## Codage en ASM

Un candidat à l’optimisation en assembleur est la fonction implémentant le filtre Butterworth d’ordre 2 (« filterN2 » dans filtre.c). Pour chaque trame de 160 échantillons, cette fonction est appelée 17 fois (pour chaque bande de fréquence). La version en C compilée en Configuration « Release » exécute en 5495 cycles.

Nous avons codé une version du filtre en assembleur («\_FilterASM » dans FilterAsm.asm) dans le but de connaître le type et le nombre d’instructions nécessaires pour une version non-optimisée et d’en extrapoler le gain en performance potentiel après optimisation.

La version en assemble non-optimisée exécute (23 instructions + 14 nop) \* 160 = 5920 cycles.

Ce qui est presque identique à la performance obtenue avec la version «Release » en C.

Donc, en remplaçant les « nop » et en plaçant plusieurs instructions en parallèle, un gain de performance significatif (au moins 40%) serait possible.

## Comparaison avec résultats Matlab

Les résultats obtenus sur le simulateur et la carte avec le filtre passe-bande Butterworth d’ordre 2 diffèrent de ceux obtenus avec Matlab. Nous avons généré des fichiers avec le simulateur en appliquant le filtre avec 4, 8 et 17 bandes de notre échelle de fréquences. Nous remarquons que plus nous traitons de bandes avec le filtre, plus le signal semble affecté, jusqu’au point de devenir inintelligible lorsque toutes les 17 bandes de fréquence sont traitées. C’est peut-être dû à une accumulation d’erreurs (le processeur utilise des données en point flottant simple précision alors que Matlab est en double précision). Une autre piste serait une erreur dans la conception du filtre. Nous avons réutilisé du code du site suivant : <http://www-users.cs.york.ac.uk/~fisher/mkfilter/trad.html> en y remplaçant les coefficients et le gain par ceux calculés avec Matlab.

La détection du niveau de bruit diverge aussi des résultats obtenus avec Matlab. Pour le même buffer audio, Matlab détermine que 50% des trames ne contienne que du bruit (sans voix) tandis que le code DSP détermine que 90% des trames ne contiennent que du bruit. Bien que l’implémentation soit identique à celle de Matlab et que nous observions une distribution d’énergie semblable à celle décrite dans [1] (voir figures 1 et 2), il y a peut-être encore des problèmes d’échelle (taille des données d’échantillon) tel que discuté dans le premier rapport de ce projet.



Figure 1 : distribution cumulative de l’énergie du signal « car.wav »



Figure 2 : histogramme de la distribution de l’énergie du signal « car.wav ».

## Références

[1] Robert J. McAulay and Marilyn L. Malpass, (1980) SPEECH ENHANCEMENT USING A SOFT-DECISION NOISE SUPPRESSION FILTER