Simulation Webots

Définition des référentiels (f. 1 et 2)

Les angles et les vitesses angulaires sont positifs, lorsque le vecteur de rotation a la même direction que l'axe correspondant : Φ-x, Θ-y et Ψ-z (cf. f.1).
Dans le code, les indices utilisés correspondent au référentiel du robot, version Samir. La conversion au référentiel robot, version Webots, se fait lors du passage des arguments dans la fonction custom_robot_set_rel_force_and_torque() et la conversion au référentiel robot, version Samir, se fait lors de l'interprétation des éléments de la matrice fournie par le GPS (cf. f.2).

Forces généralisées projetées sur le robot

Modifications apportées au modèle de Samir :

- pour que l'angle et la vitesse de rotation selon Φ soient positifs, il faut remplacer U2=F4–F2 par F2–F4.
- pour calculer les moments gyroscopiques, il faut définir la variable Ω comme Ω2+Ω4-Ω1-Ω3 pour calculer TΘ positif et l'opposé pour calculer TΦ positif (cf. f3 pour les schémas). L'influence de ces forces est donc quasiment nul : le terme Ω est très faible, l'inertie rotorique Ipr est faible et les vitesses de rotation de l'hélicoptère, s'il est bien contrôlé, sont faibles. Cet influence n'est d'ailleurs pas visibles dans la simulation.
- ajout de forces de frottement d'amplitude k*v*v et de direction opposée à la vitesse. Celles-ci sont indispensables selon x et y, sinon pendant tout le temps où l'hélicoptère est déséquilibré (pas à l'horizontale), des forces horizontales sont engendrées et l'hélicoptère est accéléré selon x et y. S'il n'est pas freiné, il est accéléré et acquiert une vitesse toujours plus grande dans une direction.
- perturbations afin de déstabiliser l'hélico.

Résumé des forces appliquées (sauf les perturbations) sur f.4.

Code

description_helico.h
Ce fichier contient les déclarations et initialisations de toutes les variables globales décrivant l'hélicoptère : dimensions mécaniques, description des forces et paramétrisation du régulateur. C'est donc ce fichier qu'il faut modifier pour modéliser un hélicoptère de dimensions différentes ou modifier le régulateur (cf. f.5 pour le nom donné à chaque dimension).
variables.h
Ce fichier contient la déclaration de toutes les autres variables, non initialisables, utiles à la simulation et à une meilleure lecture du code.
myfunctions.h
Ce fichier contient la déclaration des fonctions créées pour la simulation. Afin de me simplifier le travail, toutes les variables sont globales et les fonctions ne nécessitent pas le passage de paramètres en argument (sauf la fonction régulateur qui doit distinguer quelques variables).

helico.c

Programme principal :

- reset du robot.
- calcul de la masse, du CM et des moments d'inertie du robot, affichage des résultats.
- facultatif : modification de certains paramètres par l'utilisateur.
- activation du GPS.
- initialisation de la position du robot : x[0], y[0], z[0], Φ[0], Θ[0] et Ψ[0].
- boucle de contrôle infinie :
    - lecture de la position et de l'orientation par le GPS : x[1], y[1], z[1], Φ[1], Θ[1] et Ψ[1].
    - approximation linéaire des vitesses correspondantes : dx, dy, dz, dΦ, dΘ, dΨ.
    - calculs du régulateur : sortie = forces de commande généralisées théoriques U1, U2, U3 et U4.
    - calcul des vitesses de rotation des hélices correspondantes : vpu1, vpu2, vpu3 et vpu4 (passage Ui → vpui : cf. f.6 et fichier *Relation U-vp.xls*).
    - limitation de ces vitesses par les capacités maximum des moteurs, c'est-à-dire leurs accélération (amax) et vitesse maximum (vmax) : vp1, vp2, vp3 et vp4.
    - calcul des forces généralisées de commande correspondants aux vitesses réellement applicables au robot (c'est-à-dire en tenant compte des limitations des moteurs) : U1,U2,U3 et U4.
    - calcul des moments gyroscopiques tgΦ et tgΘ : calculés à partir des dernières vitesses de rotation de l'hélico mesurées (dΦ, dΘ) et des vitesses de rotation des hélices qui vont être appliquées (vp1, vp2, vp3 et vp4).
    - calcul des forces de frottement ffrotx, ffroty et ffrotz : calculées à partir des dernières vitesses linéaires mesurées (dx, dy et dz).
    - calcul des forces généralisées à appliquer au robot : sorties du régulateurs réellement applicables U1,U2,U3 et U4 + moments gyroscopiques tgΦ et tgΘ + forces de frottements ffrotx, ffroty et ffrotz => forces généralisé fx, fy, fz, tx, ty et tz.
    - ajout de perturbations afin de déstabiliser l'hélicoptère et tester l'efficacité des régulateurs : perturbation de type impulsion, perturbation constante ou perturbation harmonique.
    - application des forces généralisées à l'hélicoptère par la fonction custom_robot_set_rel_force_and_torque().
    - mise à jour des variables de position et d'orientation pour le calcul des vitesses à la prochaine itération : x[1], y[1], z[1], Φ[1], Θ[1] et Ψ[1] → x[0], y[0], z[0], Φ[0], Θ[0] et Ψ[0].
    - simulation.

Remarque : la fonction *init()*, facultative, permet de rentrer certaines données par l'intermédiaire de la version exécutable, plutôt que de chaque fois recréer à chaque changement le fichier exécutable. Le plus rapide et pratique : à toi de voir. Pour faire une démo, ça peut être utile de changer les paramètres par l'exécutable.

Cas où les barres ne sont pas horizontales (α!=0)

Centre de masse :

Si α est différent de 0, il faut calculer différemment la position du centre de masse : ce code est inclut dans la fonction init() du fichier *myfunctions.h*, les variables supplémentaires utilisées dans ce calcul sont illustrées sur la f.7 et les calculs sont détaillés sur la f.8.

Moments d'inertie

Comme le calcul des moments d'inertie (α=0) tient compte de la position du centre de masse, les moments d'inertie sont modifiés dans ce sens. Ils ne sont toutefois pas exacts, puisque les changements d'orientation des éléments par rapport aux axes du robot ne sont pas pris en compte. Pour les éléments qui ne sont pas affectés par l'inclinaison des barres, les calculs sont exacts : électronique, support électronique et batterie. Quant aux autres éléments, voici la liste des formules qui ne sont pas exactes : Ixxmm1, Ixxbb1, Ixxpp1, Iyymm2, Iyybb2, Iyypp2, Izzmm1, Izzmm2, Izzbb1, Izzbb2, Izzpp1, Izzpp2.

Forces généralisées

Concernant les modifications sur les forces généralisées, la f.9 illustre les modifications et la f.10 donne la nouvelle expression des forces. Quelques doutes concernant la projection et l'expression des moments gyroscopiques !!!

Modifier la structure 3D de l'hélico dans Webots ?

Inclinaison des barres, moteurs et hélice de l'hélico

Webots : modifier l'angle alpha dans : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → rotation → x = -1, modifier alpha (radians).

Controller : modifier l'initialisation de l'angle alpha (degrés) dans *description_helico.h*.

Position des moteurs

Webots : modifier la translation Z du moteur : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → children → 3ème Solid (définition du moteur) → translation Z.

Controller : modifier l'initialisation de la variable distance_moteur dm (description_helico.h).

Position des hélices

Webots : modifier la translation Z de l'hélice : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → children → 2ème Solid (définition d'une hélice) → translation Z.

Controller : modifier l'initialisation de la variable entraxe ea dans *description_helico.h*.

Longueur d'une barre

Webots :

1) modifier la hauteur du cylindre définissant la demi-barre : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → children → 1er Solid (définition d'une barre) → children → Shape → geometry Cylinder → height.

2) modifier sa translation par rapport à l'origine du référentiel du robot (translation = ½ hauteur du cylindre) : CustomRobot → children → 1er Solid (définition d'un quart d'hélico) → children → DEF QUART_HELICO → children → 1er Solid (définition d'une barre) → translation Z.

Controller : modifier l'initialisation de la longueur totale d'une barre Lb dans *description_helico.h* : deux fois la longueur du cylindre de Webots.

Inclinaison des moteurs par rapport aux barres

Webots :

1) modifier l'angle alpha dans : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → children → 3ème Solid (définition d'un moteur) → rotation → x = -1, modifier alpha.
2) modifier également l'inclinaison des hélices, il modifier l'angle alpha dans : CustomRobot → children → 1er Solid (définition du 1er quart d'hélico) → children → DEF QUART_HELICO → children → 2ème Solid (définition d'une hélice) → rotation → x = -1, modifier alpha.

Controller : pas implémenté.

Masse totale du robot

Webots : modifier mass dans : CustomRobot → physics → mass (pour que la masse soit prise en compte, il faut laisser density à –1).

Controller : modifier l'initialisation de la masse de chaque élément dans *description_helico.h*, la masse totale est calculée automatiquement.

Position du centre de gravité du robot

Webots : modifier les trois premiers champs de la matrice d'inertie : CustomRobot → physics → inertiaMatrix : position du centre de masse par rapport à l'intersection des barres : 1er champ = position en y, 2ème champ = position du z et 3ème champ = position en x (référentiel Samir).

Controller : modifier la masse de chaque élément, ainsi que la position de son centre de masse relative à l'intersection des barres : variables z_ dans *description_helico.h*. La position du centre de masse par rapport à l'intersection des barres est calculée automatiquement.

Moments d'inertie du robot

Webots : modifier les 6 derniers champs de la matrice d'inertie : CustomRobot → physics → inertiaMatrix : 4ème champ = Iyy, 5ème champ = Izz, 6ème champ = Ixx, 7ème champ = Iyz, 8ème champ = Iyx et 9ème champ = Izx (référentiel Samir).

Controller : modifier les dimensions de chaque élément, ainsi que la position de son centre de masse relative à l'intersection des barres : variables z_ dans *description_helico.h*. Les moments d'inertie suivants sont calculés automatiquement : Ixx, Iyy et Izz.

Configuration de Visual C++ pour programmer un contrôleur

Création d'un projet

Créer un nouveau projet de type Win32 Console Application : spécifier le nom du projet (helico) et l'emplacement du projet (Program Files/Webots/Controllers).
Ajouter un nouveau fichier C/C++ : spécifier le nom (*helico.c*).
Et ensuite, tu codes …

Librairies et includes

- Inclure les fichiers include et library de Webots : Tools → Options → directories include files et library files.
- Inclure le répertoire contenant les fichier *.h* que tu as créé : idem.
- Linker la librairie *contoller.lib* de Webots : Project → Settings → Link → Object/library modules.
- Création du fichier *.exe* directement dans le bon dossier : Project → Settings → Link → Output file name : remplacer Debug/conroler.exe par controler.exe.

Dans Webots

Donner le nom du contrôleur à utiliser (nom du fichier *.exe*) dans le champ controller : CustomRobot → controller.

Si ça ne fonctionne pas depuis Webots (ce qui est le cas chez moi), tu ouvres le fichier *helico.wbt* avec le notepad et tu modifie le nom du contrôleur.

Simulation

Vitesse

- Pour que ça ait l'air réaliste, il vaut mieux simuler en temps réel.
- Si tu veux simuler rapidement pour faire du long terme, il faut réduire la fenêtre Console ou enlever toutes les fonctions d'affichage qui ralentissent la simulation.

Options pour rendre plus réaliste

- Limiter la vitesse des moteurs : variable vmax dans *description_helico.h.* C'est par ex. ce qui fait que l'hélicoptère ne décolle pas vraiment verticalement.
- Limiter l'accélération des moteurs : variable amax dans *description_helico.h.*
- Diminuer la résolution des capteurs ⇔ diminuer la résolution du GPS dans : CustomRobot → children → GPS → resolution.
- Dissymétrie mécanique et dynamique de l'hélicoptère : ne pas donner le même coefficient de portance et de traînée à chaque hélice dans *description_helico.h*.