Solutions techniques

# Choix de l’algorithme

Les algorithmes de calculs choisis sont la DCT-II, la formule la plus couramment utilisée, ainsi que sont inverse la DCT-III ou IDCT. Ces algorithmes sont aussi ceux utilisés par Matlab. La comparaison lors de la phase de test en sera que plus aisée.

DCT :

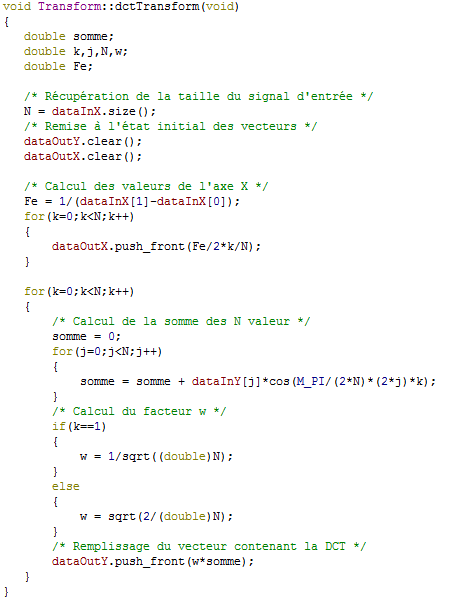
IDCT :

En plus des calculs précédents, formant l’axe y, il faut rajouter le calcul de l’axe x. Pour la DCT, le delta de temps entre deux échantillons est en faite l’inverse de Fe. On peut donc calculer Fe et par conséquent le vecteur x qui ira de 0 à Fe/2 (Shannon). N’affichant que la moitié du spectre de sortie il faudra multiplier par deux l’amplitude de chaque valeur.

Pour l’IDCT, le delta de fréquence représente l’inverse du temps maximal. On aura un vecteur x allant de 0 à deltaT\*N.

# Implémentation

Ci-dessous se trouve un premier exemple d’implémentation en C++ des algorithmes au sein de la classe transform, classe qui se chargera de toute la partie calcul mathématique.

Les deux vector dataIn et dataOut sont des attributs dans lesquels sont respectivement stockées les données d’entrée et de sortie des algorithmes (dataIn sera pré-chargé au préalable avec un signal).

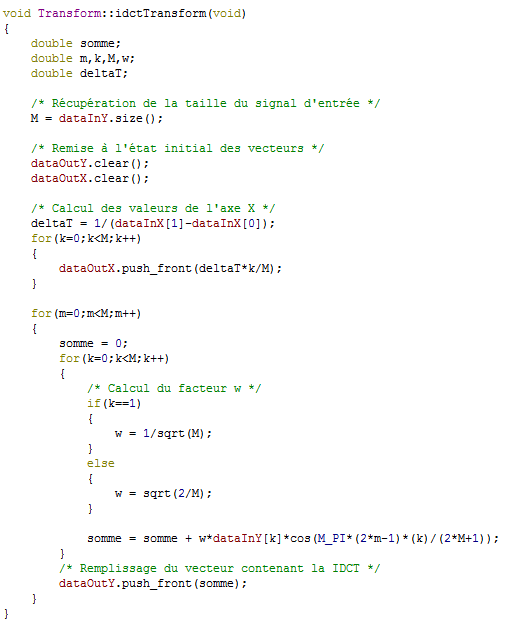
Dans un premier temps il faut récupérer la taille du vector ainsi que « reseter » les vectors de sortie, dataOutX et Y.

Dans un second temps on calcul les valeurs de l’axe X.

Pour chaque échantillons du signal de sortie, boucle incrémentant k, on calcul la somme du signal d’entrée multiplié par un factor en cos.

Vient ensuite le calcul du facteur w, dépendant de k.

A chaque boucle l’instruction dataOut.push\_front(w\*somme) permet de créer une case mémoire de le vector de sortie et ainsi sauvegarder la donner.

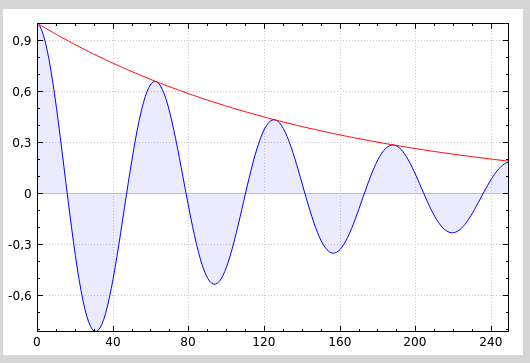


Pour le calcul d’lDCT, la structure est semblable.

L’avantage d’utiliser un type vector est de ne pas être obliger de connaître la taille finale du résultat et surtout de ne pas avoir à gérer l’allocation dynamique qui impliquerait de désallouer et réallouer de la mémoire en cas de débordement. Ceci est transparent pour nous.

# IHM

## Affichage des graphs



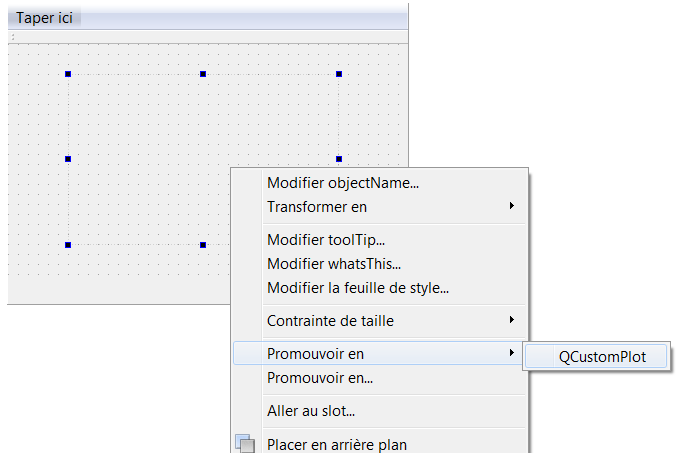
Pour la partie affichage des courbes nous avons choisit d’opter pour une librairie appelée QCustomPlot intégrant un large panel de fonctionnalités allant de l’affichage d’un simple courbe à l’ajustement automatique d’axe, un zoom intégré, …etc. De plus celle-ci est très bien documenté et intégrée à Qt.

### Intégration

1. Ajout des lignes suivantes dans le fichier projet .pro :



(widget nécessaire pour cette librairie)

1. Ajout des fichiers qcustomplot.h et qcustomplot.cpp au projet
2.  créer un widget et le promouvoir en QCustomPlot.