TP 1 : FILTRE DE CONVOLUTION

Introduction

Pour le premier TP, vous allez construire un logiciel qui va filtrer un signal. Le logiciel va recevoir de l'information pour un filtre et le nom d'un fichier contenant le signal à filtrer. Il va placer les résultats dans un fichier indiqué par l'utilisateur.

La section suivante décrit le projet. Ensuite, les modalités de remise et de correction sont décrites.

Description

Entrées

Ligne de commande

Votre programme va lire ces premières entrées sur la ligne de commande. Le langage C++ reçoit les valeurs sur la ligne de commande dans le tableau en paramètres de la fonction `main( int argc, char * argv [] )`. Le paramètre `argv` est un tableau contenant chaque argument de la ligne de commande et l'entier `argc` contient le nombre d'arguments qu'il y a dans le tableau. Par exemple, si vous lancez votre programme (nommé `tp1`) avec la commande suivante.

```
./tp1 entrees.txt sorties.txt -l 450.0 -h 550.0
```

Alors, la fonction main va recevoir les valeurs suivantes : argc = 7, argv = ["tp1", "entrees.txt", "sorties.txt", "-l", "450.0", "-h", "550.0"].

Votre logiciel doit extraire les arguments suivants sur la ligne de commande.

1. Le premier argument est toujours le nom du fichier qui contiendra les entrées.
2. Le deuxième argument est toujours le nom du fichier pour placer les résultats.
3. Ensuite, il y aura soit l'argument "-l" (pour **l**ow), ou "-h" (pour **h**igh), ou les deux.
    a. Ces arguments seront suivis d'un argument représentant une valeur de type `double`.
    b. La ligne de commande peut contenir seulement un des deux arguments (-l ou -h) ou les deux, leurs ordres n'est pas importants. Par contre, ils sont toujours suivis d'une valeur positive et non zéro de type `double`.
    c. Pour le reste de l'énoncé :

   i. Nous utiliserons le nom f_c^l pour désigner la valeur de type `double` qui suit l'argument "-l". Cette valeur représente la fréquence de coupure du filtre passe-bas.
   ii. Nous utiliserons le nom f_c^h pour désigner la valeur de type `double` qui suit l'argument "-h". Cette valeur représente la fréquence de coupure du filtre passe-haut.
4. Il pourrait y avoir l'argument "-f" suivi d'un argument représentant une valeur entière (de type `int`) positive et non zéro. Pour le reste de l'énoncé, nous utiliserons le nom f_e pour désigner la valeur entière qui suit l'argument "-f". Si l'argument n'était pas présent, alors $f_e = 44\ 100$.
5. Il pourrait y avoir l'argument "-a" suivi d'un argument représentant une valeur entière positive et non zéro. Pour le reste de l'énoncé, nous utiliserons le nom a pour désigner la valeur entière qui suit l'argument "-a". Si l'argument n'était pas présent, alors $a = 8\ 000$.
6. Il pourrait y avoir l'argument "-i". Si cet argument est présent, alors les valeurs résultantes seront de type `int`, sinon elles seront de type `double`.

S'il y a une erreur dans l'argument, votre logiciel doit afficher un message d'erreur sur le canal d'erreur (`cerr`) et terminer l'exécution du programme (`exit(x)` ou `x` est une valeur négative).

Fichier d'entrées

Le reste des entrées du programme seront lues dans un fichier. Le nom du fichier contenant ces entrées est donné par le premier argument sur la ligne de commande. Ce fichier contient simplement une suite de valeur de type `double`. Ces valeurs sont séparées par des virgules (CSV). Chaque valeur de la suite sera désignée par x_j. Donc, la première valeur du fichier est désignée par x_1, la deuxième valeur dans le fichier par x_2, etc. Nous utilisons C pour désigner le nombre de valeurs présent dans le fichier.

Si le fichier n'existe pas ou s'il y a des erreurs dans le fichier alors le logiciel affiche un message d'erreur sur le canal d'erreur et termine avec `exit` et une valeur négative.

Comme il est possible qu'il y ait vraiment beaucoup de valeur dans ce fichier, il ne sera pas possible d'allouer un tableau pour toutes les contenir. Nous allons donc seulement garder les valeurs nécessaires en mémoire. Pour cela, nous allons utiliser un tampon de données circulaire.

Traitement

Voici les étapes du traitement que votre logiciel doit faire.

1. Construire le filtre.
   a. Calculer l'ordre du filtre.
   b. Calculer les coefficients du filtre.
2. Convolution.
   a. Chargement initial d'un ensemble de données.
   b. Faire la convolution.

Construction du filtre

Le filtre est une suite de valeur de type `double`. Il est conseillé d'utiliser un tableau pour placer ces valeurs. Dans cet énoncé, nous allons utiliser la lettre h pour désigner le tableau du filtre. Donc, h_i représente la valeur à la position i du filtre. Vous devez déterminer la taille de ce tableau. Nous appelons cette taille l'ordre du filtre (n). Voici comment calculer l'ordre du filtre.

$$\Delta = \frac{a}{f_e}$$
$$n = \frac{0.9}{\Delta}$$

Arrondissez la valeur de n à la valeur impaire la plus proche. Si n est à distance égale de deux valeurs impaires, alors arrondir vers la valeur inférieure.

Il faut maintenant remplir ce filtre avec les coefficients du filtre. Pour cela, nous devons déterminer la formule du filtre. Voici les équations pour calculer chacun des h_i.

$$M = \left\lfloor \frac{n}{2} \right\rfloor$$
$$k = i - M$$
$$\Omega_l = \frac{2\pi f_c^l}{f_e}$$
$$\Omega_h = \frac{2\pi f_c^h}{f_e}$$
$$p_l(k) = \begin{cases} 0 & Si\ l'utilisateur\ n'a\ pas\ donné\ d'argument - \mathrm{l} \\ \frac{\Omega_l}{\pi} & Si\ k = 0 \\ \frac{\sin(\Omega_l k)}{k\pi} & Si\ k \neq 0 \end{cases}$$
$$p_h(k) = \begin{cases} 0 & Si\ l'utilisateur\ n'a\ pas\ donné\ d'argument - \mathrm{h} \\ \frac{\pi - \Omega_h}{\pi} & Si\ k = 0 \\ -\frac{\sin(\Omega_h k)}{k\pi} & Si\ k \neq 0 \end{cases}$$
$$h_i = p_l(k) + p_h(k)$$

Il reste à normaliser le filtre. Pour cela il suffit de faire la somme des coefficients du filtre.

$$s = \sum_{i=0}^{n-1} h_i$$

Et finalement pondérer chaque coefficient.

$$\forall i \in [0..n-1], h_i \leftarrow \frac{h_i}{s}$$

(Pour chaque coefficient du tableau, divisez ce coefficient pas s.)

Convolution

Pour faire la convolution, nous allons utiliser tableau circulaire. Un tableau circulaire est une technique simple et rapide pour garder en mémoire les n dernières valeurs insérées dans le tableau. Les anciennes valeurs sont détruites lors de l'ajout de nouvelle valeur. Simplement :

- Construire un tableau C++ contenant des valeurs de type double. Dans le reste de l'énoncé, nous utilisons x pour désigner ce tableau. Donc, x_j désigne la valeur à la case j de ce tableau. Le tableau doit avoir la même taille que le filtre (soit n).
- Quand vous voulez accéder à la case j du tableau, vous écrivez `x[j % n]` en C++. Ne fonctionne pas si j est négatif.

Votre programme devra premièrement charger du fichier d'entrées $\left\lceil \frac{n}{2} \right\rceil$ valeurs dans le tableau. S'il n'y a pas assez de valeur dans le fichier d'entrées, insérez des zéros dans le tableau. Ensuite, il reste à faire la convolution.

L'algorithme de convolution est très simple.

- Pour j = 0 à C-1
  - Calculez

$$y_j = \sum_{i=0}^{n-1} h_i \cdot \begin{cases} 0 & si\ (j - M + i) < 0 \\ x_{(j-M+i)} & sinon \end{cases}$$

  - Écrivez y_j dans le fichier de sortie sous forme d'un `double`, sauf si l'utilisateur à placé l'argument "-i" sur la ligne de commande, alors c'est un `int` arrondi vers le bas.
  - Écrivez une virgule si $j \neq C - 1$.
  - Lisez une valeur dans le fichier d'entrées et ajoutez-la dans le tableau x. S'il n'y a plus de valeurs dans le fichier d'entrées, ajoutez un zéro.

Exemple

Soit un fichier nommé entrees1.txt, contenant l'information suivante.

```
5.0, 2.43, -4.1, -9.3, -1.3, 0.2, 0.4, 4.5
```

Le logiciel est lancé avec la commande suivante.

```
./tp1 entrees1.txt sorties1.txt -l 20.0 -f 100.0 -a 19.0
```

Construire le filtre

Le programme commence par calculer la taille du filtre.

$$\Delta = \frac{a}{f_e} = \frac{19}{100} = 0.19$$

$$n = \frac{0.9}{\Delta} = \frac{0.9}{0.19} = 4.7368421052631575$$

La valeur est arrondie au nombre impair le plus près : $n = 5$.

Ensuite, il faut calculer les coefficients pour le filtre.

$$M = \left\lfloor \frac{n}{2} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = \lfloor 2.5 \rfloor = 2$$
$$\Omega_l = \frac{2\pi f_c^l}{f_e} = \frac{2\pi 20}{100} = \frac{125.66370\ldots}{100} = 1.2566370614359172$$

(Pas besoin de calculer Ω_h puisque l'argument -h n'était pas présent.)

Calculez les valeurs de h_i pour i de 0 à 4 (car $n = 5$).

- $i = 0$:

$$k = i - M = 0 - 2 = -2$$

$$\begin{aligned} h_0 &= p_l(k) + p_h(k) \\ &= p_l(-2) + p_h(-2) \\ &= p_l(-2) + 0 \\ &= p_l(-2) \end{aligned}$$

Car $p_h(-2) = 0$, puisque l'argument -h n'était pas présent.

$$\begin{aligned} p_l(-2) &= \frac{\sin(\Omega_l k)}{k\pi} \\ &= \frac{\sin(1.2566370\ldots \times -2)}{-2\pi} \\ &= \frac{\sin(-2.513274122\ldots)}{-6.2831853\ldots} \\ &= \frac{-0.5877852\ldots}{-6.2831853\ldots} \\ &= 0.09354892837886393 \end{aligned}$$

- $i = 1 : p_l(-1) = 0.3027306914562628$
- $i = 2 : p_l(0) = 0.4$
- $i = 3 : p_l(1) = 0.3027306914562628$
- $i = 4 : p_l(2) = 0.09354892837886393$

Il reste à normaliser le filtre.

$$s = \sum_{i=0}^{n-1} h_i = 0.09354892837886393 + 0.3027306914562628 + 0.4 + 0.3027306914562628 + 0.09354892837886393 = 1.1925592396702536$$

Finalement, diviser chaque valeur du filtre par s.

h = [0.07844384183776942, 0.25384960460326383, 0.33541310711793343, 0.25384960460326383, 0.07844384183776942]

Convolution

Nous sommes prêts pour l'étape de convolution.

Prendre les $\left\lceil \frac{n}{2} \right\rceil = 3$ premières valeurs du fichier et les placers dans notre tableau circulaire (x).

x :

0	1	2	3	4
5.0	2.43	-4.1		

Faire la convolution à partir de j = 0.

- j = 0 :

  Calculer

  $$y_0 = \sum_{i=0}^{n-1} h_i \cdot \begin{cases} 0 & si\ (j - M + i) < 0 \\ x_{(j-M+i)} & sinon \end{cases}$$

  $$y_0 = 0.0784 \ldots \times 0.0 + 0.2538 \ldots \times 0.0 + 0.3354 \ldots \times 5.0 + 0.2538 \ldots \times 2.43 + 0.0784 \ldots \times -4.1$$
  $$= 1.9723003232407434$$

  Ensuite, ajouter une valeur dans le tableau x.

x :

0	1	2	3	4
5.0	2.43	-4.1	-9.3	

- j = 1 :

  $$y_1 = 0.0784 \ldots \times 0.0 + 0.2538 \ldots \times 5.0 + 0.3354 \ldots \times 2.43 + 0.2538 \ldots \times -4.1 + 0.0784 \ldots \times -9.3$$
  $$= 0.3139907653482602$$

  Ensuite, ajouter une valeur dans le tableau x.

x :

0	1	2	3	4
5.0	2.43	-4.1	-9.3	-1.3

- j = 2 :

  $$y_2 = 0.0784 \ldots \times 5.0 + 0.2538 \ldots \times 2.43 + 0.3354 \ldots \times -4.1 + 0.2538 \ldots \times -9.3$$
  $$+ 0.0784 \ldots \times -1.3 = -2.828898308008203$$

  Ensuite, ajouter une valeur dans le tableau x, puisque la dernière valeur a été ajoutée à la fin, la prochaine valeur s'ajoute au début (tableau circulaire).

x :

0	1	2	3	4
0.2	2.43	-4.1	-9.3	-1.3

- j = 3 :

$$y_3 = 0.0784 \ldots \times 2.43 + 0.2538 \ldots \times -4.1 + 0.3354 \ldots \times -9.3 + 0.2538 \ldots \times -1.3 + 0.0784 \ldots \times 0.2 = -4.283822457021072$$

Ensuite, ajouter une valeur dans le tableau x.

x :

0	1	2	3	4
0.2	0.4	-4.1	-9.3	-1.3

- j = 4 :

$$y_4 = 0.0784 \ldots \times -4.1 + 0.2538 \ldots \times -9.3 + 0.3354 \ldots \times -1.3 + 0.2538 \ldots \times 0.2 + 0.0784 \ldots \times 0.4 = -3.0363106559427613$$

Ensuite, ajouter une valeur dans le tableau x.

x :

0	1	2	3	4
0.2	0.4	4.5	-9.3	-1.3

- j = 5 :

$$y_5 = 0.0784 \ldots \times -9.3 + 0.2538 \ldots \times -1.3 + 0.3354 \ldots \times 0.2 + 0.2538 \ldots \times 0.4 + 0.0784 \ldots \times 4.5 = -0.537912463540644$$

Ensuite, ajouter une valeur dans le tableau x. Puisqu'il n'y a plus de valeur dans le fichier, nous ajoutons un 0.

x :

0	1	2	3	4
0.2	0.4	-4.1	0.0	-1.3

- j = 6 :

$$y_6 = 0.0784 \ldots \times -1.3 + 0.2538 \ldots \times 0.2 + 0.3354 \ldots \times 0.4 + 0.2538 \ldots \times 4.5 + 0.0784 \ldots \times 0.0 = 1.225281390093413$$

Ensuite, ajouter une valeur dans le tableau x. Puisqu'il n'y a plus de valeur dans le fichier, nous ajoutons un 0.

x :

0	1	2	3	4
0.2	0.4	-4.1	0.0	0.0

- j = 7 :

$$y_7 = 0.0784 \ldots \times 0.2 + 0.2538 \ldots \times 0.4 + 0.3354 \ldots \times 4.5 + 0.2538 \ldots \times 0.0 + 0.0784 \ldots \times 0.0 = 1.6265875922395598$$

Nous venons de calculer la convolution pour la valeur -4.1 (car -4.1 a été multiplié par la valeur au centre du tableau h), qui est la dernière valeur du fichier. Nous avons donc terminé.

Le fichier sorties.txt devrait contenir l'information suivante :

```
1.9723003232407434, 0.3139907653482602, -2.828898308008203,
-4.283822457021072, -3.0363106559427613, -0.537912463540644,
1.225281390093413, 1.6265875922395598
```

Modalités

Directives

1. Le TP est à faire seul ou en équipe de deux.
2. Le TP est à remettre avant 23 :59, le 14 juin.
3. Code :
    a. Pas de `goto`.
    b. Pour une fonction :
        i. Additionnez le nombre de `if`, `for`, `while`, `switch` et `try`. Ce nombre ne doit pas dépasser 10.
        ii. Un seul `return`.
    c. Seules les librairies standards de C++ peuvent être utilisées. Pour ce TP, vous avez seulement besoin de cmath.
4. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
5. Commentaires
    - Commentez l'entête de chaque classe et fonction.
    - Une ligne contient soit un commentaire, soit du code, pas les deux.
    - Utilisez des noms d'identificateur significatif.
    - Une ligne de commentaire ne devrait pas dépasser 120 caractères. Continuez sur la ligne suivante au besoin.
    - Plus en détail
        - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la fonction ou la classe.
            - Courte.
            - Complète.
            - Commencez la description avec un verbe.
            - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
        - Ensuite, au besoin, une description détaillée de la fonction ou classe va suivre.
            - Indépendant du code. Les commentaires d'entêtes décrivent ce que la fonction fait, ils ne décrivent pas comment c'est fait.
            - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
        - Ensuite, placez une ligne vide.

- Décrivez les paramètres de la fonction.
- Écrivez les commentaires à la troisième personne, EN FRANÇAIS.

Remise

Remettre le TP par l'entremise de Moodle. Pour ce TP, votre code sera dans un seul fichier. Votre code va être compilé avec la commande g++ -lm nomFichier.cpp, sur les serveurs Java de l'université.

Évaluation

- Fonctionnalité (10 pts) : votre programme doit compiler et fonctionner sur les serveurs Java de l'université. Si le logiciel ne compile pas (g++ -lm nomFichier.cpp), il n'y a pas de point pour la fonctionnalité.
- Structure (3 pt) : suivez les directives.
- Lisibilité (2 pts) : commentaire (directives), indentation et noms d'identificateur significatif.

Référence

Mounir Boukadoum, Michaël Ménard, « Filtres à réponse impulsionnelle finie (RIF) ».