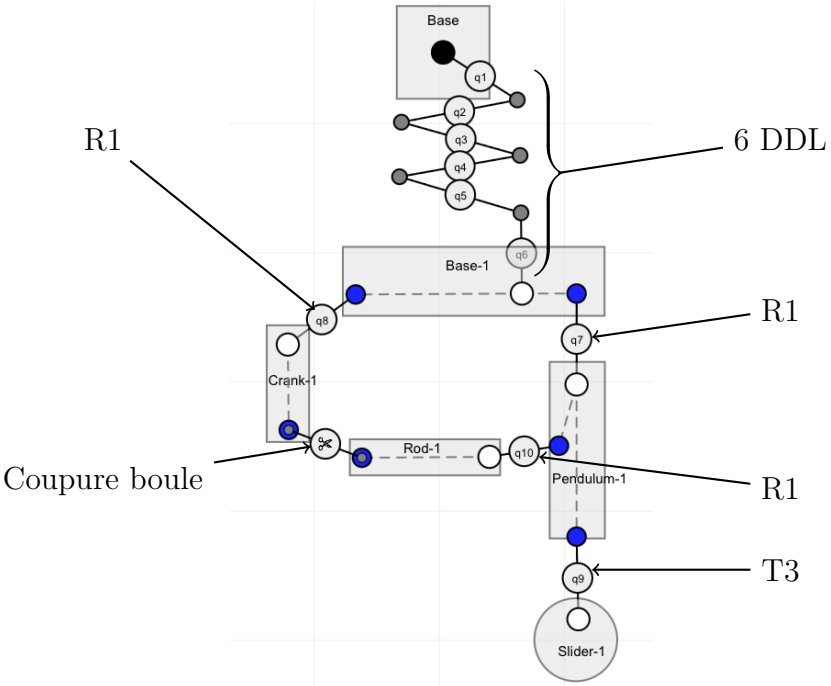


FIGURE 4.6 – Système multicorps d'un pendule/ressort dans *MBSysWeb*

Modèle du vélo

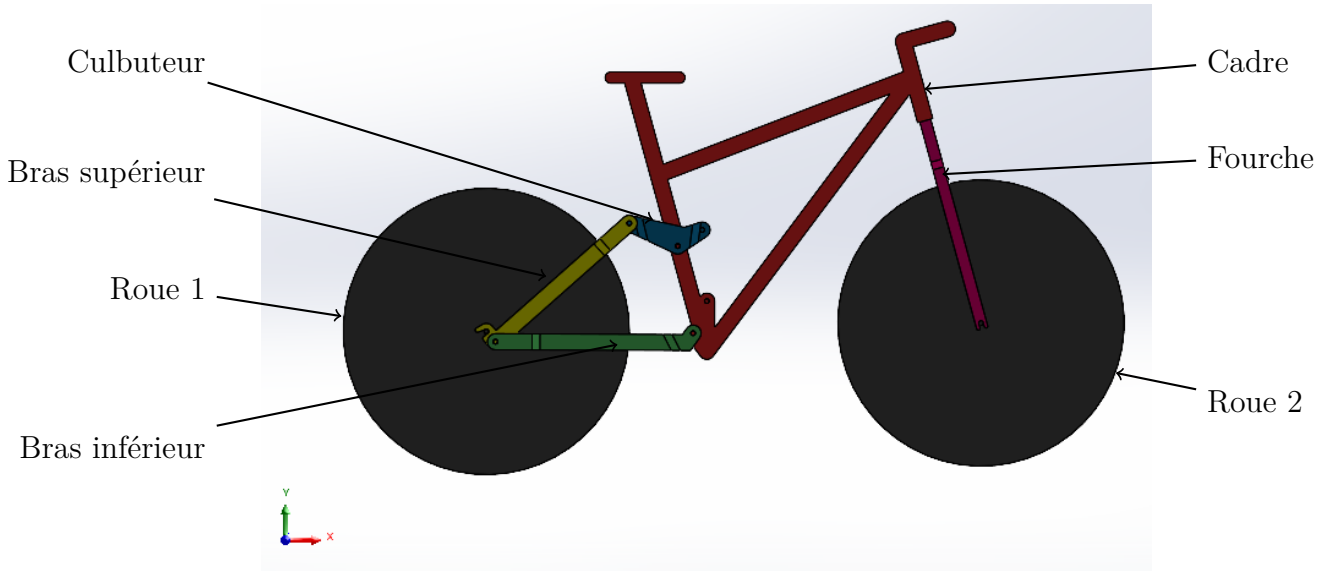


FIGURE 4.7 – Assemblage d'un vélo dans *SolidWorks*

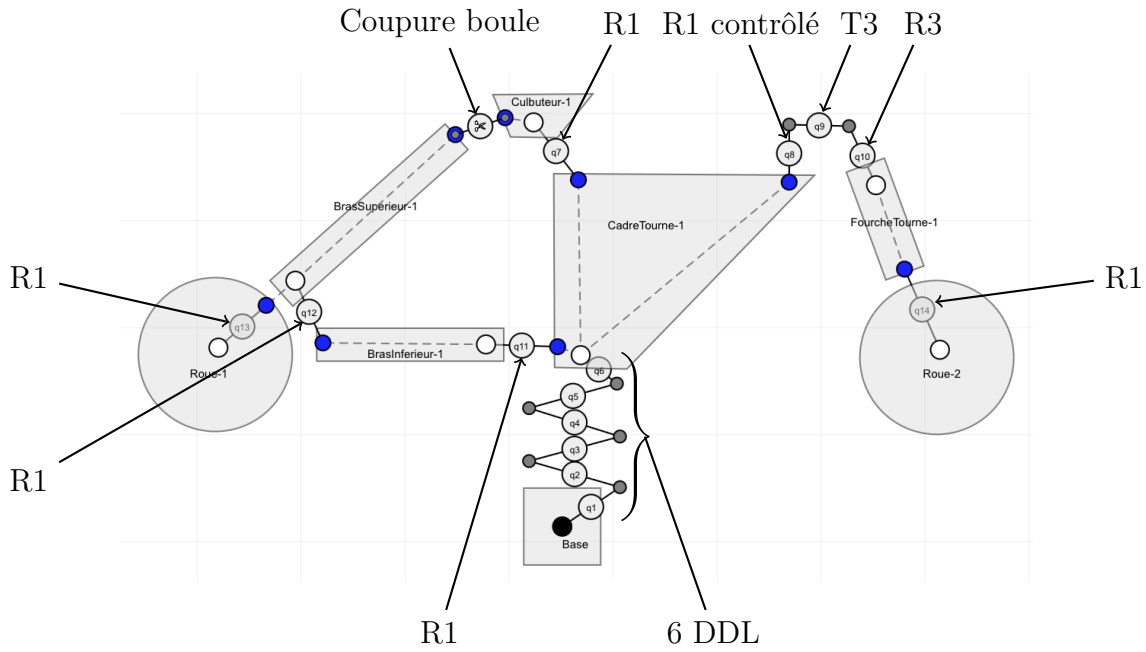


FIGURE 4.8 – Système multicorps d'un vélo dans *MBSysWeb*

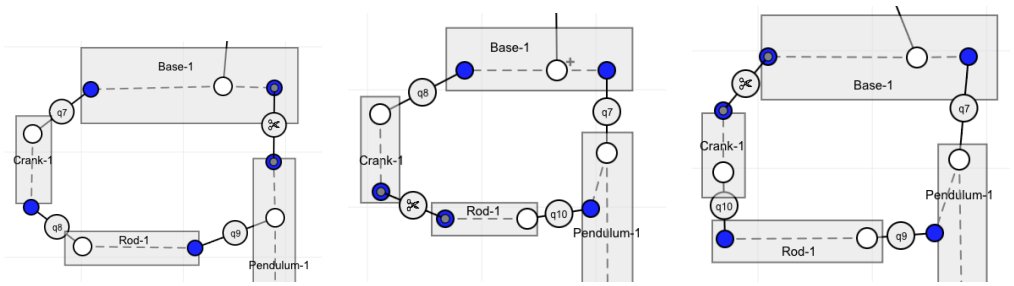
4.3.3 Analyse

Tout d'abord, nous avons pu énoncer des généralités applicables aux deux modèles.

Que ce soit pour le pendule/ressort ou pour le vélo, les interactions entre plusieurs corps on été correctement prises en compte. Cela montrait que les résultats obtenus précédemment à la section 4.2 avec deux corps peuvent être généralisés à plusieurs corps. De plus, chacun des deux modèles contenait une difficulté supplémentaire : une boucle cinématique. Nous avons observé dans les Figures 19 et 4.8 que ces boucles ont bien été supprimées à l'aide d'une coupure.

Il était d'ailleurs intéressant d'analyser plus en détail le comportement du module dans la suppression de la boucle cinématique. Pour ce point de discussion, nous avons examiné le système pendule/ressort. Si nous observons la boucle dans le sens horlogique en commençant par la base, les Figures 4.9a, 4.9b et 4.9c montrent que la coupure a été faite au début, au milieu et à la fin de cette boucle. On a vu que cela entraînait des modifications sur le modèle multicorps. Ce processus mettait en évidence l'importance de l'étape que nous avons développée dans la section 3.3.6 sur la suppression des points inutilisés et donc la raison pour laquelle nous avons initialement créé un point pour chaque contrainte. Ces points supplémentaires permettent de disposer d'une flexibilité nécessaire pour ajuster le modèle en fonction

de l'emplacement choisi pour la coupure. Une fois que les boucles cinématiques ont été supprimées et que le sens de circulation dans les chaînes cinématiques résultantes était bien établi, les points redondants pouvaient être éliminés. Cela garantissait que le modèle final soit à la fois cohérent et optimal, avec des connexions correctement orientées et structurées.



(a) Coupure au début de la boucle (b) Coupure au milieu de la boucle (c) Coupure à la fin de la boucle

FIGURE 4.9 – Comparaison des différents endroits de coupure

Les Figures 4.7 et 4.8 montrent les résultats du modèle du vélo. En plus d'une boucle cinématique, celui-ci possède une particularité dans la disposition de ses contraintes. Plus précisément au niveau la contrainte de coaxialité qui lie le cadre à la fourche avant. La Figure 4.10 nous montre l'endroit de cette contrainte, entre deux surfaces cylindriques, l'une à l'intérieur du cadre et l'autre au bout de la fourche.

En temps normal, cette contrainte aboutirait à la chaîne d'articulations suivantes "T3-R3" ($q9$ et $q10$ sur la Figure 4.8). Or dans les résultats nous avons observé un "R1 contrôlé" ($q8$ sur la Figure 4.8) qui s'ajoutait au début de cette chaîne. Cette articulation permet de représenter l'inclinaison du cadre. En d'autres mots, sans l'ajout de cette articulation, la translation et la rotation seraient purement verticales par rapport au sol. Le caractère "contrôlé" prend donc son sens dans le fait que cette rotation est fixée et permet uniquement d'aligner correctement le repère de l'articulation en translation, $q9$.

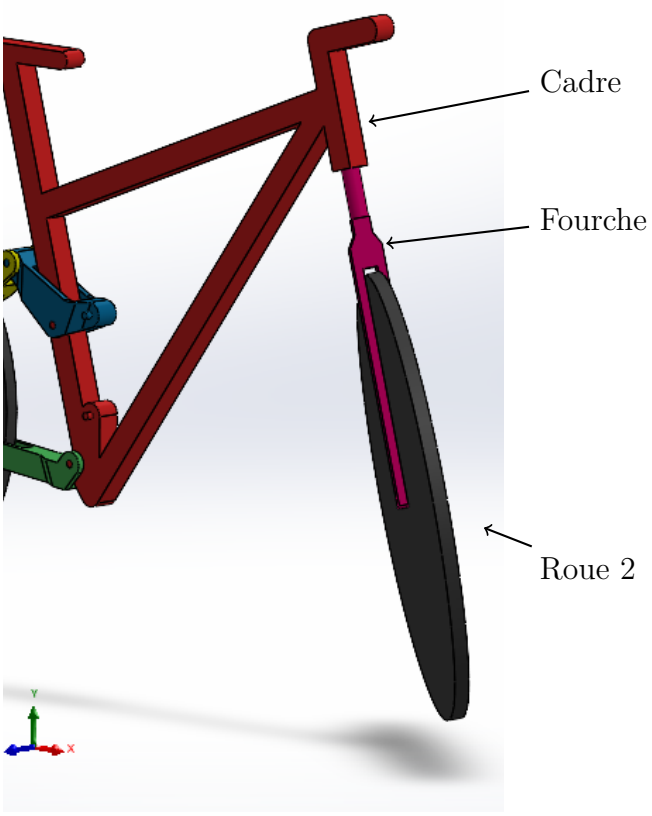


FIGURE 4.10 – Détails sur la fourche avant du vélo en position basse

Afin de pouvoir interpréter cela dans le module 2, nous avons fait appel à un peu d'algèbre linéaire. Nous avons d'abord cherché la matrice de rotation $\mathbf{R}$ entre un vecteur du repère du cadre et le vecteur principal qui caractérise notre entité (dans notre cas c'est l'axe principal du cylindre). Ensuite, nous avons séparé cette matrice de rotation générale en 3 matrices de rotation élémentaires selon les 3 angles θ_1, θ_2 et θ_3 que l'on appelle angles de *Taiht-Bryan*.

Nous avons donc considéré les deux vecteurs arbitraires en 3 dimensions, $\mathbf{v}_1$ et $\mathbf{v}_2$. L'objectif était de trouver la matrice $\mathbf{R}$ telle que :

$$\mathbf{R}\mathbf{v}_1 = \mathbf{v}_2 \tag{4.1}$$

Nous avons commencé par déterminer le produit scalaire, le produit vectoriel ainsi que la norme du produit vectoriel :

$$\mathbf{v} = \mathbf{v}_1 \times \mathbf{v}_2 \quad , \quad c = \mathbf{v}_1 \cdot \mathbf{v}_2 \quad , \quad s = \|\mathbf{v}\| \tag{4.2}$$

Le produit vectoriel $\mathbf{v}$, représente le vecteur perpendiculaire à $\mathbf{v}_1$ et $\mathbf{v}_2$, et indique l'axe autour duquel la rotation se fait. Le produit scalaire c quant à lui représente

le cosinus de l'angle θ entre les deux vecteurs ($c = \cos\theta$). Enfin la norme du vecteur $\mathbf{v}$, s nous donne le sinus de l'angle θ ($s = \sin\theta$) ;
Il nous reste à définir la *matrice antisymétrique* ou matrice *tilde* du vecteur $\mathbf{v}$:

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{4.3}$$

Une fois que nous avons obtenu tous ces éléments, la matrice de rotation résultante a pu être définie avec la *relation de Rodrigues*[43] qui permet de créer une matrice de rotation à partir d'un axe de rotation $\mathbf{v}$ et d'un angle θ :

$$\mathbf{R} = \mathbf{I} + \tilde{\mathbf{v}} + \tilde{\mathbf{v}}^2 \left(\frac{1-c}{s^2}\right) \tag{4.4}$$

ou $\mathbf{I}$ est la matrice identité.

Nous sommes passé ensuite à la deuxième étape qui consistait à décomposer cette matrice de rotation générale $\mathbf{R}$ en trois matrices de rotation élémentaires $\mathbf{R}_1$, $\mathbf{R}_2$, et $\mathbf{R}_3$, qui correspondent à des rotations autour des axes 1, 2, et 3 respectivement avec les angles θ_1, θ_2, et θ_3.
En toute généralité, la décomposition de la matrice $\mathbf{R}$ se note :

$$\mathbf{R} = \mathbf{R}_3(\theta_3)\mathbf{R}_2(\theta_2)\mathbf{R}_1(\theta_1) \tag{4.5}$$

$$= \begin{bmatrix} c\theta_2 c\theta_3 & c\theta_1 s\theta_3 + s\theta_1 s\theta_2 c\theta_3 & s\theta_1 s\theta_3 - c\theta_1 s\theta_2 c\theta_3 \\ -c\theta_2 s\theta_3 & c\theta_1 c\theta_3 - s\theta_1 s\theta_2 s\theta_3 & s\theta_1 c\theta_3 + c\theta_1 s\theta_2 s\theta_3 \\ s\theta_2 & -s\theta_1 c\theta_2 & c\theta_1 c\theta_2 \end{bmatrix} \tag{4.6}$$

ou nous utilisons $c\theta_i$ et $s\theta_i$ pour écrire $\cos\theta_i$ et $\sin\theta_i$ respectivement.

Cette matrice générale est le produit des trois rotations élémentaires. Connaissant la valeur numérique des éléments de la matrice $\mathbf{R}$, nous avons pu résoudre ce système pour obtenir les angles de *Tait-Bryan*. Pour cela nous avons utilisé la fonctions arctan2. Bien que ce développement ne soit pas détaillé ici, cette approche a permis de résoudre les angles de rotation en fonction des composantes de la matrice $\mathbf{R}$[22, 45].

$$\theta_1 = \text{ATAN2}\left(R_{32}, R_{33}\right) \tag{4.7}$$

$$\theta_2 = \text{ATAN2}\left(-R_{31}, \sqrt{R_{11}^2 + R_{21}^2}\right) \tag{4.8}$$

$$\theta_3 = \text{ATAN2}\left(R_{21}, R_{11}\right) \tag{4.9}$$

Nous avions dés à présent tout ce qu'il fallait pour définir la ou les articulations contrôlées qui permettaient d'aligner les repères.

Si nous revenons à notre exemple, vu la géométrie du cadre, il est donc normal de n'avoir qu'une seule articulation selon l'axe 1. Mais cette approche est généralisée à des géométries plus complexes où l'alignement nécessite l'ajout d'autres articulations contrôlées.

4.4 Suspension 5 points du véhicule Iltis

4.4.1 Objectifs

Ce quatrième benchmark, le plus complexe de la série, porte sur la modélisation 3D de la suspension du véhicule Bombardier Iltis de Volkswagen, un cas réel issu de l'industrie automobile publié en 1993[48]. Ce modèle sert à valider la capacité des modules à gérer des systèmes mécaniques complexes et multi-corps dans un cadre industriel.

Les objectifs spécifiques de ce benchmark sont :

- Application 3D complexe : Évaluer la capacité des modules à modéliser et simuler un système 3D industriel.
- Cas réel d'industrie : Tester l'efficacité des modules dans une application concrète de l'industrie automobile, en utilisant un modèle de suspension reconnu.
- Gestion de plusieurs boucles cinématiques : vérifier l'efficacité des modules dans la gestion de multiples boucles cinématiques.

Il est important de noter qu'un modèle Robotran de cette suspension a été réalisé par le laboratoire iMMC[49, 50]. Ce modèle sert de référence pour comparer et valider les résultats obtenus avec notre méthode.

4.4.2 Résultats

Pour ce dernier benchmark, nous avons réalisé une modélisation 3D schématique de la suspension du véhicule Bombardier Iltis de Volkswagen. La Figure 4.11 présente l'assemblage complet dans *SolidWorks*, avec une vue détaillée des composants de la suspension.
La Figure 27 illustre la complexité du MBS qui en résulte, notamment à travers les multiples boucles cinématiques. Si nous comparons avec le modèle de référence fait par le laboratoire iMMC, en annexe 26 nous pouvons voir que notre modèle est

grandement similaire. Nous discuterons par la suite de ces différences, et notamment de l'ajout d'une articulation "R2" avant chaque barre ainsi que des 6 DDL qui sont réduits à une articulations "T3" dans le modèle référent.

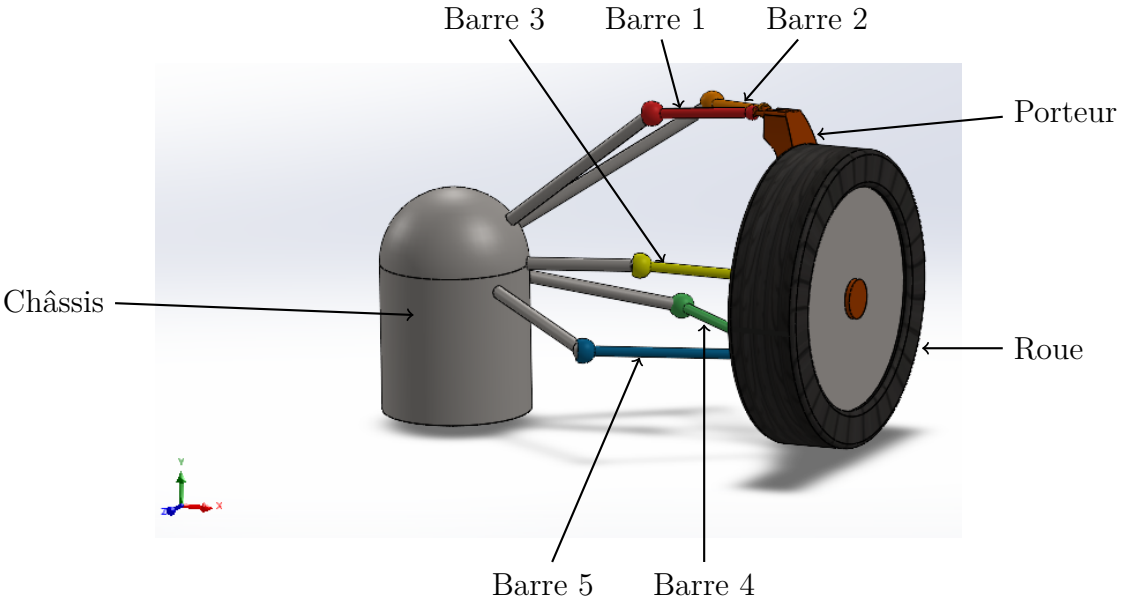


FIGURE 4.11 – Assemblage de la suspension 5 points dans *SolidWorks*

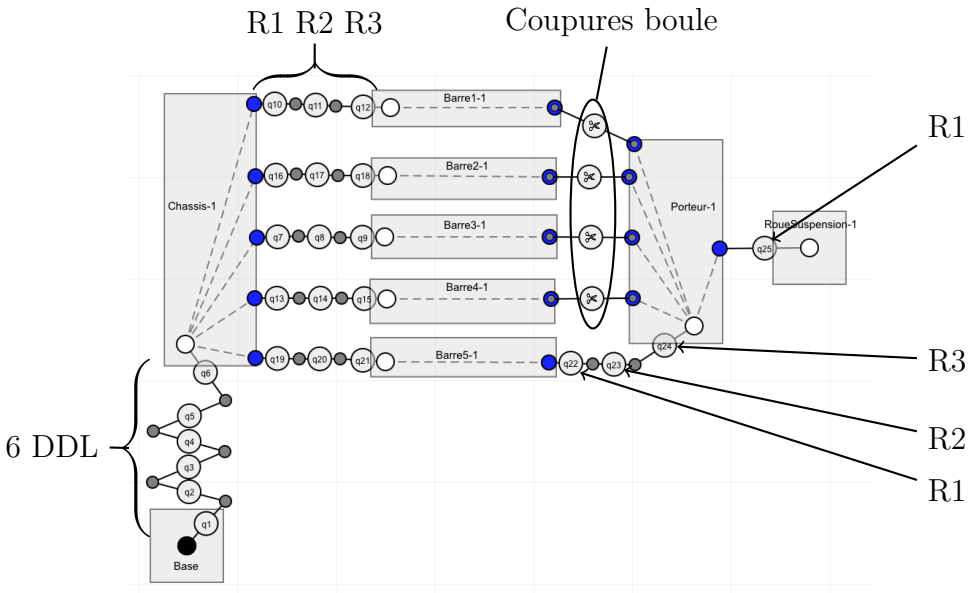


FIGURE 4.12 – Système multicorps de la suspension 5 points du véhicule Iltis dans *MBSysWeb*

4.4.3 Analyse

Nous avons examiné ces résultats dans deux analyses différentes. L'une traite des différents types de coupures qui pourraient être appliquées dans cet exemple et comment celles-ci sont gérées par les modules. Et l'autre discute des différences avec le modèle de référence 26 et du sens physique qui en découle.

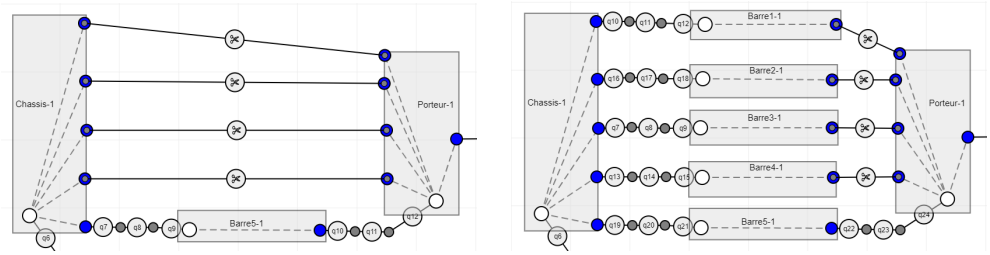
Dans un premier temps, nous avons analysé comment le module traite les boucles et les différents types des coupures qui vont permettre de supprimer celles-ci.
Pour rappel, dans *Robotran*, il existe trois types de coupures principales : solide ("solid"), boule ("ball") et bielle ("rod"). Chacune de ces coupures permet de rompre une boucle cinématique au sein du système multicorps (MBS) en ajoutant un nombre de contraintes ($h_i(q)$) qui varient en fonction du type de coupure utilisé [22].

La coupure solide ("solid") est la plus générale mais aussi la plus coûteuse en termes de calculs. Elle consiste à couper un corps en deux parties, l'original et son ombre (corps fictif). Cette coupure impose six contraintes au système, correspondant à trois contraintes de position et trois contraintes d'orientation, garantissant que les deux corps restent parfaitement alignés en termes de position et d'orientation à tout moment. Bien que cette coupure soit très précise, elle est souvent évitée dans les applications pratiques en raison de sa complexité.
La coupure boule ("ball") s'applique généralement aux boucles contenant une articulation sphérique dite "idéale" [1]. Elle est définie en connectant deux points d'ancrage et impose trois contraintes de position, assurant que les positions des deux points d'ancrage coïncident en permanence. Cette coupure est simple à implémenter, elle est donc couramment utilisée pour les articulations idéales qui ne transmettent aucun couple.
La coupure bielle ("rod") s'applique aux boucles contenant une bielle (connecting rod) dont la masse et l'inertie peuvent être négligées et dont les deux articulations peuvent être considérées comme idéales. Cette coupure impose une contrainte unique, exprimant que la distance entre les deux points d'ancrage de la bielle reste constante. C'est une méthode efficace pour traiter les boucles cinématiques dans les systèmes où la bielle peut être simplifiée de cette manière, ce qui en fait une option courante dans de nombreux systèmes mécaniques.

En pratique, l'usage de la coupure solide est relativement rare en raison de sa complexité, c'est pourquoi notre analyse s'est concentrée principalement sur les coupures boule et bielle.

1. Articulation sphérique considérée sans frottement, qui ne transmet aucun couple

Les figures ci-dessous montrent le modèle de suspension avec des coupures de type bielle (à gauche) et boule (à droite).



(a) Utilisation de coupure bielle (b) Utilisation de coupure boule

FIGURE 4.13 – Comparaison des différents type de coupures utilisées

Nous avons pu voir que l'usage de coupure bielle ou de coupure boule engendre un résultat fortement différent dans le système multicorps. Cela est dû à l'interprétation de ces deux coupures.

D'un coté, nous avons une boule qui est simplement considérée comme une articulation sphérique "idéale", elle est donc toute désignée pour remplacer une chaîne d'articulation $R1 - R2 - R3$.
De l'autre, nous avons une bielle qui fixe une distance entre deux points et qui est donc considérée comme un corps sans masse de longueur fixe avec deux articulations sphérique "idéales" aux extrémités.

Cela explique donc pourquoi une coupure bielle supprime un corps là où une coupure boule supprime une chaîne d'articulation. A noter qu'une bielle est caractérisée par une longueur. Afin de fixer celle-ci, le module propose à l'utilisateur la distance qui sépare les deux points d'ancrages liés au corps supprimé avant coupure.

Ensuite, nous avons comparé avec le modèle de référence 26 et nous avons vu que notre module a ajouté une articulation "R2" avant chaque barre de la suspension. Étant donné le modèle *SolidWorks*, cela ne représente pas une erreur en soit. En effet, les contraintes qui lient le châssis aux différentes barres sont des coïncidences entre deux points ; ce qui se traduit par les trois DDL en translation bloqués et les trois DDL en rotation libres (i.e. R1, R2 et R3).
La raison pour laquelle cette articulation n'est pas présente dans le modèle référent est due à une interprétation physique du système général. En effet, cela est dû au *moment angulaire* très faible dans la direction longitudinale de la barre, ce qui aura un impact négligeable dans *l'équation d'Euler*. Pour rappel, cela vient des *équations du mouvement de Newton-Euler* que nous allons rapidement rappeler mais qui sont

longuement détaillées dans ce livre de référence [44]. Pour un corps C, ces équations du mouvement sont caractérisées par un système de deux équations.
L'*équation de translation* ou *équation de Newton* qui découle de *la seconde loi de Newton* : la résultante de toutes les forces extérieures agissant sur un corps est égale à la dérivée temporelle de sa quantité de mouvement linéaire.

$$\dot{\mathbf{N}}(C) = \mathbf{F} \quad \Rightarrow \quad m(C)\ddot{\mathbf{x}}^G = \mathbf{F} \tag{4.10}$$

Et l'*équation de rotation* ou *équation d'Euler* qui relie *le couple résultant* appliqué au corps $\mathbf{L}^G$ à la dérivée temporelle du *moment angulaire* $\mathbf{H}^G(C)$ du corps C par rapport à son centre de masse :

$$\dot{\mathbf{H}}^G(C) = \mathbf{L}^G \tag{4.11}$$

Dans le cas d'un corps rigide, nous pouvons ré-écrire cette *équation d'Euler* en faisant apparaître la matrice d'inertie $\mathbf{I}^G$:

$$\dot{\mathbf{H}}^G(C) = \mathbf{I}^G \cdot \omega + \tilde{\omega} \cdot \mathbf{I}^G \cdot \omega = \mathbf{L}^G$$

On voit donc clairement que si un élément de cette matrice d'inertie $\mathbf{I}^G$ est beaucoup plus petit que les autres, sa contribution sur les équations du mouvement pourrait être négligée. C'est le cas pour l'ensemble des barres de suspension du modèle qui peuvent être approximées à un cylindre aligné avec l'axe y dont le rayon est bien inférieur à la hauteur ($R << h$), ce qui permet d'obtenir une matrice $\mathbf{I}^G$:

$$\mathbf{I}^G = m \begin{bmatrix} \frac{R^2}{4} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{R^2}{4} & 0 \\ 0 & 0 & \frac{R^2}{4} + \frac{h^2}{12} \end{bmatrix} \approx m \begin{bmatrix} \frac{h^2}{12} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{h^2}{12} \end{bmatrix} \tag{4.12}$$

Cette analyse souligne l'intérêt de combiner les contraintes géométriques avec des considérations dynamiques pour améliorer la cohérence des modèles générés. Cela montre que le module permet uniquement de donner une base de travail et n'est pas capable d'ajouter un sens physique sur le modèle qu'il construit. Une bonne piste d'amélioration serait d'intégrer ce genre d'analyse au module, il pourrait évoluer pour fournir des interprétations plus précises et adaptées à l'analyse voulue, ouvrant la voie à des améliorations futures.

5 Conclusion

5.1 Perspectives d'améliorations

Dans ce travail, nous avons développé une approche permettant d'intégrer une méthode d'automatisation pour la création d'un système multicorps dans *Robotran* à partir d'un assemblage *SolidWorks*. Les résultats obtenus démontrent la faisabilité de cette approche pour générer automatiquement des modèles multicorps fonctionnels dans diverses applications. Cette étude constitue une première piste de réflexion, offrant ainsi une base solide pour de futures améliorations. En effet, bien que les résultats obtenus soient prometteurs, ils ouvrent également la voie à plusieurs perspectives d'amélioration qui pourront renforcer l'utilité et la performance de la méthode.

La première proposition d'amélioration concerne l'expérience utilisateur et l'élargissement des fonctionnalités disponibles dans le module. Actuellement, la méthode se concentre sur la conception d'un système multicorps dans l'éditeur graphique 2D de *Robotran*, *MBsysPad*. Toutefois, *Robotran* est également capable de proposer une simulation 3D visuelle via *MBsysSim*. Dans cette simulation, la visualisation 3D des différents corps est souvent schématique et basée sur des formes géométriques simples (boîte, cylindre, sphère et cône). Cependant, il est aussi possible d'y intégrer des géométries plus complexes grâce à des fichiers *mesh*. Ce type de fichier, composé de points, de lignes et de surfaces, permet de modéliser fidèlement des objets en trois dimensions. Parmi les fichiers *mesh*, il existe une large variété de formats [51]. Dans *Robotran*, il est notamment possible d'utiliser les formats *.wrl* et *.obj*. Une amélioration notable serait donc d'intégrer au module 1 une fonction capable d'extraire l'ensemble des *mesh* de l'assemblage *SolidWorks* et de laisser à l'utilisateur la possibilité de les intégrer à la simulation. Cela nécessiterait une utilisation accrue de l'*API SolidWorks*, permettant d'extraire ce type de fichiers à partir des composants qui forment l'assemblage [52, 53, 54, 55]. L'intégration de cette fonctionnalité améliorerait l'analyse du système en offrant une visualisation claire pour l'utilisateur, tout en permettant de produire un rendu final de qualité pour les livrables d'une analyse. Cela offrirait aux utilisateurs de

Robotran la possibilité de fournir un rendu visuel facilement compréhensible, adapté aux besoins des industriels lors de la présentation de projets.

Une autre amélioration significative consisterait à élargir la compatibilité de notre méthode avec d'autres logiciels de CAO. Actuellement, cette méthode est conçue pour interagir avec des assemblages provenant de *SolidWorks*, qui est largement utilisé dans l'industrie. Toutefois, comme indiqué dans la section 2.1.1, il existe de nombreux autres logiciels de CAO, tels que *CATIA* et *Fusion 360*, ce dernier offrant une alternative gratuite. Ces deux logiciels disposent d'API permettant d'adopter une démarche similaire à celle utilisée avec *SolidWorks*, en particulier *Fusion 360*, qui a été conçu pour être directement compatible avec *Python* [56, 57, 58]. La structure modulaire de la méthode, organisée en deux modules distincts, permettrait d'adapter spécifiquement le module 1 pour qu'il soit compatible avec différents logiciels sources d'assemblages. Le principal défi résiderait dans la capacité à extraire et structurer les données selon un format uniformisé en sortie du module 1, indépendamment du logiciel d'origine. Ainsi, le module 2 pourrait être utilisé sans modification, garantissant la cohérence et la continuité du processus de modélisation.

Une amélioration intéressante à envisager serait d'intégrer une fonctionnalité de détection des assemblages déjà traités, permettant ainsi d'éviter la répétition complète du processus à chaque fois qu'un modèle est modifié. Cette fonctionnalité, agissant comme une sorte de "mémoire", permettrait au module de reconnaître les modifications apportées à un assemblage existant et d'adapter le modèle MBS en ne recalculant que les éléments ayant subi des changements. Outre la réduction du temps de calcul, cela permettrait d'optimiser le modèle MBS à partir de l'assemblage *SolidWorks* initial. Cette mémoire intégrée pourrait être mise en oeuvre en stockant les informations importantes des assemblages sous la forme d'une clé unique au moyen de *fonction de hachage*[59, 60]. Chacune des clés pourrait être stockée dans un fichier de configuration, qui serait consulté à chaque nouvelle simulation pour identifier les changements à apporter.

Les trois pistes propositions mentionnées ci-dessus se concentrent principalement sur l'amélioration de l'utilisation et de l'interface utilisateur de la méthode. Les propositions qui suivent se penchent davantage sur l'élargissement des capacités de la méthode et concernent principalement des améliorations du module 2.

L'une des principales améliorations techniques concerne l'élargissement de la "table de validité" 2.2, en augmentant le nombre de contraintes et d'entités compatibles avec le module 2. Actuellement, ce module se limite à trois types de contraintes (coïncidence, coaxialité et parallélisme) et à cinq entités (point, ligne, plan, cylindre et cercle), ce qui limite la complexité des assemblages pouvant être

interprétés. En élargissant la gamme de contraintes et d'entités reconnues par le module, nous pourrions mieux représenter les interactions physiques au sein des assemblages complexes. Dans un premier temps, il serait pertinent d'intégrer l'ensemble des *contraintes standard* [8] avec leurs entités associées. Par la suite, l'intégration des *contraintes avancées* [61] et des *contraintes mécaniques* [62], qui représentent des interactions plus complexes dans *SolidWorks*, pourrait également être envisagée. Cela permettrait d'interpréter directement l'assemblage par la méthode, sans nécessiter une vérification préalable des contraintes utilisées.

Apporter au module 2 une meilleure compréhension du sens physique des assemblages constitue également une perspective d'amélioration majeure. Comme nous l'avons vu dans l'analyse des résultats sur la suspension 5 points dans la section 4.4.3, il est possible que des articulations soient négligeables aux yeux de la simulation, qui peuvent être omises sans affecter de manière significative le résultat final. Par exemple, dans notre analyse, cela s'est traduit par un moment d'inertie négligeable au regard des équations du mouvement. Une fonctionnalité permettant d'identifier automatiquement ces articulations et de les traiter en conséquence pourrait simplifier les modèles, en éliminant les éléments superflus tout en conservant les interactions dynamiques du modèle. Cette approche pourrait inclure la définition de critères basés sur des seuils dynamiques ou énergétiques pour déterminer l'importance relative des différentes articulations dans un assemblage donné.

Enfin, dans un assemblage *SolidWorks*, ainsi que dans d'autres logiciels de CAO, il est courant de trouver des éléments inertes du point de vue d'une analyse dynamique. Ces éléments, tels que des vis, des supports ou d'autres composants de montage, n'apportent souvent aucune information pertinente pour la simulation dynamique. Il serait donc intéressant de développer une fonctionnalité capable de détecter ces corps dans le but de les fusionner avec les corps principaux auxquels ils sont attachés. Toutefois, bien que ces éléments soient souvent négligeables en termes de masse, il est essentiel de s'assurer que leurs contributions soient correctement intégrées dans la modélisation notamment en ajustant la position du centre de masse et la matrice d'inertie du corps fusionné. En pratique, cela pourrait s'intégrer au module via une contrainte spécifique dites de *blocage*[63] qui permet de fixer un objet dans l'assemblage. Cette fonctionnalité permettrait de réduire la complexité des modèles générés sans sacrifier la précision des simulations et contribuerait à une utilisation plus efficace des ressources de calcul dans la solution des équations du mouvement.

5.2 Conclusion générale

La principale réalisation de ce travail réside dans le développement d'une méthode pour la conception automatisée de systèmes multicorps à partir d'assemblages réalisés dans le logiciel de CAO, *SolidWorks*. Cette méthode, articulée en deux modules distincts, permet de convertir un modèle CAO en un modèle multicorps fonctionnel dans *Robotran*, facilitant ainsi le processus de création du modèle dynamique. Chaque module est structuré en plusieurs étapes, allant de l'extraction des données pertinentes via l'*API SolidWorks* à l'interprétation de celles-ci pour arriver à générer un modèle MBS prêt à être simulé.

Ce travail a permis de démontrer l'efficacité de cette méthode automatisée, ouvrant ainsi de nouvelles perspectives pour le développement de modèles de simulation plus complexes et plus réalistes. Bien que fonctionnelle, cette méthode représente une première version qui pourrait être perfectionnée et ouvre ainsi la voie à de nombreuses possibilités d'amélioration. Les améliorations proposées visent à renforcer l'efficacité, la précision et l'accessibilité du module, en tenant compte des besoins des utilisateurs ainsi que des exigences techniques des simulations multicorps.

Son développement futur pourra la rendre plus polyvalente, capable de s'adapter à une variété d'applications industrielles concrètes, en étendant notamment sa compatibilité à d'autres logiciels de CAO et en optimisant son efficacité. À terme, cette méthode pourrait être intégrée directement au système *Robotran*, offrant ainsi une solution complète et efficace pour l'analyse des systèmes multicorps. Une telle intégration permettrait de simplifier grandement le processus de modélisation pour les utilisateurs.

Le travail accompli constitue une base solide pour des développements futurs et les perspectives envisagées montrent qu'il reste encore beaucoup à explorer pour optimiser et enrichir ce processus automatisé. En combinant la puissance de *SolidWorks* pour la conception mécanique avec les capacités de *Robotran* en matière de dynamique multicorps, cette méthode promet de devenir un outil polyvalent pour la modélisation et l'optimisation des systèmes complexes dans le cadre des simulations multicorps. La combinaison des améliorations proposées pourrait permettre d'atteindre un niveau de sophistication et de performance encore plus élevé, consolidant ainsi l'utilité et la pertinence de cette méthode dans le domaine de la simulation dynamique.

Annexes

A Benchmark 1 - Géométries simples

A.1 Cylindrique

SolidWorks

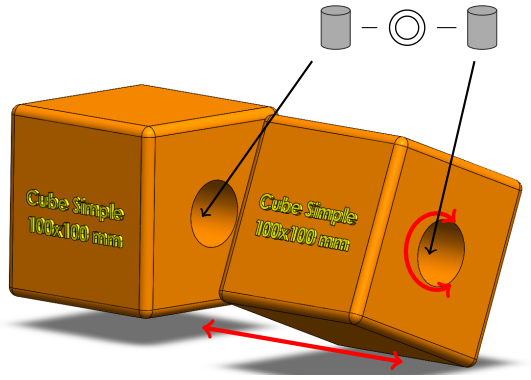


FIGURE 1 – Assemblage cylindrique entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

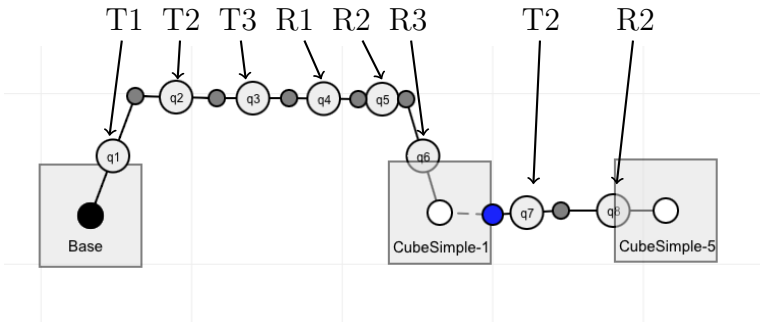


FIGURE 2 – MBS de l'assemblage cylindrique entre deux cubes dans *MBSysWeb*

A.2 Co-planaire

SolidWorks

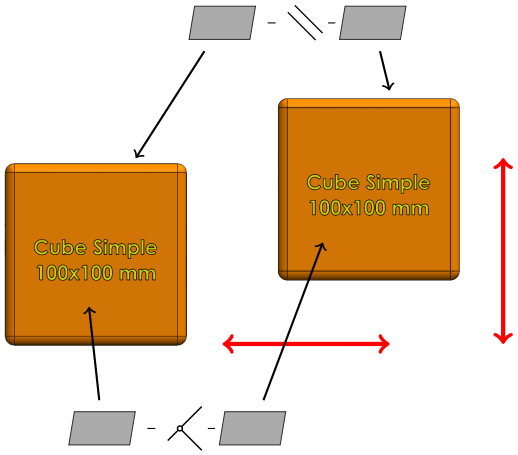


FIGURE 3 – Assemblage co-planaire entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

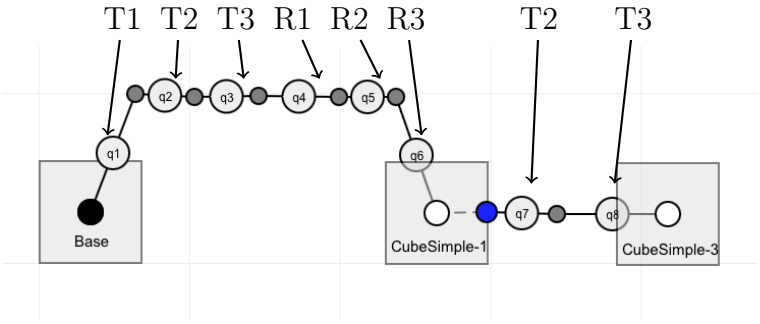


FIGURE 4 – MBS de l'assemblage co-planaire entre deux cubes dans *MBSysWeb*

A.3 Planaire

SolidWorks

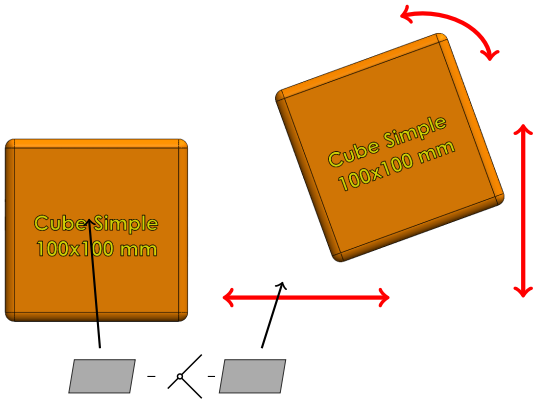


FIGURE 5 – Assemblage planaire entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

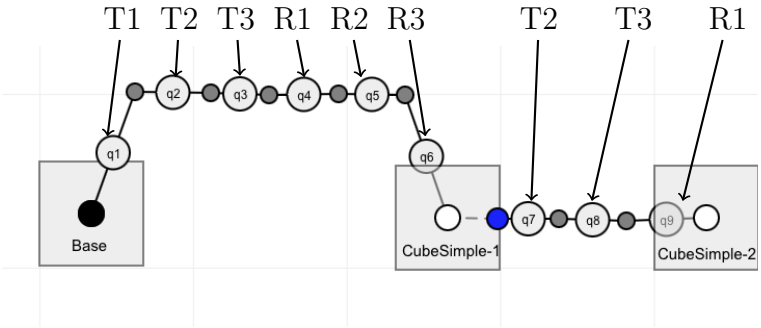


FIGURE 6 – MBS de l'assemblage planaire entre deux cubes dans *MBSysWeb*

A.4 Rotation

SolidWorks

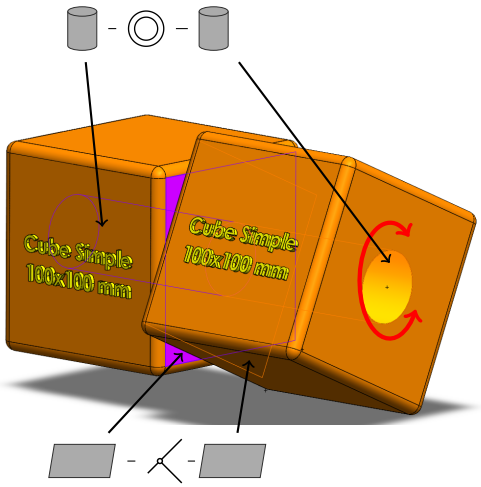


FIGURE 7 – Assemblage de rotation entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

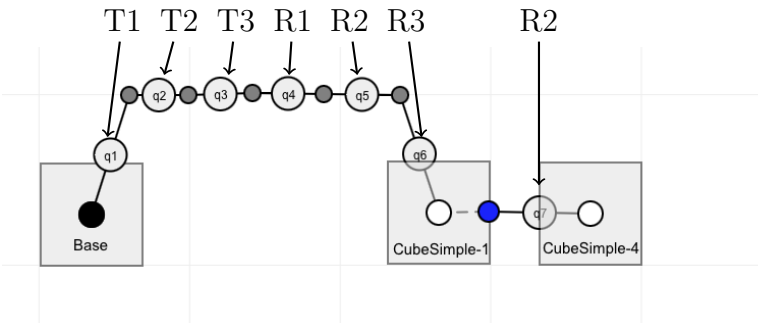


FIGURE 8 – MBS de l'assemblage de rotation entre deux cubes dans *MBSysWeb*

A.5 Prismatique 1

SolidWorks

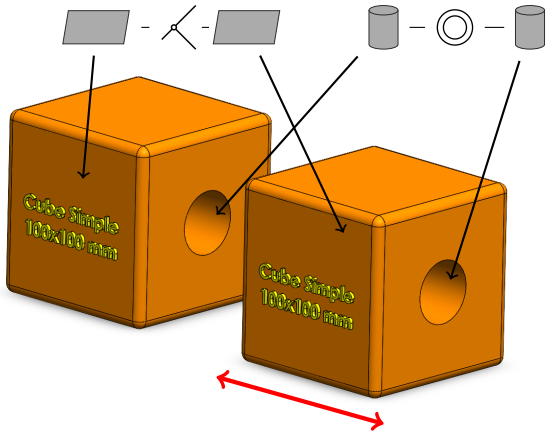


FIGURE 9 – Premier assemblage prismatique entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

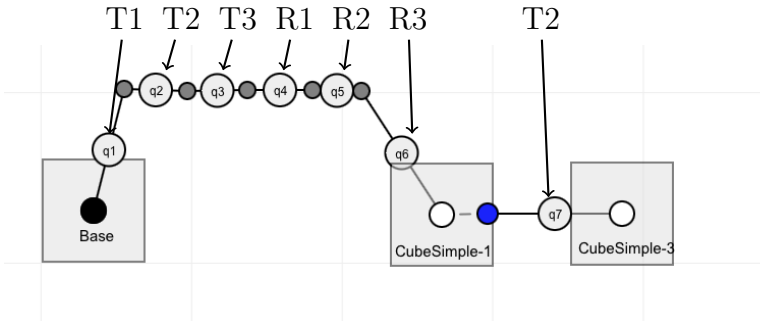


FIGURE 10 – MBS du premier l'assemblage prismatique entre deux cubes dans *MBSysWeb*

A.6 Prismatique 2

SolidWorks

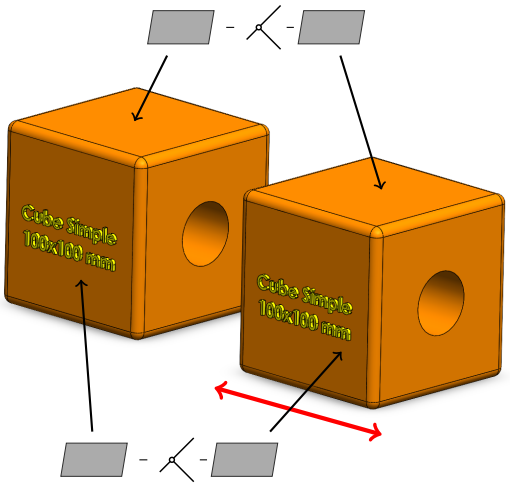


FIGURE 11 – Second assemblage prismatique entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

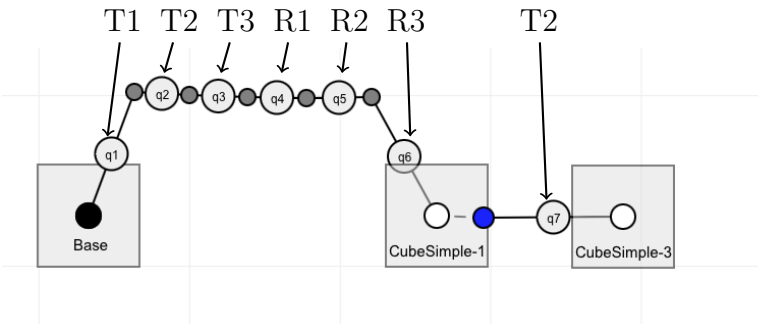


FIGURE 12 – MBS du second l'assemblage prismatique entre deux cubes dans *MBSysWeb*

A.7 Sphérique

SolidWorks

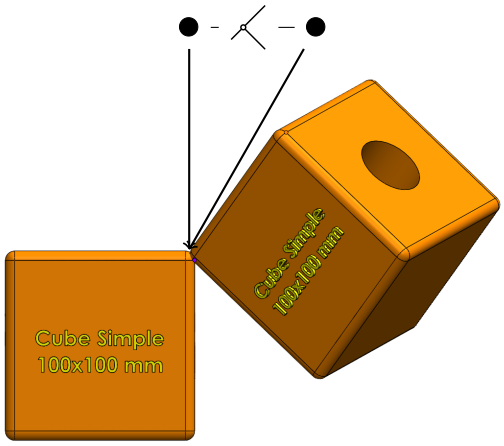


FIGURE 13 – Assemblage sphérique entre deux cubes dans *SolidWorks* avec la mise en évidence des contraintes et DDL associés (en rouge)

Résultat de la méthode

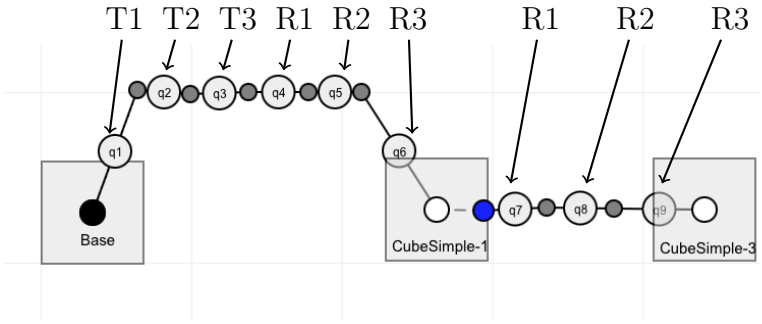
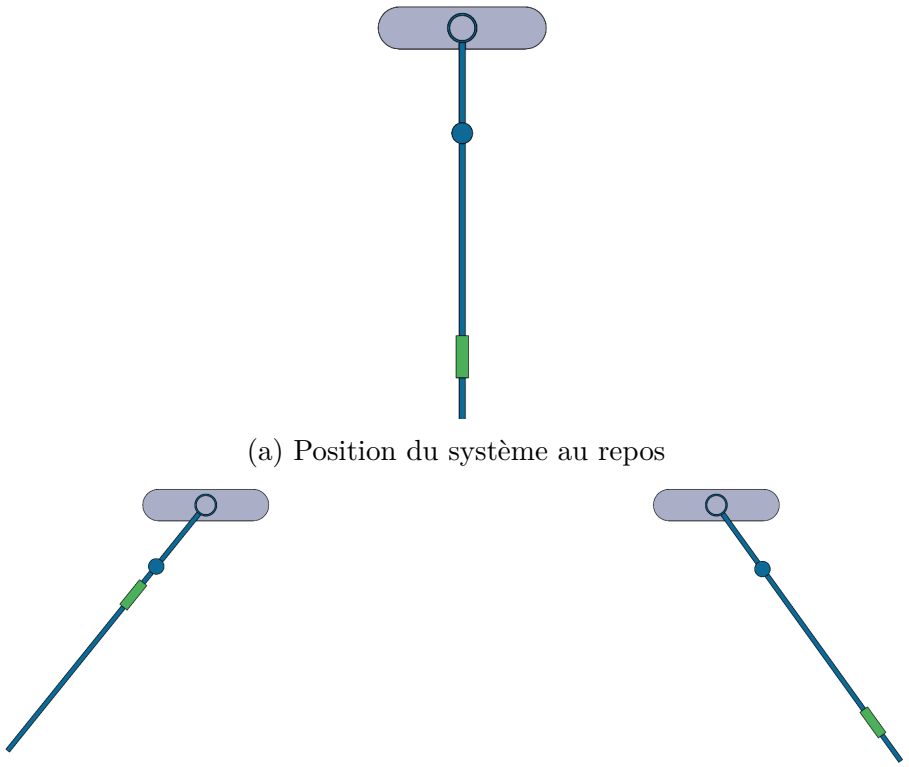


FIGURE 14 – MBS de l'assemblage sphérique entre deux cubes dans *MBSysWeb*

B Benchmark 2 - Système pendule/ressort

B.1 Pendule simplifié

SolidWorks



(a) Position du système au repos

(b) Oscillation du pendule avec la glissière en position haute

(c) Oscillation du pendule avec la glissière en position basse

FIGURE 15 – Assemblage d'un système pendule avec une glissière dans *SolidWorks*

Résultat de la méthode

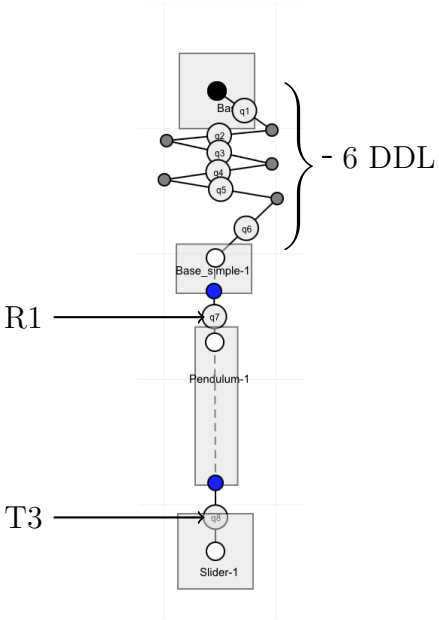
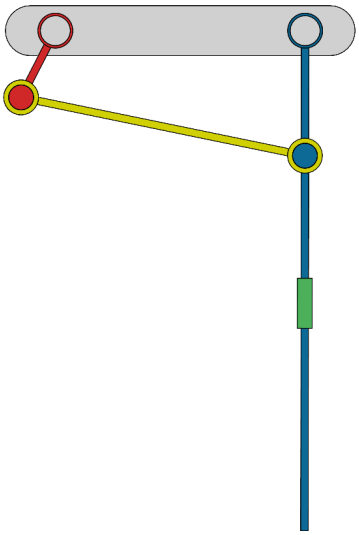


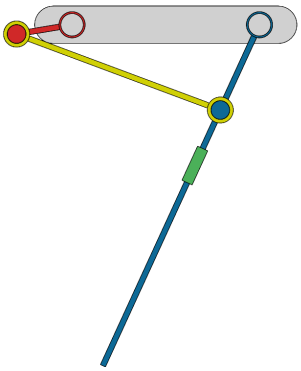
FIGURE 16 – MBS d'un système pendule avec une glissière dans *MBSysWeb*

B.2 Pendule complet

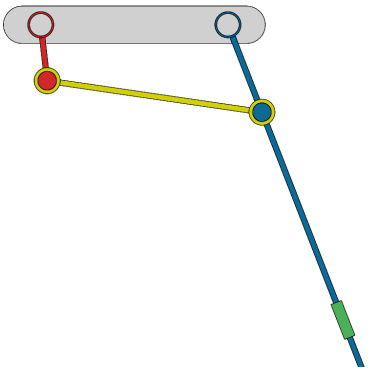
SolidWorks



(a) Position du système au repos



(b) Oscillation du pendule avec la glissière en position haute



(c) Oscillation du pendule avec la glissière en position haute

FIGURE 17 – Assemblage d'un système pendule avec une glissière et une boucle cinématique dans *SolidWorks*

Robotran

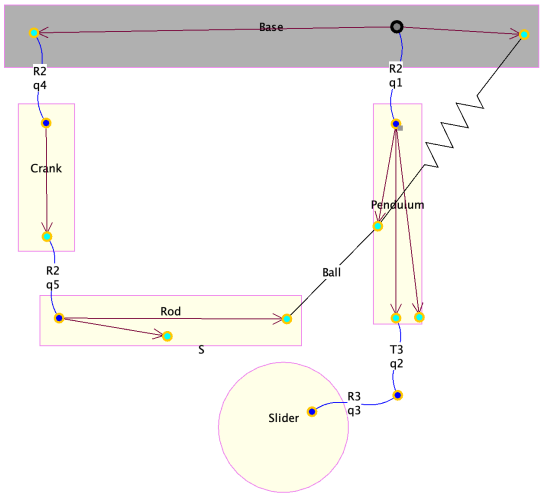


FIGURE 18 – MBS de référence d'un système pendule avec une glissière et une boucle cinématique issu du tutoriel *Robotran* dans *MBsysPad*

Résultat de la méthode

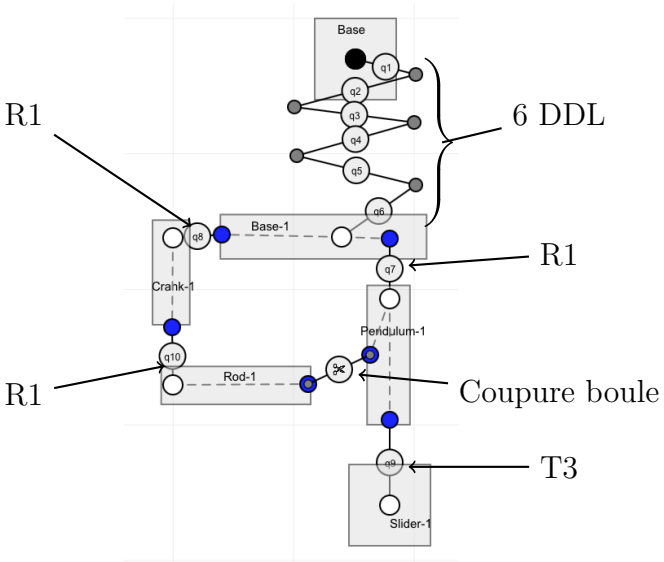
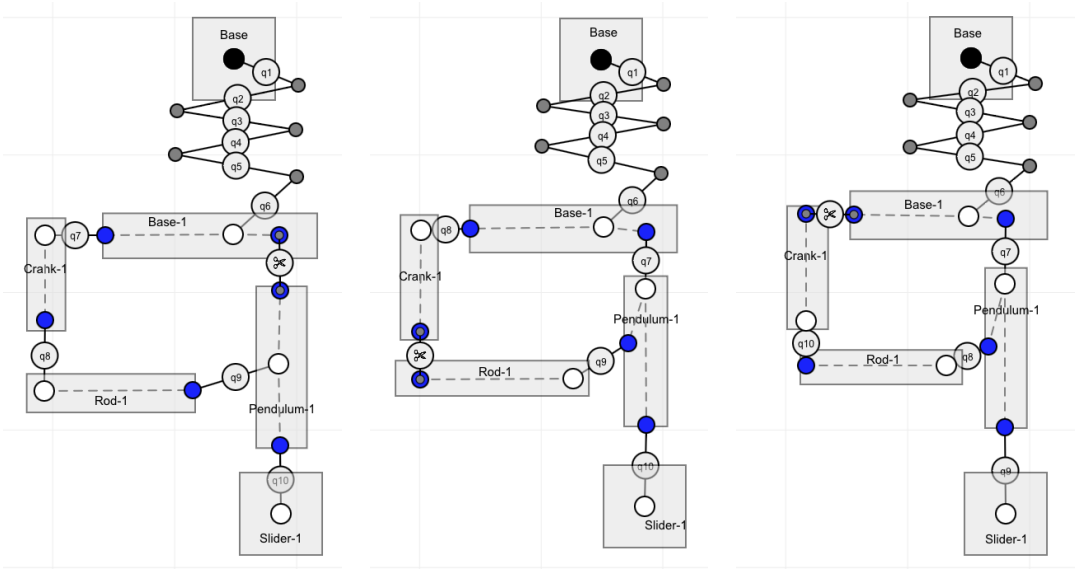


FIGURE 19 – MBS d'un système pendule avec une glissière et une boucle cinématique dans *MBSysWeb*

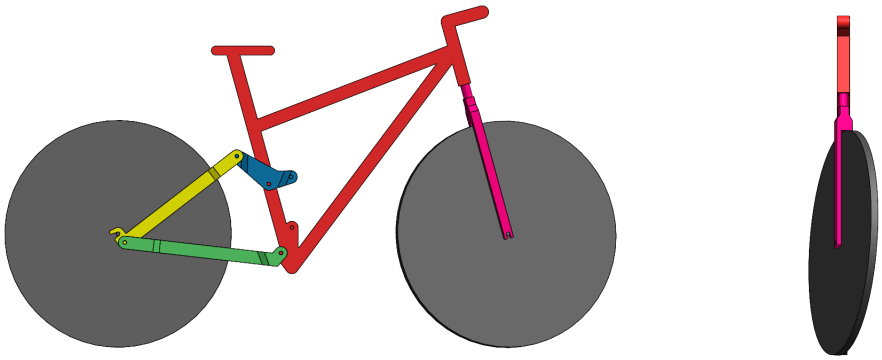


(a) Coupure au début de la boucle cinématique (b) Coupure au milieu de la boucle cinématique (c) Coupure à la fin de la boucle cinématique

FIGURE 20 – Visualisation des autres endroits de coupure de la boucle cinématique dans *SolidWorks*

C Benchmark 3 - Vélo

SolidWorks



(a) Vue de profil (b) Vue de face

FIGURE 21 – Assemblage d'un vélo composé de six corps dans *SolidWorks*

Robotran

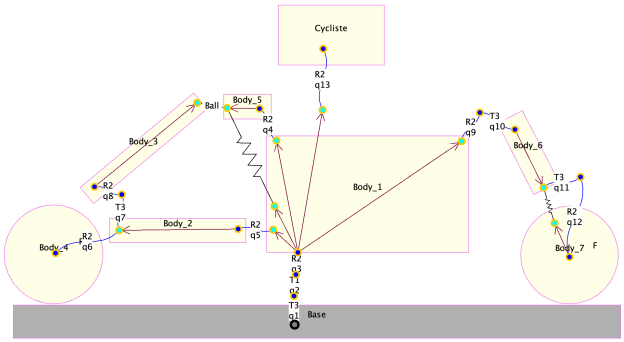


FIGURE 22 – MBS de référence du vélo validé par Cyril Jánosi et Martin Servais dans *MBsysPad*

Résultat de la méthode

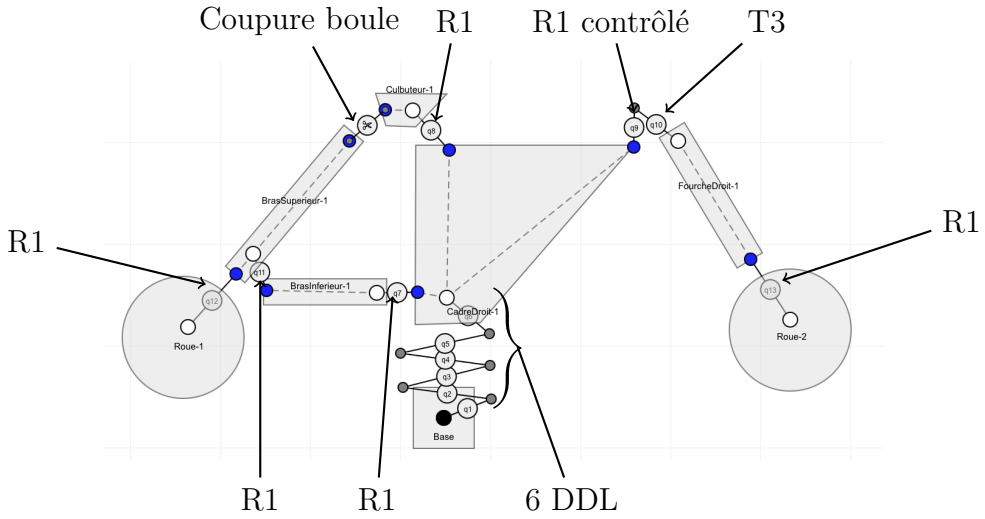


FIGURE 23 – MBS d'un vélo rectiligne dans *MBSysWeb*

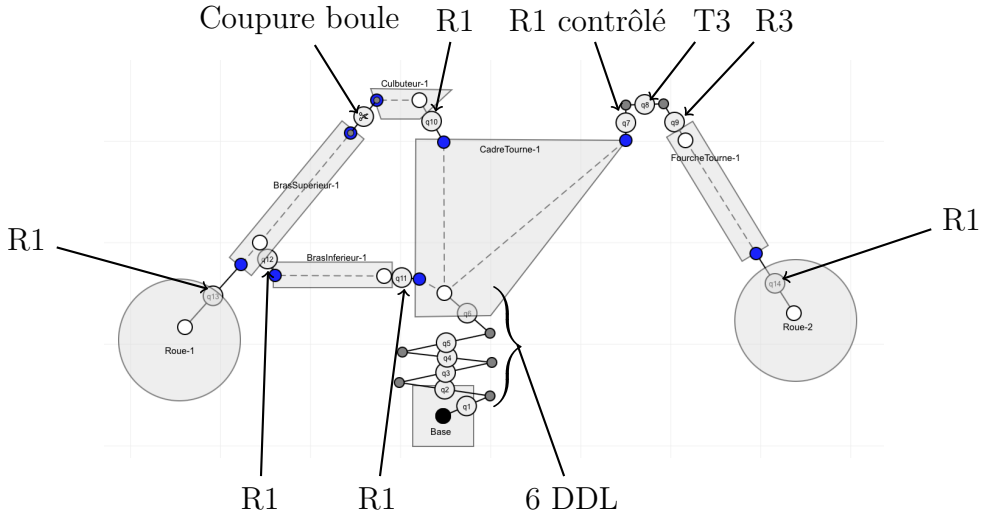
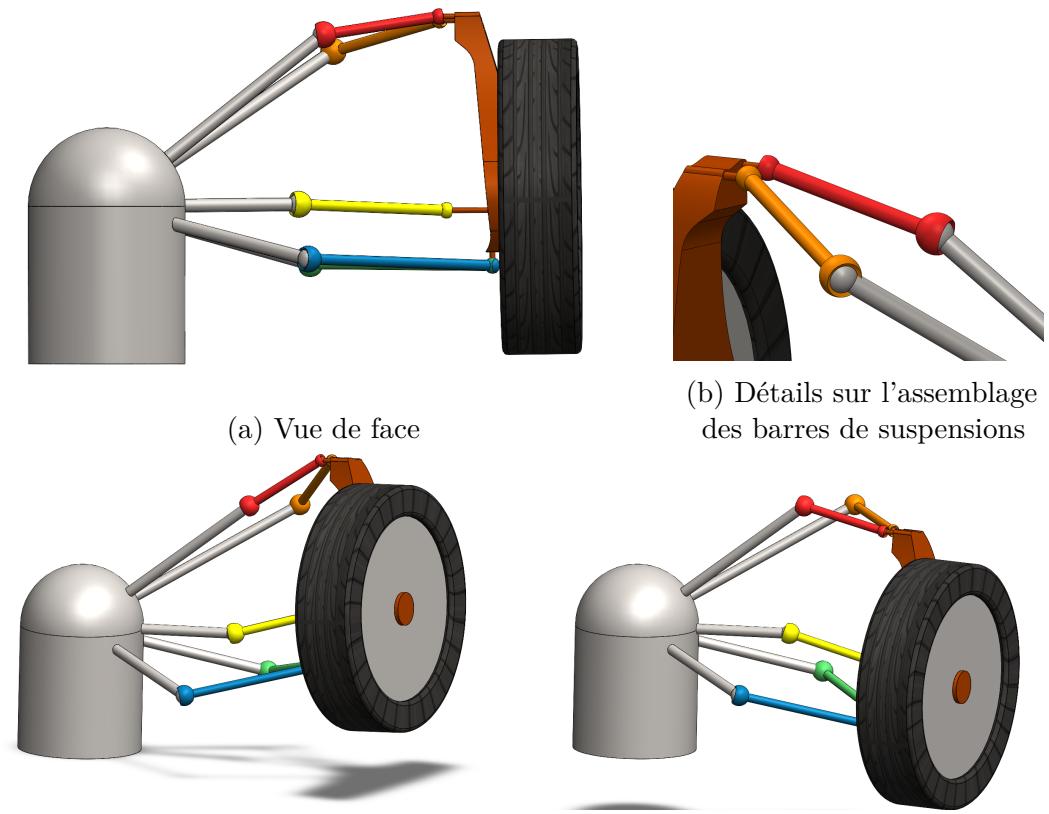


FIGURE 24 – MBS d'un vélo tournant dans *MBSysWeb*

D Benchmark 4 - Suspension 5 points du véhicule Iltis

D.1 Modèle de suspension isolée

SolidWorks



(a) Vue de face

(b) Détails sur l'assemblage des barres de suspensions

(c) Position haute

(d) Position basse

FIGURE 25 – Assemblage du système de suspension 5 points du véhicule Iltis dans *SolidWorks*

Robotran

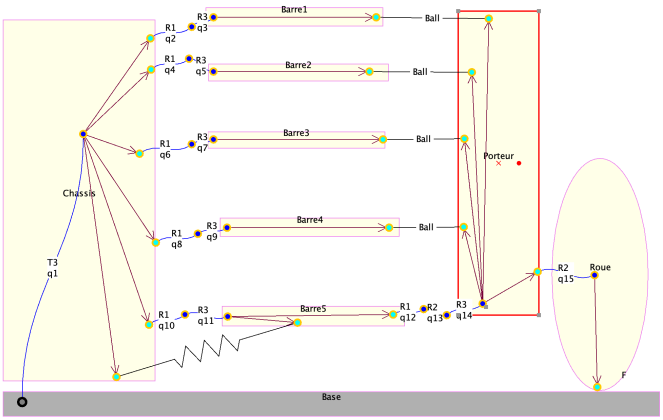


FIGURE 26 – MBS de référence du système de suspension 5 points du véhicule Iltis validé par l'iMMC dans *MBsysPad*

Résultat de la méthode

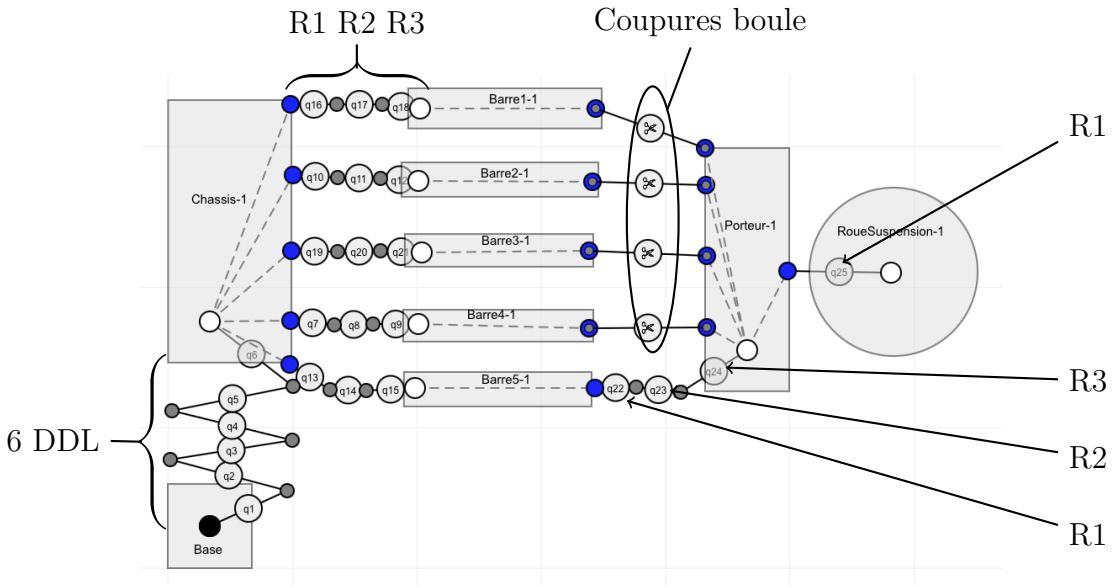
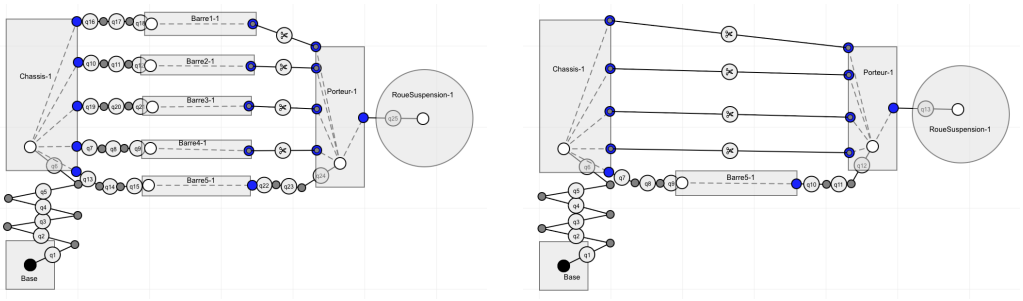


FIGURE 27 – MBS du système de suspension 5 points du véhicule Iltis dans *MBSysWeb*

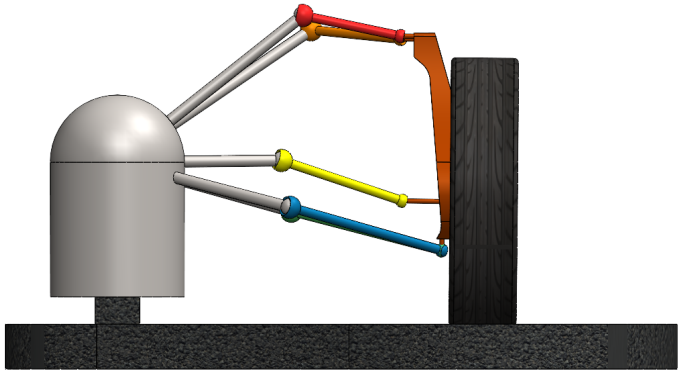


(a) Coupure boule (b) Coupure bielle

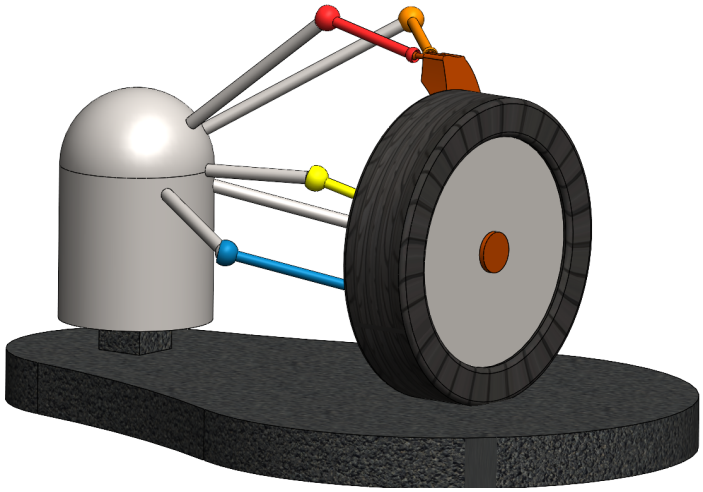
FIGURE 28 – Comparaison des des coupures de type boule et de type bielle sur le MBS du système de suspension 5 points du véhicule Iltis dans *MBSysWeb*

D.2 Modèle de suspension avec une base

SolidWorks



(a) Vue de face



(b) Vue de coté

FIGURE 29 – Assemblage du système de suspension 5 points du véhicule Iltis avec une base dans *SolidWorks*

Bibliographie

[1] Dassault Systèmes - SolidWorks Corporation. *Pourquoi choisir SOLIDWORKS ?* URL : https://www.solidworks.com/fr/solution/why-choose-solidworks. (accessed : 06.07.2024).

[2] iMMC Lab. *Robotran*. URL : https://www.robotran.be/. (accessed : 06.07.2024).

[3] iMMC Lab. *Robotran : Tutorials*. URL : https://www.robotran.be/docs/tutorials/. (accessed : 06.06.2024).

[4] ACFT Bureau d'etudes. *Logiciels de DAO, CAO, FAO, 60ans d'evolution*. 21 août 2019. URL : https://www.acft-be.fr/logiciels-de-dao-cao-fao/.

[5] LEMELS N-MIT. *Ivan Sutherland, Sketchpad*. URL : https://lemelson.mit.edu/resources/ivan-sutherland.

[6] Ivan Edward Sutherland. *Sketchpad : A man-machine graphical communication system*. Rapp. tech. University of Ccambridge, September 2023.

[7] COGiSTEM. *Une breve histoire de la CAO*. 10 septembre 2021. URL : https://www.cogistem.com/actualites/18_Une-br%C3%A8ve-histoire-de-la-CAO.

[8] Dassault Systèmes - Aide de SOLIDWORKS. *Contraintes standard*. URL : https://help.solidworks.com/2023/french/SolidWorks/sldworks/c_Standard_Mates_SWassy.htm. (accessed : 23.11.2023).

[9] Dassault Systèmes - Aide de SOLIDWORKS. *Arbre de création FeatureManager*. URL : https://help.solidworks.com/2023/French/SolidWorks/sldworks/c_featuremanager_design_tree.htm#:~:text=The%20FeatureManager%20design%20tree%20on,and%20views%20in%20a%20drawing.. (accessed : 12.01.2024).

[10] IBM. *What is an API (application programming interface) ?* URL : https://www.ibm.com/topics/api. (accessed : 20.12.2023).

[11] Dassault Systèmes - SOLIDWORKS API Help. *Welcome*. URL : https://help.solidworks.com/2025/English/api/sldworksapiprogguide/Welcome.htm.

[12] Dassault Systèmes - SOLIDWORKS API Help. *IAssemblyDoc Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IAssemblyDoc.html. (accessed : 13.11.2023).

[13] Dassault Systèmes - SOLIDWORKS API Help. *IAssemblyDoc Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IAssemblyDoc_members.html. (accessed : 13.11.2023).

[14] Dassault Systèmes - SOLIDWORKS API Help. *IComponent2 Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SOLIDWORKS.Interop.sldworks~SOLIDWORKS.Interop.sldworks.IComponent2.html. (accessed : 13.11.2023).

[15] Dassault Systèmes - SOLIDWORKS API Help. *IComponent2 Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IComponent2_members.html. (accessed : 13.11.2023).

[16] Dassault Systèmes - SOLIDWORKS API Help. *IMate2 Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMate2.html. (accessed : 20.02.2024).

[17] Dassault Systèmes - SOLIDWORKS API Help. *IMate2 Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMate2_members.html. (accessed : 20.02.2024).

[18] Dassault Systèmes - SOLIDWORKS API Help. *IMateEntity2 Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMateEntity2.html. (accessed : 07.03.2024).

[19] Dassault Systèmes - SOLIDWORKS API Help. *IMateEntity2 Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMateEntity2_members.html. (accessed : 10.03.2024).

[20] Dassault Systèmes - SOLIDWORKS API Help. *IMassProperty2 Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMassProperty2.html. (accessed : 13.01.2024).

[21] Dassault Systèmes - SOLIDWORKS API Help. *IMassProperty2 Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMassProperty2_members.html. (accessed : 10.01.2024).

[22] *Modeling Multibody Systems with ROBOTRAN*. July 2024.

[23] Nicolas Docquier, Sébastien Timmermans et Paul Fisette. "Haptic Devices Based on Real-Time Dynamic Models of Multibody Systems". In : *Sensors* 21 (juill. 2021), p. 4794. DOI : 10.3390/s21144794.

[24] The MathWorks - Help Center. *Mates and Joints*. URL : https://nl.mathworks.com/help/smlink/ref/mates-and-joints.html. (accessed : 15.05.2024).

[25] Dassault Systèmes - Aide de SOLIDWORKS. *Contrainte coïncidente*. URL : https://help.solidworks.com/2023/french/SolidWorks/sldworks/r_Coincident_Mate_SWassy.html. (accessed : 20.04.2024).

[26] Dassault Systèmes - Aide de SOLIDWORKS. *Contrainte coaxiale*. URL : https://help.solidworks.com/2023/french/SolidWorks/sldworks/r_Concentric_Mate_SWassy.html. (accessed : 20.04.2024).

[27] Dassault Systèmes - Aide de SOLIDWORKS. *Contraintes parallèles et perpendiculaires*. URL : https://help.solidworks.com/2023/french/SolidWorks/sldworks/r_Parallel_and_Perpendicular_Mates_SWassy.html. (accessed : 25.04.2024).

[28] Louis Devillez. *pySwTools*. https://github.com/ldevillez/pySwTools. 2023.

[29] *solidworks$_p$ython$_l$ynkoa*. Novembre 2019.

[30] Dassault Systèmes - SOLIDWORKS API Help. *GetComponentByID Method (IAssemblyDoc)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IAssemblyDoc~GetComponentByID.html. (accessed : 20.06.2024).

[31] Dassault Systèmes - SOLIDWORKS API Help. *Mass Property (IMassProperty2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMassProperty2~Mass.html.

[32] Dassault Systèmes - SOLIDWORKS API Help. *CenterOfMass Property (IMassProperty2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMassProperty2~CenterOfMass.html.

[33] Dassault Systèmes - SOLIDWORKS API Help. *GetMomentOfInertia Method (IMassProperty2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMassProperty2~GetMomentOfInertia.html. (accessed : 15.06.2024).

[34] Dassault Systèmes - SOLIDWORKS API Help. *IMathTransform Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SOLIDWORKS.Interop.sldworks~SOLIDWORKS.Interop.sldworks.IMathTransform.html.

[35] Dassault Systèmes - SOLIDWORKS API Help. *Type Property (IMate2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMate2~Type.html.

[36] Dassault Systèmes - SOLIDWORKS API Help. *EntityParams Property (IMateEntity2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SOLIDWORKS.Interop.sldworks~SOLIDWORKS.Interop.sldworks.IMateEntity2~EntityParams.html. (accessed : 23.06.2024).

[37] Robert SEDGEWICK Kevin WAYNE. *Algorithms FOURTH EDITION*. Addison-Wesley, March 2011.

[38] Coiin Academy. *Qu'est-ce qu'un DAG ou graphe orienté acyclique ? (Directed Acyclique Graph)*. URL : https://coinacademy.fr/academie/dag-directed-acyclic-graph-graphe-acyclique-oriente/. (accessed : 08.07.2024).

[39] Dassault Systèmes - SOLIDWORKS API Help. *swMateType_e Enumeration*. URL : https://help.solidworks.com/2023/english/api/swconst/solidworks.interop.swconst~solidworks.interop.swconst.swmatetype_e.html. (accessed : 30.07.2024).

[40] Dassault Systèmes - SOLIDWORKS API Help. *swMateEntity2ReferenceType_e Enumeration*. URL : https://help.solidworks.com/2023/english/api/swconst/solidworks.interop.swconst~solidworks.interop.swconst.swmateentity2referencetype_e.html. (accessed : 30.07.2024).

[41] Alyssa walker. "Types de graphiques dans la structure de données avec exemple". In : *GURU99* (2024). URL : https://www.robotran.be/works/bombardier-iltis-vehicle/.

[42] Sabry FOUAD. *Depth First Search*. One Billion Knowledgeable, 2023, 2022.

[43] Carlo TOMASI. *Vector Representation of Rotations*. Rapp. tech. Duke Trinity College Of Arts et Sciences, 2013. (accessed : 10.06.2024).

[44] Paul FISETTE Jean-Claude SAMIN. *Symbolic Modeling of Multibody Systems*. Springer Science = Business Media, 2003.

[45] Gregory G. SLABAUGH. *Computing Euler angles from a rotation matrix*. Rapp. tech. University of London, Decembre 2022. (accessed : 17.05.2024).

[46] *LEPL 1504 - Livrable 2 Analyse dynamique du VTT complet et morphologie du vélo chargé*. Rapp. tech. Ecole Polytechnique de Louvain, 2024. (accessed : 14.07.2024).

[47] Cyril JÁNOSI Martin SERVAIS. "Analyse de la dynamique et de la consommation d'énergetique d'un vélo". Mém. de mast. École Polytechnique de Louvain, 2024.

[48] G. Leister W. Schwartz S. FRIK. "Simulation of the IAVSD Road Vehicle Benchmark Bombardier Iltis with FASIM, MEDYNA, NEWEUL and SIMPACK". In : *Vehicle System Dynamics* 22.sup1 (1993), p. 215-253. DOI : https://doi.org/10.1080/00423119308969496.

[49] ROBOTRAN. *Bombardier Iltis vehicle*. URL : https://www.robotran.be/works/bombardier-iltis-vehicle/. (accessed : 24.08.2024).

[50] *IAVSD BENCHMARK For Multibody Simulation Software*. Rapp. tech. Ecole Polytechnique de Louvain, Novembre 1991. (accessed : 10.06.2024).

[51] CADINTEROP. *What are the most commun 3d mesh files format and how to solve interoperability issues*. URL : https://www.cadinterop.com/en/component/content/article/825-what-are-the-most-common-3d-mesh-file-formats-and-how-to-solve-interoperability-issues.html?catid=29&Itemid=393#:~:text=STL%20(.,including%203DS%20Max%20and%20Blender.. (accessed : 22.07.2024).

[52] Dassault Systèmes - SOLIDWORKS API HELP. *GetBodies3 Method (IComponent2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IComponent2~GetBodies3.html.

[53] Dassault Systèmes - SOLIDWORKS API HELP. *IBody2 Interface Members*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IBody2_members.html.

[54] Dassault Systèmes - SOLIDWORKS API Help. *GetMeshBody Method (IBody2)*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IBody2~GetMeshBody.html.

[55] Dassault Systèmes - SOLIDWORKS API Help. *IMeshBody Interface*. URL : https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IMeshBody.html.

[56] Autodesk DevTV. *Fusion 360 Online Hackathon Session 2 : Introduction to the API*. Youtube. 2016. URL : https://www.youtube.com/watch?v=6vUEJ5Iy2AQ.

[57] Autodesk. *Welcome to the Fusion API (Application Programming Interface)*. Autodesk. URL : https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-A92A4B10-3781-4925-94C6-47DA85A4F65A.

[58] Dassault Systemes. *Developer guides*. Dassault Systemes. URL : https://www.3ds.com/support/documentation/developer-guides.

[59] Wikipédia. *Fonction de hachage*. URL : https://fr.wikipedia.org/wiki/Fonction_de_hachage.

[60] alvinashcraft. *Exemple de programme C : création et hachage d'une clé de session - Win32 apps*. 13 juin 2023. URL : https://learn.microsoft.com/fr-fr/windows/win32/seccrypto/example-c-program-creating-and-hashing-a-session-key (visité le 17/08/2024).

[61] Dassault Systèmes - Aide de SOLIDWORKS. *Contraintes avancées*. URL : https://help.solidworks.com/2025/french/SolidWorks/sldworks/c_Advanced_Mates.htm. (accessed : 02.06.2024).

[62] Dassault Systèmes - Aide de SOLIDWORKS. *Contraintes mécaniques*. URL : https://help.solidworks.com/2025/french/SolidWorks/sldworks/c_Mechanical_Mates.htm. (accessed : 06.06.2024).

[63] Dassault Systèmes - Aide de SOLIDWORKS. *Contrainte de blocage*. URL : https://help.solidworks.com/2025/french/SolidWorks/sldworks/r_Lock_Mate_SWassy.htm. (accessed : 20.07.2024).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**