Travail pratique sur les intégrales

1 But du travail et rendu

Le but de ce travail pratique est d'implémenter les méthodes numériques de calcul d'intégrales que nous avons vues en cours, afin de les comprendre de façon un peu plus approfondie. Puis, il faudra utiliser ces méthodes pour calculer des convolutions afin de filtrer un signal.

Le noyau de calcul de votre travail pratique doit être réalisé en C et il doit être compilable avec un `Makefile`. Pour la réalisation de graphiques vous êtes libres d'utiliser l'outil de votre choix (la librairie matplotlib par exemple). Certaines visualisations peuvent être faites à l'aide de la librairie SDL pour obtenir un bonus.

Vous devrez rendre un petit rapport (environ 10 pages) qui explique ce que vous avez fait et dans quel but. Il devra contenir en premier lieu une introduction générale à votre travail qui peut être comprise par n'importe qui. Puis une courte introduction théorique (rappelant les formules et le but du travail), une partie expliquant dans les grandes lignes des algorithmes (pas de copier-coller du code, pas de captures d'écran non plus sous peine de sanctions terribles). La partie importante est celle contenant les résultats obtenus et leur discussion. Finalement, il faut inclure une conclusion résumant votre travail.

Le travail doit être effectué par groupes de deux (oui c'est une obligation). N'oubliez pas de mentionner les deux noms sur le rapport et dans le code. Je dois pouvoir exécuter le code afin de pouvoir reproduire les résultats présentés dans le rapport (un petit *readme* pour les instructions est le bienvenu). Je dois aussi pouvoir définir ma propre fonction à intégrer de façon simple. Le rapport et le code doivent être déposés sur `Cyberlearn` et `gitedu` respectivement.

La note sera une combinaison entre le code rendu et le rapport (moitié/moitié).

2 Intégration numérique

2.1 Méthodes d'intégration

Dans un premier temps, le but est donc d'écrire un code où l'utilisateur spécifie une fonction $f(x)$ (il écrira lui-même le code de la dite fonction) qu'on suppose "gentille'' (pas besoin de vérifier si elle est bien définie partout par exemple), un intervalle $[a, b]$, et un nombre de subdivisions N. Le code devra rendre la valeur

numérique obtenue pour l'intégrale de la fonction

$$I(a, b, N, f(x)) \cong \int_a^b f(x)\mathrm{d}x \tag{1}$$

pour deux méthodes vues en cours (méthode du rectangle à gauche, et méthode du trapèze) [1]. Dans le cas de la méthode du rectangle à gauche on a

$$I(a, b, N, f(x)) = \sum_{i=0}^{N-1} f(a + i\delta x)\delta x, \quad \delta x = \frac{b-a}{N}. \tag{2}$$

2.2 Validation

Lorsqu'on calcule numériquement une intégrale, on souhaite que la valeur de celle-ci *converge*. C'est-à-dire que plus N est grand, plus la valeur de l'approximation est bonne. Pour vérifier que cela se passe avec nos méthodes d'intégration, vous devrez effectuer une étude de l'erreur de vos méthodes d'intégration numériques. Pour ce faire, nous choisissons une fonction $f(x)$ dont la primitive est simple à calculer

$$f(x) = -\frac{2x-1}{\exp(x^2 - x + 2)}, \tag{3}$$

3 et un intervalle sur lequel la fonction est bien définie. Choisissons ici $[a, b]$ avec $a = 1$ et $b = 5$. On peut donc calculer l'intégrale exactement (analytiquement) et on notera ce résultat exact I

$$I = \int_a^b -\frac{2x-1}{\exp(x^2 - x + 2)}\mathrm{d}x. \tag{4}$$

Puis, il faut calculer l'erreur commise par l'évaluation de la fonction $I(a, b, N, f(x))$ pour $N = 5, 10, 50, 100, 500, 1000$ pour chacune des méthodes que vous avez implémentées ci-dessus. L'erreur $E(N)$ se calcule de la façon suivante

$$E(N) = \left| \frac{I - I(a, b, N, f(x))}{I} \right| \tag{5}$$

Ces résultats devront être illustrés sous forme de graphique (E en fonction de N en échelle log-log). Que constatez-vous? Pouvez-vous mesurer le taux de décroissance de l'erreur (a.k.a. l'ordre de l'erreur)?

3 Convolutions et filtrage

Nous voulons à présent essayer de voir comment utiliser la convolution pour filtrer un signal simple.

3.1 Convolution en 1 dimension

3.1.1 La convolution continue

Le signal que nous souhaitons filtrer est définit par la fonction $s(x)$

$$s(x) = \sin(2\pi\omega_1 x) + \sin(2\pi\omega_2 x). \tag{6}$$

1. Vous retrouverez les formules dans le polycopié.

La fonction de filtrage que nous allons utiliser est définie par $f(x)$

$$f(x) = \begin{cases} \frac{1}{\psi}, & \text{si } x \in [-\psi/2, \psi/2] \\ 0, & \text{sinon.} \end{cases} \tag{7}$$

Afin de se familiariser un peu avec ces deux fonctions, dessiner les pour différentes valeur de ω_1, ω_2, et ψ.

Puis, calculer analytiquement (à la main avec du papier et un crayon [2]) la convolution $(f * s)(x)$ (rappelez vous qu'on a fait des choses similaires en cours). Ensuite, on souhaite filtrer s à l'aide de f. Pour cela, on veut "enlever" totalement la partie ω_1 (ou ω_2) de $f * s$. En choisissant astucieusement ψ on se rend compte que cela est plus simple qu'il n'y paraît! Utiliser ces relations pour illustrer le filtrage de $s(x)$ pour différentes valeurs de ω_1 et ω_2.

3.1.2 La convolution discrète

Utiliser une des méthodes implémentées dans le chapitre précédent pour calculer numériquement le filtrage de $s(x)$ par la fonction $f(x)$ pour différentes valeurs de ω_1, ω_2 et ψ (essayer par exemple de reproduire les résultats de la section précédente). Puis, répétez l'opération avec une autre fonction de filtrage $h(x)$ définie par

$$h(x) = \frac{1}{\sqrt{2\pi\psi}} \exp(-x^2/(2\psi)). \tag{8}$$

Voyez-vous des différences?

3.2 Convolution en 2 dimensions

3.2.1 Théorie

Dans le cadre de ce tp, nous allons nous concentrer sur la convolution discrète d'un signal discret en deux dimensions. Pour représenter notre signal discret en deux dimensions, nous pouvons utiliser des matrices. Par exemple :

$$\underline{\underline{A}} = \begin{pmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{pmatrix}, \quad \underline{\underline{B}} = \begin{pmatrix} b_{-2,-2} & b_{-2,-1} & b_{-2,0} & b_{-2,1} & b_{-2,2} \\ b_{-1,-2} & b_{-1,-1} & b_{-1,0} & b_{-1,1} & b_{-1,2} \\ b_{0,-2} & b_{0,-1} & b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,-2} & b_{1,-1} & b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,-2} & b_{2,-1} & b_{2,0} & b_{2,1} & b_{2,2} \end{pmatrix} \tag{9}$$

Une image matricielle peut être interprétée comme un signal discret en deux dimensions.

Pour rappel, la formule du produit de convolution en 1 dimension d'un signal discret est :

$$(s * u)[t] = \sum_{n=-N}^{+N} s[n] \cdot u[t-n] \tag{10}$$

2. Exceptionnellement un stylo est également toléré.

Lorsque l'on rajoute une nouvelle dimension la formule devient, avec $-M$ l'indice de ligne le plus petit de la matrice $\underline{\underline{A}}$, et $+M$ le plus grand, ainsi que $-N$, $+N$ pour les indices de colonne :

$$(\underline{\underline{A}} * \underline{\underline{B}})[i,j] = \sum_{m=-M}^{+M} \sum_{n=-N}^{+N} \underline{\underline{A}}[m,n] \cdot \underline{\underline{B}}[i-m, j-n] \tag{11}$$

Si on reprend par exemple les matrices $\underline{\underline{A}}$ et $\underline{\underline{B}}$ de l'équation 9, et qu'on choisit le centre (1,1), on obtient :

$$\begin{aligned} (\underline{\underline{A}} * \underline{\underline{B}})[1,1] &= \sum_{m=-1}^{+1} \sum_{n=-1}^{+1} \underline{\underline{A}}[m,n] \cdot \underline{\underline{B}}[1-m, 1-n] \\ (\underline{\underline{A}} * \underline{\underline{B}})[1,1] &= a_{-1,-1} \cdot b_{2,2} + \cdots + a_{1,1} \cdot b_{0,0} \end{aligned} \tag{12}$$

Si l'on essaye de calculer $(\underline{\underline{A}} * \underline{\underline{B}})[2,2]$, on découvre qu'il nous faut des valeurs qui ne sont pas dans notre matrice, comme par exemple, $b_{3,3}$. Voici 3 solutions différentes pour définir nos valeurs manquantes :

- Les valeurs en dehors de la matrice sont nulles, $b_{3,3} = 0$.
- Recopier la valeur du voisin le plus proche, $b_{3,3} = b_{2,2}$.
- Définir que notre signal est périodique, on a donc $b_{3,3} = b_{-2,-2}$.

3.2.2 Exercice

Au risque de se répéter, rappelons quelques contraintes administratives.

Vous allez devoir implémenter une solution de traitement d'images en nuances de gris basé sur la convolution de signal en deux dimensions par groupe de deux. Le language imposé est le C. Vous devrez rendre le code sur `gitedu` ainsi qu'un rapport succinct sur `Cyberlearn` (moins de 10 pages par groupe).

Pour commencer, vous devrez implémenter un outil permettant de lire des images au format PGM (n'hésitez pas à réutiliser celui que vous avez déjà implémenté en programmation séquentielle). Vous utiliserez le format binaire pour stocker la valeur de vos pixels. Votre programme devra respecter les spécifications du format PGM (http://netpbm.sourceforge.net/doc/pgm.html).

Pour visualiser votre image, vous pouvez à choix l'afficher avec la librairie SDL (bonus sur la note finale) ou alors la sauvegarder au format PGM, et utiliser un outil compatible (par ex: ImageMagick).

Lorsque vous appliquerez les différents filtres, si vous tombez sur des valeurs inférieures à 0 ou supérieur à votre valeur maximale (par ex: $2^{16} - 1$); vous mettrez la valeur cohérente la plus proche, 0 si < 0 et max si $> max$.

3.2.2.1 Partie 1 Calculez à la main le produit de convolution de ces deux matrices, en utilisant la méthode de votre choix pour la gestion des bords :

$$\underline{\underline{A}} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \underline{\underline{B}} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

3.2.2.2 Partie 2 Appliquez les 5 filtres ci-dessous en faisant le produit $\underline{\underline{F_n}} * \underline{\underline{\mathcal{I}}}$, où $\underline{\underline{\mathcal{I}}}$ est l'image "part2.pgm" jointe à l'énoncé. Expliquez avec vos mots l'effet de ces filtres, est essayant d'être le plus descriptif possible (évitez les phrases de 3 mots).

$$\underline{\underline{F_0}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \underline{\underline{F_1}} = \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \underline{\underline{F_2}} = \frac{1}{256} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

$$\underline{\underline{F_3}} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \underline{\underline{F_4}} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

3.2.2.3 Partie 3 Récupérez sur cyberlearn l'image nommée `part3_<n>.pgm`, où `n` est votre numéro de groupe. Cette image a été fortement bruitée, heureusement (quelle chance vraiment :)), le bruit est périodique, et peut être supprimé à l'aide d'un filtre moyenneur.

$$\underline{\underline{F}} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Pour débruiter l'image vous devez lui appliquer le filtre $\underline{\underline{F}}$. Afin d'éviter les problèmes liés à la gestion des bords, vous n'afficherez que la partie interne de l'image filtrée. Autrement dit, vous supprimerez les 2 premières ainsi que les 2 dernières lignes et colonnes (si votre image fait 100x100, vous garderez le centre de taille 96x96).

Si vous obteniez malgré tout une image totalement grise, vous pourriez utiliser la fonction suivante pour normaliser les valeurs de votre matrice $x' = (2^n - 1) \cdot \frac{x - min(\mathcal{I})}{max(\mathcal{I}) - min(\mathcal{I})}$ [3] pour obtenir une image n bits. Cette fonction permet de maximiser l'écart entre les différents niveaux de gris qui composent votre image, ce qui a pour effet d'accentuer les différences.

3. x' la nouvelle valeur de votre pixel, x l'ancienne valeur de votre pixel, $\mathcal{I}$ la matrice de votre image.