Le programme numéro 4 est le plus rapide des 4 programme pour le calcul d’un angle avec à un accéléromètre. Ce programme utilise une table des valeurs du arc-cosinus précalculées, ce qui permet d’alléger le nombre de calculs que le CPU doit effectuer pour retourner le résultat. Cette méthode nécessite de trouver l’indice du tableau contenant le résultat du arccos en fonction de la valeur de la composante en « z » de l’accéléromètre. La valeur du module utilisé pour le calcul de l’angle est lui aussi calculé à l’avance. On notera également que ce programme utilise 5 variables volatiles ce qui ralenti le CPU.

Nous avons ensuite le programme numéro 1 qui est environ 2 fois moins rapide que le précédent. Ce dernier n’utilise pas de table de valeurs précalculées de la fonction arc-cosinus mais effectue lui-même le calcul ce qui ralenti le programme. La valeur du module est quant à elle déjà calculée comme pour le programme numéro 4. On notera que pour celui-ci, seulement 4 variables volatiles sont utilisées, la variable « module » est ici déclarée comme un simple float.

On peut déjà affirmer que le calcul de arc-cosinus ralenti énormément le programme, passant du simple au double.

En troisième position nous avons le programme numéro 3, qui est aussi long que l’utilisation des 2 programmes précédents l’un à la suite de l’autre. Ce programme-ci ne possède aucune donnée précalculée, ainsi il effectue à la fois le calcul du arc-cosinus, mais également le calcul du module en utilisant la fonction racine carrée.

On observe ici que, d’une façon générale, les calculs ralentissent plus ou moins fortement un programme. Précalculer les valeurs d’un calcul permet d’augmenter la vitesse d’un programme ce qui peut être une solution non négligeable dans certaines applications.

Enfin nous avons en dernière position le programme numéro 2 qui est presque identique au programme numéro 1 à la différence qu’il ne possède pas la variable contenant la valeur du module. La valeur est directement ajoutée dans le calcul de l’angle sans passer par la variable module. Cette approche rend le code presque 3 fois plus lent qu’avec l’utilisation d’une variable.

En conclusion de cette observation, précalculer la valeur des calculs, utiliser des variables plutôt que de donner directement la valeur et éviter le plus possible l’utilisation des variables volatiles permet d’augmenter la vitesse du programme, donc de ménager le processeur durant l’exécution de celui-ci. On obtient donc, en appliquant tout ce que l’on a découvert un temps d’exécution encore plus rapide que le programme numéro 4.

