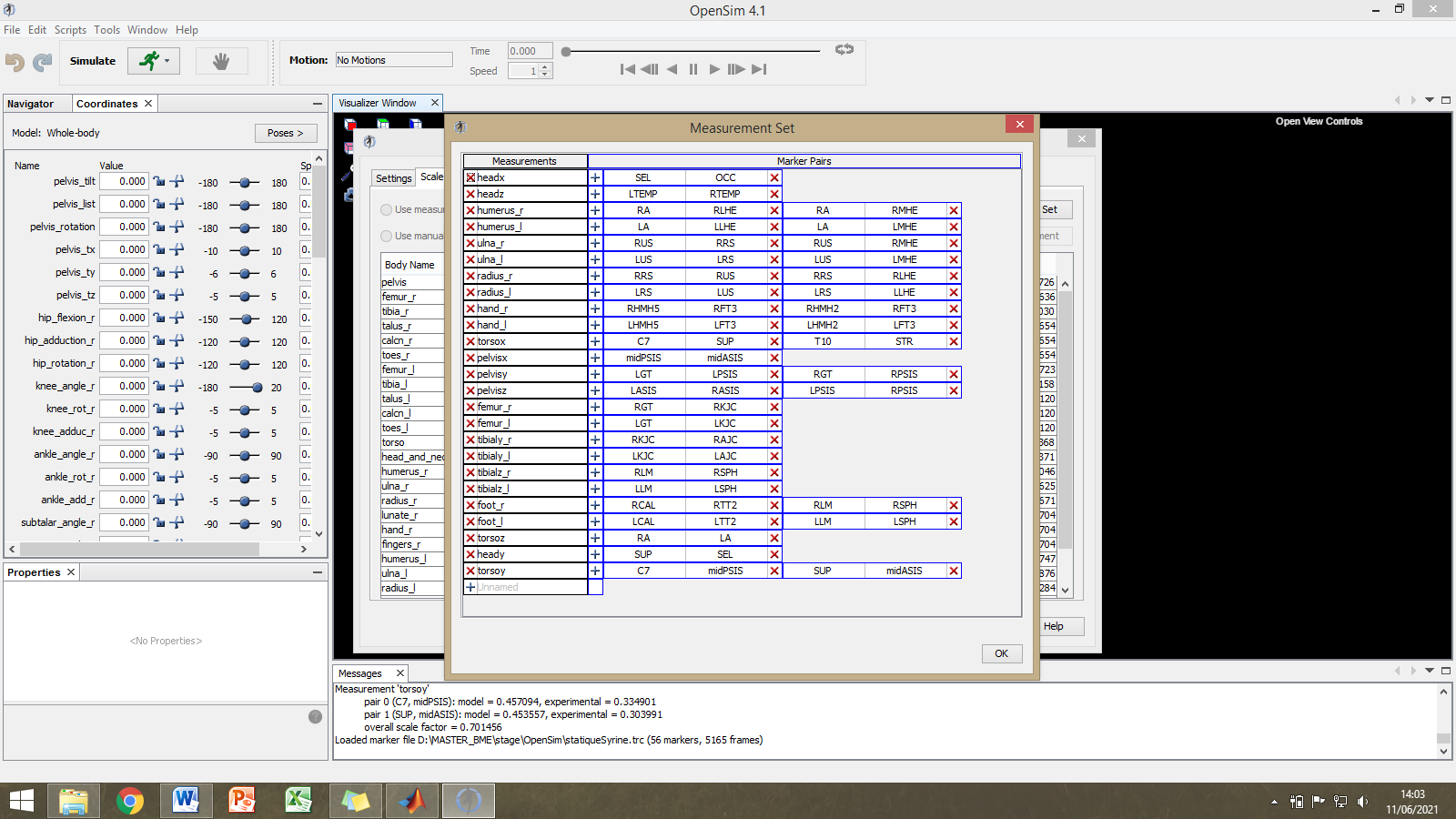
**Obtention des données**

Mise à l’échelle du modèle OpenSim

Modification du modèle pour rajouter certains marqueurs outils (milieu des genoux/chevilles, centre des crêtes iliaques…) et enlever des marqueurs qui n’ont pas été mis expérimentalement. J’ai utilisé le modèle wholebody\_3DoF, sur chaque articulation tous les degrés de liberté ont été libérés. La mise à l’échelle se fait sur quelques frames où les filles ne bougent pas.



Measurement set : permet de définir avec quels couples de marqueurs on fait la mise à l’échelle. Marco l’a très bien expliqué dans son rapport, donc je remets juste les couples de marqueurs utilisés, c’est pas exactement les mêmes que lui, j’en ai enlevé et rajouté en tâtonnant. Ensuite il faut choisir les poids à associer à chaque marqueur : plus un marqueur a un poids élevé, plus on veut que sa position dans le modèle soit proche de sa position réelle. J’ai globalement mis des poids plus élevés au niveau des membres inférieurs vu qu’on voulait pas spécialement la cinématique inverse des membres supérieurs, mais c’était du bidouillage clairement. Les poids étaient pas exactement les mêmes pour Lina et Syrine.

On évalue la mise à l’échelle visuellement et avec la maximum marker error. Globalement, inférieur à 2cm ça passe. J’ai juste mis les membres supérieurs + la tête en mode invisible depuis OpenSim, et j’ai changé leurs propriétés inertielles pour les mettre à zéro et mettre toute la masse sur le tronc. J’ai pas changé la matrice d’inertie du tronc, juste sa masse.

Reconstruction des données

Moitié-moitié Vicon/Matlab. L’étiquetage des marqueurs se fait sur Nexus. Il est possible de créer un squelette à partir d’un essai et ensuite Nexus reconnaît ce squelette pour d’autres marches. Ça a bien marché pour Syrine mais pas pour Lina, les déviations étaient trop importantes. Pour chaque essai, bien s’assurer que chaque marqueur reste labellisé de la même manière tout du long, des fois certains sont inversés à un moment. On peut le vérifier directement sur le modèle ou avec l’outil qui trace la position 3D des marqueurs dans l’espace : si il y a une variation brusque il y a souvent un problème. Pas mal de bons tutos youtube sur le sujet. Une fois la reconstruction faite j’ai interpolé ce qui pouvait l’être, d’abord sur nexus puis sur matlab (intégré au code *main*). Sur Nexus c’est pas compliqué y’a une fonction qui le fait toute seule, mais ça marche pas à tous les coups. Sur matlab j’ai créé ma propre fonction, en partant du principe que je pouvais faire que de l’interpolation, pas d’extrapolation : tous les gaps au début ou à la fin des essais ont été mis de côté. J’ai fait une interpolation linéaire où c’était possible. Ensuite filtré avec un butterworth.

Pour les plateformes j’ai eu des problèmes de bruit sur des plateformes donc j’ai sorti l’artillerie lourde : ça marche, mais c’est un peu pénible à lancer. Une fenêtre apparaît avec les données des plateformes pour l’essai considéré, et il faut dire si oui (1) ou non (0) chaque plateforme est activée. Si on répond non pour une plateforme, ses données sont mises à 0 automatiquement, donc ça enlève beaucoup de pics inutiles. Sur les plateformes actives avec des pics de bruit, qui durent en général 2/3 frames, un code les enlève brutalement. Ensuite on filtre ce qui reste avec un butterworth et on identifie la fenêtre majoritaire où la plateforme est active (= la partie où les valeurs de la plateforme selon l’axe vertical sont négatives, donc qu’il y a un contact). Une fois cette fenêtre d’activité définie, on ne garde les valeurs de plateforme que pour ces frames-là : ça permet d’enlever les oscillations liée au filtrage (comme on passe d’une valeur nulle à une valeur non nulle très rapidement, le filtre provoquait des oscillations). Les données sont ensuite mises dans les fichiers traitables par Opensim dans des codes indépendants.

Cinématique inverse

La cinématique inverse se fait aussi avec opensim. Il faut rentrer un fichier trc qui contient la position dans le temps de tous les marqueurs. J’avais fait les fichiers trc avec Matlab, à partir des données des marqueurs filtrées. Il faut juste faire attention à l’indentation et la ponctuation. Le code correspondant est enregistrementTRC qui prend comme argument le nom de la marche. Dans ce code on fait la rotation entre le repère de Vicon et celui d’opensim, qui ne sont bien sûr pas les mêmes. Cette rotation est indispensable si on veut avoir les résultats OpenSim qui correspondent à quelque chose : la flexion se fait autour de l’axe z d’opensim, donc si c’est pas le bon axe les résultats ne valent rien (ça sent le vécu…). A part ça le code utilise deux fonction du package BTK, la syntaxe est bien précisée sur leur site et assez logique. Pour la cinématique inverse aussi on peut mettre des poids pour réduire la maximum marker error, c’est aussi très subjectif et j’ai choisi à tâtons… une indication pour les poids est également la maximum marker error.

Dynamique inverse

On est sur du processus complexe là. Opensim a besoin de deux fichiers .xml et deux fichiers .mot pour tourner. Un des fichiers .mot, c’est le résultat de la cinématique inverse, pas de modifications à faire. L’autre fichier .mot c’est les forces des plateformes, ce qui est plus compliqué. Un code matlab s’occupe d’écrire les deux fichiers xml et le fichier .mot, c’est write\_grf\_xml\_full.

Le fichier .mot des plateformes doit contenir les données des plateformes où il y a contact avec deux grosses subtilités : les données doivent être exprimées dans le même repère que le repère OpenSim et leur signe doit absolument être inversé : les données fournies par les plateformes sont les forces de réaction du pied sur le sol, mais la dynamique inverse considère les forces du sol sur le pied. Donc on multiplie tout (forces, moments) par -1. On met également dans ce fichier le point de calcul des forces et moments. Ici tout se fait au point fourni par les plateformes et non pas au centre de pression. Le point de calcul des forces/moments est donné dans le fichier c3d dans le repère de la plateforme, il faut donc l’exprimer dans le repère opensim.

Le premier fichier xml appelle le fichier .mot des plateformes de force : pour chaque contact, il définit le point, la force et le moment (selon xyz) correspondants, dire à quel partie du corps sont appliqués tout ça (le calcaneus) et dans quel repère c’est exprimé (ground).

Le deuxième fichier xml appelle le premier fichier xml et le fichier mot des résultats de la cinématique inverse, et indique le time range qui nous intéresse.

Petite subtilité de mon code, les fichires mot et xml générés sont en fait sortis de matlab comme des fichiers txt dont je modifie manuellement l’extension. Quand je générais directement des xml ou mot ils n’étaient pas lisibles par opensim et j’ai pas cherché à batailler plus.

Les cycles intéressants ont été identifiés en regardant principalement sur vicon les contacts, et ensuite entrés dans matlab.

Puissance

J’ai repris les moments utilisés précédemment pour les cycles complets que j’ai filtrés. A partir des données cinématiques filtrées j’ai calculé la vitesse angulaire, qu’il faut bien faire attention à passer en radians. Puis la puissance en multipliant la vitesse angulaire par le moment.

Données EMG

J’ai commencé par découper les signaux de marche en fonction des cycles de marche, pour avoir une vision claire de ce qui se passe à chaque cycle et pas être embêtée par les activations qui continuent sur le cycle suivant. Le code detectEventsCustomMade utilise le code de Nicolas Turpin en le transformant en fonction pour pouvoir l’appeler pour chaque cycle. L’entrée de ce code c’est une estimation à partir du signal filtré de la proportion du signal qui correspond à la baseline. Généralement la baseline correspond à 10 à 50% du signal, ou même 5% quand Lina était toujours activée. Ensuite on vérifie pour chaque cycle que l’estimation fournie par l’algo est convenable, et on corrige les cycles où l’estimation ne l’est pas. C’est un peu long à faire sorry. Ensuite échantillonnage entre 0 et 100 et calculs statistiques classiques.

Pour les formes des EMG le traitement est classique : courbe moyenne et déviation standard. Pour les calculs des activations j’ai séparé, pour chaque muscle, en différents modes d’activation : 1,2,3 ou plus d’activations. Comme on considère un cycle de marche qui commence et finit avec un heel strike, un muscle qui serait activé autour de heel strike aurait deux activations : 0-20% et 80-100% (au pif) alors que l’activation est constante. C’est juste la méthode choisie…

Considérations communes à la plupart de mes codes

Les fichiers eventsSyrine/eventsLina contiennent l’ensemble des évènements de tous les cycles de marche : les heel strikes et les toe offs. Le code toeOffMoyen permet d’avoir la valeur moyenne du pourcentage du cycle où le pied décolle, ce qui rend bien sur les figures.

Les fichiers boundariesSyrine/boundariesLina contient l’ensemble des limites des données qu’on a : pour des raisons de gap filling pas possible en début ou en fin de marche, certaines marches ont été tronquées, et la première valeur (de position de marqueur, de force de plateforme, …) ne correspond pas forcément à t=0.

Toutes les données sont interpolées pour être exprimées en pourcentage du cycle de marche (sauf la dynamique inverse concentrée sur la phase de simple support, là l’échelle temporelle est en pourcentage de la phase de simple support).