IMA TP2 : Équation de la chaleur - Convolution avec une gaussienne

pierre.maurel@irisa.fr

`http://www.normalesup.org/~pmaurel/IMA/`

1 Préambule

1.1 Introduction

Un exemple classique d'équation aux dérivées partielles est l'équation de la chaleur. Elle s'écrit, pour une fonction initiale u_0 de $\Omega \subset \mathbb{R}^n$ dans $\mathbb{R}$:

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}) \end{cases}$$

avec des conditions aux bords de Ω. La solution de cette équation est une fonction u de $\Omega \times \mathbb{R}_+$ dans $\mathbb{R}$, qui décrit la propagation au cours du temps t de la chaleur dans un milieu isotrope.

L'objectif de ce TP est de mettre en œuvre la diffusion linéaire exprimée par l'équation de la chaleur sur des images en niveau de gris ainsi que sur des images en couleur. Cela doit permettre de vérifier le résultat important suivant : la solution de l'équation de la chaleur à l'instant t est donnée par la convolution avec une gaussienne d'écart-type $\sigma = \sqrt{2t}$.

1.2 But du TP

Le travail demandé dans ce TP se décompose en trois parties :
- implémentation du schéma de discrétisation de l'équation de la chaleur 1D
- implémentation de la diffusion linéaire par l'équation de la chaleur 2D
- implémentation de la convolution par un noyau gaussien et vérification du résultat vu en cours.

En utilisant la fonction `edit`, commencez par créer un fichier `demo_tp2.m` depuis lequel vous appellerez les fonctions créées pour ce TP, ainsi que toutes les fonctions qui vous sembleront utiles pour illustrer vos résultats. Vous pouvez commencer votre fichier par les commandes suivantes : `clc; clear; close all;` (`help` pour en savoir plus).

2 Équation de la chaleur pour un signal 1D

On commence par considérer le cas $n = 1$. L'équation de la chaleur s'écrit alors pour $x \in \mathbb{R}$:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \\ u(x,0) = u_0(x) \end{cases}$$

On commence par définir les pas de discrétisation ainsi que le signal-test :

```
dx = 0.05; % pas spatial de discrétisation
dt = 0.0001; % pas temporel de discrétisation
r = dt/(dx*dx);
x=-1:dx:1; % enlevez le ; final pour visualiser la valeur de x
u0 = sin(x*5)+x; % signal 1D de test
plot(x,u0,'-*'); % affichage du signal
```

Nous allons utiliser le schéma de discrétisation étudié en cours :

$$u_k^{n+1} = (1-2r)u_k^n + r(u_{k+1}^n + u_{k-1}^n) \quad \text{avec} \quad r = \frac{\Delta t}{\Delta x^2} \tag{1}$$

Exercice 1 Dans un fichier `diffusion1D.m`, programmez une fonction `diffusion1D` dont la première ligne sera :

```
function u_n = diffusion1D(u, n, dt, dx)
```

qui prend en argument un signal 1D `u`, un entier `n` et les 2 pas de discrétisation `dt` et `dx`. Cette fonction doit renvoyer le résultat du schéma numérique (1) appliqué n fois au signal u avec les pas dt et dx.

Indications : Vous utiliserez les conditions aux bords dites "de Neumann" qui suppose que les dérivées sur signal sont nulles aux bords. Cela revient à prolonger l'image par réflexion et donc à prolonger à gauche (en 0) et à droite (en $L+1$) ainsi : $u_0^n = u_1^n$ et $u_{L+1}^n = u_L^n$ (si L est la taille du signal).

On pourrait donc calculer u_1 à partir de u_0 avec une boucle `for` de la manière suivante :

```
%calcul de u1 avec une boucle for
L = length(u0);
u1_for(1) = (1-2*r)*u0(1) + r*(u0(2)+u0(1));
for i=2:L-1
  u1_for(i) = (1-2*r)*u0(i) + r*(u0(i+1)+u0(i-1));
end
u1_for(L) = (1-2*r)*u0(L) + r*(u0(L)+u0(L-1));
```

Mais de manière générale, il faut éviter au maximum les boucles en Matlab. En particulier pour les opérations vectorielles ou matricielles un gain de temps considérable peut être obtenu en préférant les fonctions et opérateurs Matlab à une écriture explicite via des boucles. On préférera donc calculer u_1 ainsi :

```
%calcul de u1 sans boucle for
% on décale u vers la gauche et on recopie le dernier à la fin
u_gauche = [u0(2:end) u0(end)];
u_droite = [u0(1) u0(1:end-1)]; % même principe dans l'autre sens
u1 = (1-2*r)*u0 + r*(u_gauche+u_droite);
```

Exercice 2 Tester votre fonction dans `demo_tp2.m` et afficher le résultat pour différentes valeurs de n. Vous pourrez également modifier la fonction `diffusion_1D` pour afficher l'évolution du signal au cours du temps.

Exercice 3 Testez différentes valeurs de r, Δx et Δt et vérifiez le résultat de stabilité prouvé en cours.

3 Équation de la chaleur pour une image 2D

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ u(x, y, 0) = u_0(x, y) \end{cases}$$

Deux images de test sont proposées : `tigre.jpg` (niveaux de gris) et `fleur.png` (couleurs). Pour charger ces images dans Matlab, vous utiliserez la fonction `imread`, puis convertirez ces images en double précision. Dans le cas de l'image couleur il faudra travailler sur les trois canaux de couleurs séparément. Pour réaliser ces opérations, vous pouvez inclure les lignes suivantes dans le fichier `demo_tp2.m` :

```
I1=double(imread('tigre.jpg'));

I2=double(imread('fleur.png');
IR2=I2(:,:,1);
IG2=I2(:,:,2);
IB2=I2(:,:,3);
```

En ce qui concerne l'image en couleur, vous pourrez alors travailler sur les trois images `IR2`, `IG2` et `IB2`.

Pour visualiser vos images, vous pourrez utiliser la commande `imshow` (voir pense-bête).

Dans le cas d'une image, on choisit $\Delta x = \Delta y = 1$ et on note $u^n_{i,j} = u(i, j, n\Delta t)$. En raisonnant de la même manière que dans le cas 1D, on peut montrer que le schéma de discrétisation suivant est consistant à l'équation de la chaleur dans le cas 2D :

$$u^{n+1}_{i,j} = (1 - 4\Delta t)u^n_{i,j} + \Delta t(u^n_{i+1,j} + u^n_{i-1,j} + u^n_{i,j+1} + u^n_{i,j-1}) \tag{2}$$

Exercice 4 Complétez la fonction `diffusion2D` qui prend en argument une image `I`, un entier `n` et le pas de discrétisation `dt`. Cette fonction doit renvoyer le résultat du schéma numérique (2) appliqué n fois à l'image I avec le pas de temps dt.

Exercice 5 Tester cette fonction dans `demo_tp2.m` et afficher le résultat pour différentes valeurs de n pour les deux images de test. Vous pourrez également modifier la fonction `diffusion_2D` pour afficher l'évolution des images au cours du temps. Pour afficher une image couleur `I` (qui doit être de taille $p \times q \times 3$) il faut d'abord la reconvertir en `uint8` (entiers non signés sur 8 bit) : `imshow(uint8(I))`.

Exercice 6 Tester différentes valeurs de Δt. Que peut-on supposer sur la condition sur Δt nécessaire pour avoir la stabilité du schéma considéré ?

4 Convolution avec un noyau gaussien

Cette partie consiste à mettre en œuvre la diffusion linéaire en appliquant une convolution entre l'image traitée et un noyau gaussien, puis à vérifier que la solution de l'équation de la chaleur (obtenue dans la partie précédente) peut bien être obtenue par cette opération.

Pour cela, nous aurons besoin de définir un noyau gaussien. Un noyau gaussien est l'approximation d'une fonction gaussienne définie sur un nombre fini de pixels (une gaussienne tend vers 0 lors que x tend vers $\pm\infty$). On choisit donc un entier impair `taille_noyau` et on définit un noyau de taille `taille_noyau`×`taille_noyau` et d'écart-type σ de la manière suivante :

```
function Ng = noyau_gaussien(sigma, taille_noyau)
demi_cote = (taille_noyau-1)/2;
x = -demi_cote:demi_cote;
y = -demi_cote:demi_cote;
[X,Y] = meshgrid(x,y);
% on applique la formule de la gaussienne
Ng = exp( -(X.^2+Y.^2) / (2*sigma^2) );
Ng = Ng / sum(Ng(:));% on normalise pour avoir sum(Ng(:))=1
```

Exercice 7 Implémentez cette fonction dans un fichier `noyau_gaussien.m`. Ensuite, dans un nouveau fichier, nommé `gauss.m`, programmez une fonction `gauss` dont les paramètres sont les suivants :

- *Entrées* : l'image à traiter, l'écart-type du noyau gaussien et la taille du noyau notée
- *Sortie* : l'image après convolution.

Cette fonction doit effectuer la convolution 2D entre l'image et un noyau gaussien défini par les paramètres de la fonction, au moyen de la commande `conv2`.

Testez cette fonction. En particulier regardez les dimensions de l'image de sortie de votre fonction. Utilisez la commande `help conv2` pour corrigez votre fonction et obtenir une image après convolution de même dimension que l'image initiale.

Exercice 8 Vous testerez votre fonction pour différentes valeurs d'écart-type (en utilisant une taille de noyau égale à $8\sigma + 1$ ce qui permet d'assurer une bonne approximation mais ralentit les calculs), sur les images `tigre.jpg` et `fleur.png` en l'appelant depuis le fichier `demo_tp2.m`.

Exercice 9 Vérifiez le résultat rappelé dans l'introduction : la solution de l'équation de la chaleur à l'instant t est donnée par la convolution avec une gaussienne d'écart-type $\sigma = \sqrt{2t}$. Vous pourrez afficher la différence entre les deux images.

Exercice 10 Que remarque-t-on aux bords de l'image ? Comment peut-on corriger ce problème ? Essayer d'implémenter votre solution.