Projet de programmation numérique

Contact : fdang@aneo.fr
—Master IHPS - Première Année —

Simulation de propagation de front par méthode itérative et différences finies de deuxième ordre

Le but de ce projet est de réaliser un solveur basique capable de donner les solutions de l'équation Eikonale simulant la propagation d'un front simple.

Equation Eikonale

L'étude de la propagation de front d'ondes intervient dans de nombreuses applications : imagerie médicale, geoscience, finance, aéronautique, etc. On peut par exemple reconsistuer une surface 3D à partir des variations de luminosité d'une image 2D, calculer les temps d'arrivée d'une onde sismique dans un environnement complexe à partir de plusieurs sources, réaliser des segmentations d'image IRM du cerveau.

Cette propagation peut être représentée sous forme mathématiques par l'équation Eikonale de la forme :

$$\begin{cases} c(x).|\nabla u(x)| & = 1, x \in \Omega \subset \mathbb{R}^n \\ u(x) & = 0, x \in \Gamma \end{cases} \tag{1}$$

u est la fonction solution à trouver, ∇u correspond au gradient de u (voir annexe A), c est une fonction de vitesse de propagation strictement positive, et Γ correspond au front d'évolution.

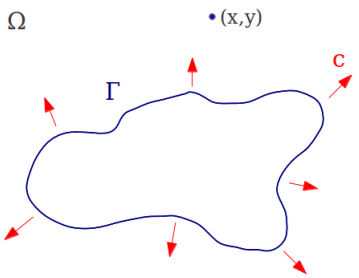

FIGURE 1 – Evolution de front de propagation

On veut trouver les solutions de la fonction u numériquement. Pour celà, on va utiliser des méthodes de discrétisation par différences finies.

Discrétisation

Par simplicité, la vélocité c est supposée constante et on travaillera en deux dimensions sur une grille cartésienne (fig. 2) T de $n \times n$ élements. Le maillage est structuré et régulier de pas (distance entre chaque noeud) constant h. Un noeud correspond aux intersections du maillage.

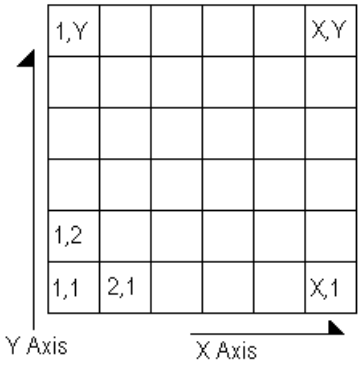

FIGURE 2 – Grille cartésienne

Afin de discrétiser l'équation 1, on a recours à un schéma dit décentré ("upwind scheme"). Cette méthode consiste à utiliser les différences finies pour simuler la direction de propagation du front. En notant $u_{i,j}$ la solution u au noeud de coordonnées (i,j), on définit les opérateurs de calcul $D_{i,j}$ suivant :

Schéma différence finie d'ordre 1

$$\begin{cases} D_{i,j}^{x+} = \frac{u_{i,j}-u_{i-1,j}}{h}, & D_{i,j}^{x-} = \frac{u_{i+1,j}-u_{i,j}}{h} \\ D_{i,j}^{y-} = \frac{u_{i,j}-u_{i,j-1}}{h}, & D_{i,j}^{y+} = \frac{u_{i,j+1}-u_{i,j}}{h} \end{cases} \tag{2}$$

Schéma différence finie d'ordre 2

$$\begin{cases} D_{i,j}^{x+} = \frac{3u_{i,j}-4u_{i-1,j}+u_{i-2,j}}{2h}, & D_{i,j}^{x-} = \frac{3u_{i,j}-4u_{i+1,j}+u_{i+2,j}}{2h} \\ D_{i,j}^{y-} = \frac{3u_{i,j}-4u_{i-1,j}+u_{i-2,j}}{2h}, & D_{i,j}^{y+} = \frac{3u_{i,j}-4u_{i+1,j}+u_{i+2,j}}{2h} \end{cases} \tag{3}$$

A partir de ces opérateurs, étant donné un point du maillage (i,j), la valeur $u_{i,j}$, qui est la solution locale recherchée de l'équation Eikonale, peut être calculée grâce à une méthode numérique de Godunov de premier ordre :

$$\max(D_{i,j}^{x-}, -D_{i,j}^{x+}, 0)^2 + \max(D_{i,j}^{y-}, -D_{i,j}^{y+}, 0)^2 = \frac{1}{c_{i,j}^2} \tag{4}$$

L'idée est de calculer la nouvelle valeur de $u_{i,j}$ à partir des noeuds voisins. En réitérant le processus plusieurs fois sur toute la grille, on est en mesure de prévoir l'évolution du front avec cette méthode.

Algorithme

Afin d'appliquer la méthode sur toute la grille on va utiliser un algorithme itératif type brute force :

Algorithme 1 : Algorithme Rouy-Tourin

Data : u^0
Result : u^∞
begin
  for *all node* (i,j) *in the grid* **do**
    $u_{i,j} = 0$ on Γ_0
    $u_{i,j} = \infty$ elsewhere
  while *convergence is not reached* **do**
    for *all node* (i,j) *in the grid* **do**
      Solve the Eikonal equation at node $u_{i,j}$

Une illustration du déroulement de l'algorithme est disponible sur la figure 3 ci-dessous. Les noeuds du front initial sont initialisés à 1. Les noeuds non calculés sont initialisés avec une valeur très grande. Les noeuds verts représentent les noeuds qui sont en train d'être calculés. Si le noeud ne change plus de valeur il a atteint la convergence (en rouge). La figure 4 montre une carte de distance obtenue par l'algorithme qu'on appelle carte de distance géodésique.

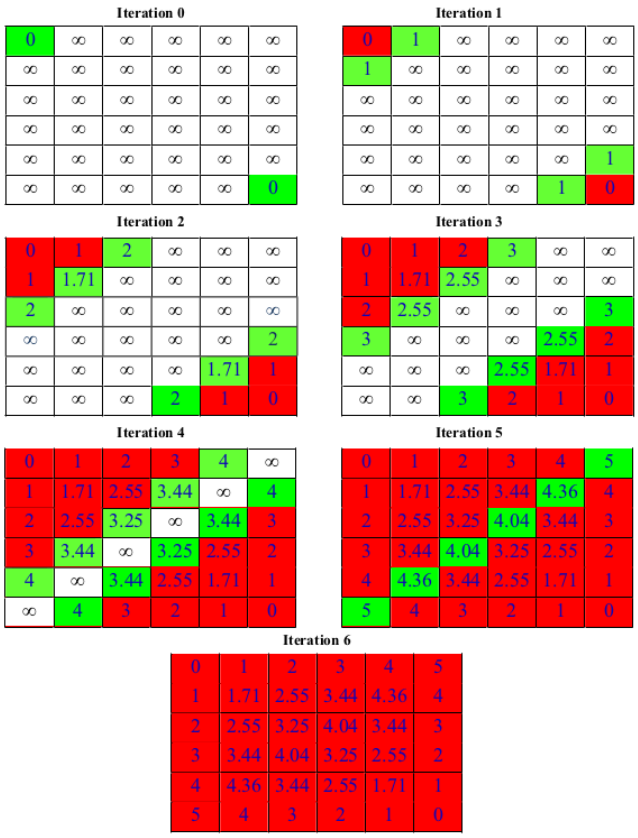

FIGURE 3 – Algorithme itératif Rouy-Tourin

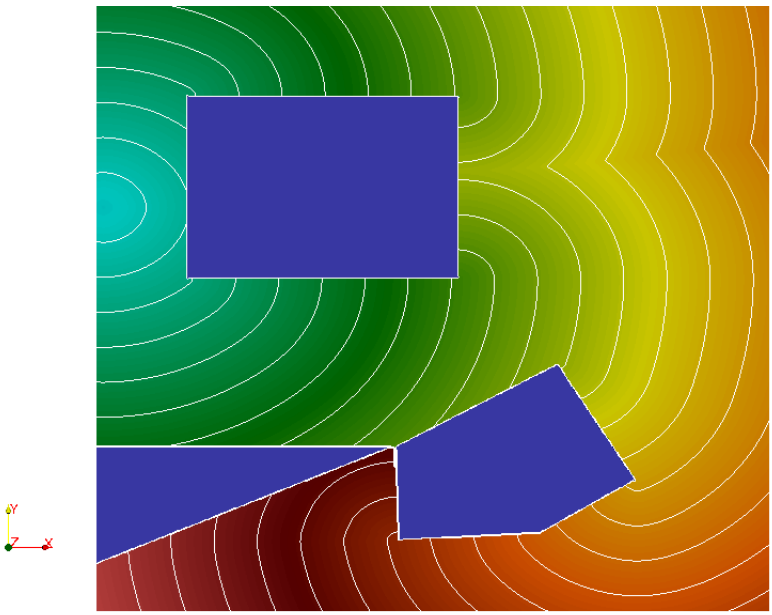

FIGURE 4 – Carte de distance géodésique avec trois obstacles polygones

Travail demandé

1. Trouver l'équation quadratique associée au schéma Rouy-Tourin. On pourra par commodité poser $u_{xmin} = \min(u_{i-1,j}, u_{i+1,j})$ et $u_{ymin} = \min(u_{i,j-1}, u_{i,j+1})$. Résoudre cette équation (voir annexe B) numériquement.
2. Prendre en main de la bibliothèque de visualisation scientifique VTK et de l'outil de visualisation Paraview (voir annexe D).
3. Coder l'algorithme itératif Rouy-Tourin dans un premier temps en utilisant des différences finies de première ordre puis deuxième ordre en le testant avec différents types de front initial (point, cercle, carré, segment...).
4. Calculer les erreurs L_∞ et L_1 pour chaque ordre de différence finie différent par rapport à une distance euclidienne simple.

Améliorations possibles

– Simuler avec des vélocités non constantes, des obstacles.

A Gradient

Le gradient d'une fonction $\vec{\nabla} u$ (ou $\overrightarrow{grad}\, u$) est un vecteur qui indique la direction et le sens de la fonction u dans l'espace. Dans notre cas, dans un repère orthonormé en deux dimensions $(O, \vec{e_x}, \vec{e_y})$, le gradient de u a pour expression :

$$\nabla u = \frac{\partial u}{\partial x}\vec{e_x} + \frac{\partial u}{\partial y}\vec{e_y}$$

B Résolution numérique d'une équation quadratique

Pour résoudre numériquement une équation du second degré $a.x^2 + b.x + c = 0$ avec $a; b; c$ rééls, on calcule dans un premier temps $q = -\frac{1}{2}\left[b + sgn(b).\sqrt{b^2\text{-}4ac}\right]$ avec $sgn(b)$ qui désigne le signe de b. Les deux solutions sont données par les valeurs $x_1 = \frac{q}{a}$ et $x_2 = \frac{c}{q}$.

C Evaluation des erreurs

En considérant x_i les valeurs justes et y_i les valeurs calculées, on définit les distances L_∞ et L_1 par :
$L_\infty = max(|x_1 - y_1|, ..., |x_n - y_n|)$ et $L_1 = \frac{\sum |x_i - y_i|}{n}$ avec n le nombre de sommets dans la grille.

D Exemple de fichier VTK

Ci-dessous se trouve un fichier simple au format VTK décrivant une simulation. On utilisera Paraview pour visionner cette dernière. N'hésitez pas à consulter les liens utiles pour avoir davantage de détails.

```
# vtk DataFile Version 3.6.2
Exemple FrontPropagation ASCII DATASET
RECTILINEAR GRID DIMENSIONS 51 51 1
X COORDINATES 51 float
-2 -1.92 -1.84 -1.76 -1.68 -1.6 -1.52 -1.44 -1.36 -1.28 -1.2 -1.12
-1.04 -0.96 -0.88 -0.8 -0.72 -0.64 -0.56 -0.48 -0.4 -0.32 -0.24
-0.16 -0.08 0 0.08 0.16 0.24 0.32 0.4 0.48 0.56 0.64 0.72 0.8 0.88
0.96 1.04 1.12 1.2 1.28 1.36 1.44 1.52 1.6 1.68 1.76 1.84 1.92 2
Y COORDINATES 51 float
-2 -1.92 -1.84 -1.76 -1.68 -1.6 -1.52 -1.44
-1.36 -1.28 -1.2 -1.12 -1.04 -0.96 -0.88 -0.8 -0.72 -0.64 -0.56
-0.48 -0.4 -0.32 -0.24 -0.16 -0.08 0 0.08 0.16 0.24 0.32 0.4 0.48
0.56 0.64 0.72 0.8 0.88 0.96 1.04 1.12 1.2 1.28 1.36 1.44 1.52 1.6
1.68 1.76 1.84 1.92 2
Z COORDINATES 1 float
0
POINT DATA 2601
SCALARS A int 1
LOOKUP TABLE default
0
```

E Références, liens utiles

Multistencils Fast Marching Methods : A Highly Accurate Solution to the Eikonal Equation on Cartesian Domains
http://www.cvip.uofl.edu/wwwcvip/research/publications/Pub_Pdf/2007_2/Sabry_Farag_TPAMI_2007.pdf
VTK file formats : http://www.vtk.org/VTK/img/file-formats.pdf
http://www.vtk.org/Wiki/ParaView/Users_Guide/VTK_Data_Model
http://www.idris.fr/ada/ada-paraview-doc.html