Rapport de Stage

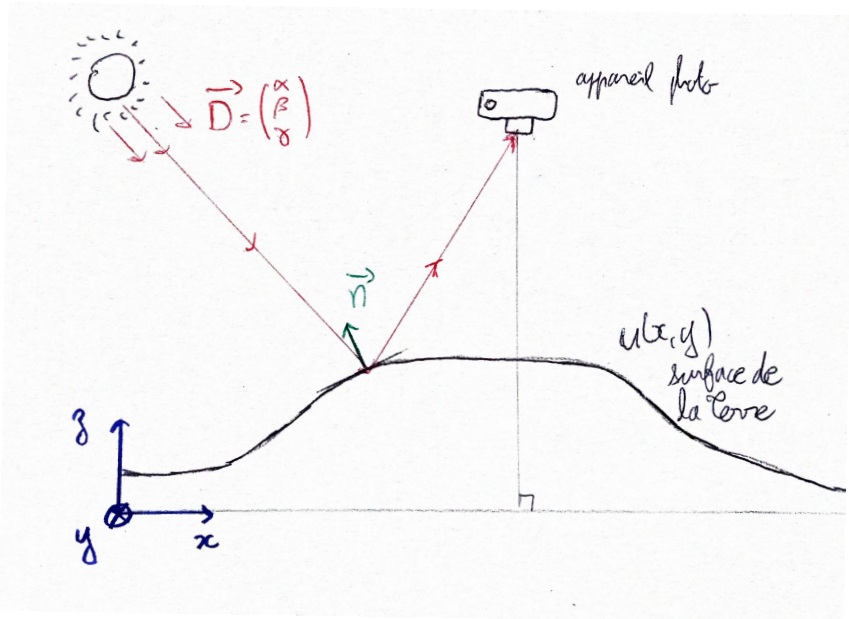
Vianney de la Salle & Romain Dugast

1 Introduction

Que ce soit pour la détermination d'un volume de tas de charbon, l'analyse du relief d'un astéroïde ou de l'environnement proche d'un robot, la reconstitution du relief d'une surface à partir d'une photo peut être un enjeu intéressant. (A poursuivre)

2 Modélisation

Problème à résoudre : Comment reconstruire le relief d'une surface à partir d'une photo ?
Le relief est mathématiquement modélisé par une fonction u donnant l'altitude aux points de coordonnées $(x, y) \in \Omega$ (où Ω est un ouvert de $\mathbb{R}^2$). La photo est quant à elle la donnée d'une fonction I attribuant l'intensité reçue par l'appareil photo pour des rayons lumineux provenant des points de la surface étudiée (les points $(x, y, u(x, y))$ avec $(x, y) \in \Omega$). Nous cherchons donc à déterminer u, connaissant I. Pour cela, nous avons besoin d'une relation mathématique entre ces deux fonctions, et celle-ci est donnée par le modèle physique de l'éclairage Lambertien comme le présente Horn dans son article de 1975 [1] :



Dans le repère orthonormée $(Oxyz)$, $\overrightarrow{D}$ est le vecteur de norme 1 qui donne la direction du soleil et $\overrightarrow{n}$ est le vecteur normal à la surface. A est l'albédo de la surface réfléchissante que l'on supposera uniforme, I_0 est l'intensité du soleil et B est la luminosité ambiante. Toutes ces données, à l'exception du vecteur normal $\overrightarrow{n}(x, y)$ sont supposées connues.
Le modèle physique donne la formule suivante : $I = AI_0\langle D, n\rangle + B$
On peut alors expliciter le vecteur $\overrightarrow{n}$ en fonction des dérivées directionnelles de u pour obtenir la relation entre I et u :

$$I(x, y) = A\frac{-\alpha u_x - \beta u_y + \gamma}{\sqrt{1 + u_x^2 + u_y^2}} + B \tag{1}$$

Du point de vue de notre problème (problème indirect), on appelle cette équation "équation de Shape From Shading" ou "équation SFS" puisqu'il s'agit d'une équation aux dérivées partielle dont la solution est le relief de la surface et qui est résolu à partir de l'intensité lumineuse I.

L'enjeu de ce stage est donc de résoudre cette équation non linéaire avec une méthode numérique bien particulière (Fast Marching). Nous commencerons par traiter ce problème avec des conditions aux bords de Dirichlet,

puis nous nous intéresserons aux conditions de bords mixtes (ce qui n'a pas encore été fait dans la littérature).

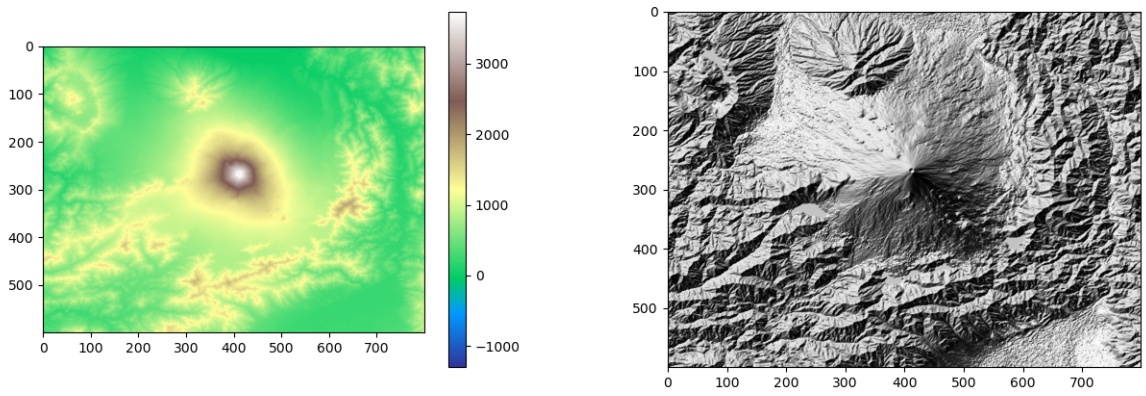
Les notations introduites dans cette partie seront réutilisées par la suite.

3 Prise en main du problème

Afin de se familiariser avec le sujet nous commençons par une tâche simple qui consiste à vérifier l'équation (1) en calculant une intensité I à partir d'un relief u connu. Il s'agit du problème direct, qui ne nécessite pas de résolution d'équation différentielle.

3.1 Résolution du problème direct

Nous disposons de la donnée d'un relief u sous forme de tableau *numpy* et nous voulons afficher l'intensité I correspondante (pour des paramètre α, β et γ choisis). Il s'agit ainsi de la simulation d'une photo du relief u éclairé par le soleil sous une direction $\overrightarrow{D} = (\alpha, \beta, \gamma)$. On va vérifier que l'on obtient bien une image ayant l'apparence d'une photographie.
On peut utiliser la discrétisation classique des dérivées directionnelles (à l'ordre 1) ou encore la fonction *gradient* du module *numpy* de *python* qui est plus précise. On obtient des images très similaires, mais l'image obtenue en utilisant la fonction *gradient* de *numpy* semble légèrement plus régulière.



Relief u du Mont-Fuji affichée avec une palette de couleur, et la simulation d'une photographie en utilisant la fonction *gradient* de *numpy*.

Le résultat étant convainquant (on a l'impression de voir une véritable photo), nous pouvons nous convaincre que la relation (1) est pertinente et nous allons donc par la suite tenter de l'utiliser pour reconstituer le relief d'une surface à partir d'une photo associée. Avant de mettre en place la méthode numérique permettant la résolution de l'équation SFS, nous commençons par nous intéresser à certains points théoriques sur lesquels nous pourrons nous appuyer lors de la mise en place du schéma.

4 Equations de Hamilton-Jacobi

On étudie ici certains résultats sur les équation de Hamilton-Jacobi qui serviront dans la résolution de l'équation Shape from Shading.

4.1 Introduction

Les équations de Hamilton-Jacobi avec conditions au bords nulles (conditions de type Dirichlet) sont de la forme :

$$(E) : \begin{cases} H(x, \nabla u(x)) = 0 \text{ sur } \Omega \\ u(x) = 0 \text{ sur } \partial\Omega \end{cases}$$

H est le Hamiltonien de l'équation, et Ω est un domaine borné ouvert de $\mathbb{R}^d$.

Ces équations ne possèdent en général pas de solutions $\mathcal{C}^1$.
Par exemple, l'équation sur $[-1,1]$ donnée par $|u'(x)| = 1$ avec des conditions aux bords nulles n'admet pas de solution $\mathcal{C}^1$, cela se voit directement avec le théorème de Rolle par exemple.

On définie maintenant les deux Hamiltoniens que nous allons principalement étudier dans la suite :

- Pour l'équation Shape from Shading (dimension 2), le Hamiltonien est tel que, pour x, y, p, q dans $\mathbb{R}$:

$$H(x,y,p,q) = \frac{I(x,y) - B}{A}(\sqrt{1+p^2+q^2}) + \alpha p + \beta q - \gamma$$

.

Dans la suite, on note par abus de notation "$I(x,y) := \frac{I(x,y) - B}{A}$" afin de simplifier. Ces termes seront bien considérés plus tard, et cette simplification ne change rien aux résultats démontrés.

- On considère le cas précédent pour un éclairage vertical, c'est-à-dire $\alpha = \beta = 0$. L'équation Shape from Shading devient, en notant $p = \langle \nabla u, e_1 \rangle$ et $q = \langle \nabla u, e_2 \rangle$, avec e_1 et e_2 les vecteur de la base cannonique de $\mathbb{R}^2$:

$$\frac{I(x,y)}{\gamma}\sqrt{1+p^2+q^2} = 1$$

Donc :

$$\sqrt{p^2+q^2} = \sqrt{\frac{\gamma^2}{I(x,y)^2} - 1}$$

Autrement dit :

$$\|\nabla u\| = \eta(x)$$

avec η une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$.
On remarque que le calcul précédent est possible si I est ne s'annule pas, ce qui est le cas pour un éclairage vertical lorsque le relief n'est pas discontinu, ce qui est raisonnable de supposer.
Cette équation correspond à l'équation dite Eikonale.

Dans la suite, on commencera par traiter cette équation en considérant le cas $\eta(x) = 1$ afin de simplifier. On considère ainsi l'équation, en dimension n :

$$(E') : \begin{cases} \|\nabla u\| = 1 \text{ sur } \Omega \\ u(x) = 0 \text{ sur } \partial\Omega \end{cases}$$

Cette équation n'a pas de solution forte comme montré précédemment, et a fortiori l'équation shape from shading non plus. Il est donc nécessaire de définir une notion de solution faible pouvant nous servir dans la suite.

4.2 Caractéristiques

4.3 Solutions de Viscosité

4.3.1 Définition

Afin de définir une notion de solution faible, on utilise la notion de solution de viscosité introduite par Crandall et Lions en 1982. Celle-ci se base sur la méthode dite de *viscosité évanescente*, qui consiste à transformer l'équation en lui rajoutant un terme d'ordre 2, on résout alors :

$$\epsilon \Delta u + H(x, \nabla u) = 0$$

On peut alors montrer que sous certaines conditions sur le domaine et le Hamiltonien, la suite de solutions $\mathcal{C}^1$: $(u_\epsilon)_\epsilon$ ainsi obtenue admet une suite extraite convergente (en utilisant le théorème d'Arzela-Ascoli) au sens de la convergence uniforme.

La solution limite ainsi trouvée est uniformément continue & bornée (1), et vérifie les conditions aux bords (2). De plus, elle vérifie les conditions suivantes qui vont constituer la définition de solution de viscosité :
Pour toute fonction v de classe $\mathcal{C}^\infty$ sur Ω, et pour tout x_0 dans Ω : (3)

$$\begin{cases} (u-v)(x_0) \text{ est un maximum local} \Rightarrow H(x_0, \nabla v(x_0)) \leq 0 \\ (u-v)(x_0) \text{ est un minimum local} \Rightarrow H(x_0, \nabla v(x_0)) \geq 0 \end{cases}$$

Une fonction vérifiant les conditions (1), (2) et (3) sur Ω est alors appelée *solution de viscosité* (on ne s'intéresse ici qu'aux solutions de viscosité continues).

4.3.2 Propriétés

On commence par montrer certaines propriétés des solutions de viscosité qui confirment leur pertinence.

Propriété : Si u est solution $\mathcal{C}^1$ de (E), alors u est solution de viscosité.

Démonstration : Si u est une solution $\mathcal{C}^1$ de l'équation (E), alors elle vérifie directement les conditions aux bords, et est uniformément continue car continue sur un domaine compact (Heine). Donnons-nous v une fonction $\mathcal{C}^\infty$ sur Ω.
On suppose que $u - v$ atteint un extremum en x_0. Alors on a nécessairement $\nabla u(x_0) = \nabla v(x_0)$. On a donc par suite :

$$H(x_0, \nabla v(x_0)) = H(x_0, \nabla u(x_0)) = 0$$

❑

Propriété : Si u est solution de viscosité de (E) et est différentiable en x_0 alors : $H(x_0, \nabla u(x_0) = 0$

Démonstration :

- On commence par montrer qu'il existe v de classe $\mathcal{C}^1$ tel que $u(x_0) = v(x_0)$ et $u - v$ a un maximum local en x_0. On montre ce résultat dans le cas $x_0 = u(x_0) = \nabla u(x_0) = 0$, puisque s'il est vrai dans ce cas il l'est aussi dans le cas général.
  Par ces hypothèses, et puisque u est supposée être différentiable en 0, on en déduit que la fonction ρ_1 définie par $\rho_1(x) := \dfrac{u(x)}{\|x\|}$ est bien définie et continue sur Ω. De plus, on défini sur $\mathbb{R}_+$ la fonction : $\rho_2 : x \mapsto \max\limits_{x\in\Omega} |\rho_1(x)|$.
  Enfin, on pose

$$v : x \mapsto \int_{|x|}^{2|x|} \rho_2(r)dr + |x|^2$$

  On a $v(0) = 0$. On constate que v vérifie bien les conditions souhaitées :

$$u(x) - v(x) = |x|\rho_1(x) - \int_{|x|}^{2|x|} \rho_2(r)dr - |x|^2$$

$$\leq -|x|^2 \leq 0 = u(0) - v(0)$$

  par définition de ρ_2, pour x dans un voisinage de 0 contenu dans Ω. Enfin, v est bien $\mathcal{C}^1$ sur Ω en dehors de 0. En 0, on a l'inégalité : $|v(x)| \leq \|x\|\rho_2(2|x|) + \|x\|^2$ donc v est bien différentiable en 0 de gradient nul. De l'égalité : $\nabla v(x) = \dfrac{2x}{\|x\|}\rho_2(\|x\|) - \dfrac{x}{\|x\|}\rho_2(\|x\|) + 2x$, on déduit que v est bien $\mathcal{C}^1$ sur Ω.

- On considère une fonction v comme explicitée précédemment, pour un point x_0 où u est différentiable. On a donc $u - v$ qui admet un maximum local en x_0. On considère une suite régularisante $(\eta_\epsilon)_\epsilon$ telle que $(\eta_\epsilon * v)_\epsilon$ est une suite de fonctions $\mathcal{C}^\infty$ qui convergent uniformément vers v (dans un voisinage $B(x_0, r)$ de x_0), et dont les gradients convergent aussi uniformément vers ∇v.
  On dispose alors d'une suite (x_ϵ) qui converge vers x_0 telle que $u - v_\epsilon$ ait un maximum en x_ϵ $(*)$. Puisque u est solution de viscosité de (E), on en déduit immédiatement : pour tout $\epsilon \leq r, H(x_\epsilon, \nabla v(x_\epsilon)) \leq 0$. Par passage à la limite : $H(\nabla v(x_0))$. Or, $u - v$ admet un maximum en x_0 donc : $\nabla u(x_0) = \nabla v(x_0)$ donc $H(x_0, \nabla u(x_0)) \leq 0$.
  On applique le même raisonnement à $-u$, pour trouver une fonction $\tilde{v}$ telle que $u - \tilde{v}$ admette un minimum en x_0. On obtient alors de la même façon : $H(x_0, \nabla u(x_0)) \geq 0$ et donc finalement : $H(x_0, \nabla u(x_0)) = 0$.

- $(*)$ En effet, pour $r > 0$ suffisamment petit, on a : $\max\limits_{\partial B(x_0,r)} (u - v)(x) \leq (u - v)(x_0)$. Puisque $(u - v_\epsilon)$ converge uniformément vers $u - v$, pour ϵ suffisamment petit on a : $\max_{\partial B}(u - v_\epsilon)(x) \leq (u - v_\epsilon)(x_0)$ et donc par théorème de Weierstrass on dispose de x_ϵ dans $\mathring{B}(x_0, r)$ tel que $u - v_\epsilon$ atteigne son maximum en x_ϵ. On trouve ainsi bien une suite comme voulu.

❑

Propriété : La fonction distance aux bords $x \mapsto d(x, \partial\Omega)$ est solution de viscosité de (E')

Démonstration : La fonction distance euclidienne aux bords du domaine vérifie directement les conditions aux bords nulles. De plus, elle est bien continue sur un domaine fermé borné donc uniformément continue par théorème de Heine. Enfin, si la fonction distance est différentiable en $x \in \Omega$, alors
On vérifie ensuite le point important de la définition de solution de viscosité : on se donne v une fonction de classe $\mathcal{C}^\infty$ sur Ω.

- On suppose d'abord que $u - v$ admet un maximum en x_0 (avec u la fonction distance). On suppose par l'absurde que $\|\nabla v(x_0)\| > 1$. On prend h colinéaire à $\nabla v(x_0)$, et on a alors :

$$v(x_0 + h) = v(x_0) + \langle \nabla v(x_0), h \rangle + o(h)$$

$$= v(x_0) + \|h\|\|\nabla v(x_0)\| + o(h)$$

  Par définition de la fonction u, on a aussi : $u(x_0 + h) \leq u(x_0) + \|h\|$. On en déduit donc :

$$(u - v)(x_0 + h) \leq (u - v)(x_0) + \|h\|(1 - \|\nabla v(x_0)\|) + o(h)$$

$$\leq (u - v)(x_0) \text{ par hypothèse, et pour } h \text{ assez petit}$$

  C'est donc absurde puisque $u - v$ est supposée avoir un maximum en x_0.

- On suppose maintenant que $u - v$ admet un minimum en x_0, et par l'absurde on suppose que $\|\nabla v(x_0)\| < 1$. Puisque $\partial\Omega$ est fermé, il existe y dans $\partial\Omega$ tel que $u(x_0) = \|x_0 - y\|$. On note g la fonction définie sur Ω par $x \mapsto \|x - y\|$. g est différentiable sur Ω et est telle que : $\forall x \in \Omega, \|\nabla g(x)\| = 1$. On note $e = -\nabla g(x_0)$, et on prend h dans $\mathbb{R}$.
  On a alors :

$$(g - v)(x_0 + he) = (g - v)(x_0) - h + h\langle e, \nabla v(x_0)\rangle + o(h)$$

$$\leq (g - v)(x_0) + h(\|\nabla v(x_0)\| - 1) + o(h) \text{ par inégalité de Cauchy-Schwarz}$$

  Par hypothèse, on en déduit : $(g-v)(x_0+he) \leq (g-v)(x_0)$ pour h assez petit. Par définition de u et de g, on a : $\forall x \in \Omega, u(x) \leq g(x)$. On en déduit : $u(x_0+he) \leq g(x_0+he) \leq g(x_0) = u(x_0)$ ce qui est absurde par hypothèse.

On en déduit que $x \mapsto d(x, \partial\Omega)$ est solution de viscosité de (E') sur Ω.

❑

4.3.3 Existence & Unicité

On énonce à présent deux théorèmes, un d'existence et un d'unicité concernant les solutions d'une équation de Hamilton-Jacobi comme présentée précedemment.
On suppose ici plusieurs hypothèses sur le Hamiltonien H :

- H est K-Lipschitzien en x
- H est convexe en p
- Il existe $\underline{u}$ sous-solution de viscosité stricte de (E) de classe $\mathcal{C}^1$, i.e. $\underline{u}$ vérifie : $H(x, \nabla\underline{u}) < 0 sur \Omega$.

Théorème : Sous ces hypothèses, on dispose de l'unicité pour les solutions de viscosité de (E)

Démonstration :

On suppose données u et v deux solutions de viscosité de (E).

- On pose $u_t = tu+(1-t)\underline{u}$. Par hypothèse, on dispose de f strictement négative sur $\overline{\Omega}$ telle que $H(x, \nabla \underline{u}) < f(x)$ sur Ω. Quitte à remplacer $\underline{u}$ par $\underline{u} - M$ qui vérifie les même hypothèses, on peut supposer : $\underline{u} \leq u$ sur Ω. On a alors : $u_t \leq u$, $u_t -> u$ quand t tend vers 1, et u_t continue $\forall t \in]0,1[$.
  On se donne maintenant ϕ dans $\mathcal{C}^\infty(\Omega)$ et x_0 tels que $(u_t - \phi)(x_0)$ soit un maximum local de $u_t - \phi$. On note $p = \nabla\phi(x_0)$.
  On pose $q = p - \frac{1-t}{t}\nabla\underline{u}$. On vérifie que $H(x_0, q) \leq 0(1)$. On obtient alors :

$$H(x_0, p) \leq tH(x_0, q) + (1-t)H(x_0, \nabla\underline{u}(x_0)) \leq (1-t)f(x)$$

  par hypothèses, en utilisant la convexité du Hamiltonien.

- On raisonne à t fixé. On a $u_t \leq v$ sur $\partial\Omega$. On suppose que $M = sup(u_t - v) > 0$ sur Ω. On pose alors : $\psi_\epsilon(x, y) = u_t(x) - v(y) - \frac{\|x-y\|^2}{\epsilon^2}$ définie sur $\Omega \times \Omega$. On considère la maximum de ψ_ϵ sur $\overline{\Omega}^2$ qui est compact, et on note (x_ϵ, y_ϵ) et M_ϵ un point de maximum et la valeur atteinte en ce maximum. On commence par montrer le Lemme suivant :

  Lemme : Lorsque ϵ tend vers 0 :

  1. $M_\epsilon \to M$
  2. $\frac{\|x_\epsilon - y_\epsilon\|^2}{\epsilon^2} \to 0$
  3. $u_t(x_\epsilon) - v(y_\epsilon) \to M$
  4. $x_\epsilon, y_\epsilon \in \Omega$ (pour ϵ assez petit)

  Pour démontrer cela, on commence par remarquer que pour $x \in \Omega$, $\psi_\epsilon(x, x) \leq M_\epsilon$ donc $u_t(x) - v(x) \leq M_\epsilon$, et $M \leq M_\epsilon$.
  On a donc : $M \leq \psi_\epsilon(x_\epsilon, y_\epsilon) \leq 2R - \frac{\|x_\epsilon - y_\epsilon\|^2}{\epsilon^2}$ avec u_t et v bornées par R. Puisque $M > 0$, on obtient $\frac{\|x_\epsilon - y_\epsilon\|^2}{\epsilon^2} \leq 2R$, et finalement : $\|x_\epsilon - y_\epsilon\| \leq \sqrt{2R}\epsilon$.
  On pose maintenant $m_v : t \mapsto \sup_{|x-y| \leq t} |v(x) - v(y)|$. Puisque v est uniformément continue, on remarque que $\lim_{t\to 0} m_v(t) = 0$.

  On a alors : $M \leq u_t(x_\epsilon) - v(y_\epsilon) + m_v(\|x_\epsilon - y_\epsilon\|) \leq u_t(x_\epsilon) - v(x_\epsilon) + m_v(\|x_\epsilon - y_\epsilon\|) \leq M + m_v(\|x_\epsilon - y_\epsilon\|)$. On en déduit :

$$M \leq M_\epsilon \leq u_t(x_\epsilon) - v(y_\epsilon) \leq M + m_v(\sqrt{2R}\epsilon)$$

  . Cela démontre 1), 2) et 3). Enfin, si on dispose d'une suite (ϵ_n) telle que $x_{\epsilon_n} \in \partial\Omega$, alors : $u_t(x_{\epsilon_n}) - v(y_{\epsilon_n}) \leq u_t(x_{\epsilon_n}) - v(x_{\epsilon_n}) + \mu \leq \mu$ pour n assez grand puisque $u_t \leq v$ sur $\partial\Omega$. Par passage à la limite, on aurait alors $M \leq 0$ ce qui est absurde. On raisonne de même sur y_ϵ, ce qui permet d'en déduire (4).

- On remarque que x_ϵ est un point de maximum de la fonction $x \mapsto u_t(x) - (v(y_\epsilon) + \frac{\|x - y_\epsilon\|^2}{\epsilon^2})$. Par propriété de u_t, on en déduit : $H(x_\epsilon, \frac{2(x_\epsilon - y_\epsilon)}{\epsilon^2}) \leq (1-t)f(x_\epsilon)$. De même, y_ϵ est un point de minimum de $y \mapsto v(y) - (u(x_\epsilon) - \frac{\|x_\epsilon - y\|^2}{\epsilon^2})$. Par propriété sur v, on en déduit : $H(y_\epsilon, \frac{2(x_\epsilon - y\epsilon)}{\epsilon^2} \geq 0$. En combinant les deux inégalités, et par Lipschitziannité du Hamiltonien, on obtient :

$$(t-1)f(x_\epsilon) \leq H(y_\epsilon, \frac{2(x_\epsilon - y_\epsilon)}{\epsilon^2} - H(x_\epsilon, \frac{2(x_\epsilon - y_\epsilon)}{\epsilon^2} \leq K\|x_\epsilon - y_\epsilon\|$$

  . Or , d'après le lemme, c'est absurde puisque $-f$ est stricement positive. On en déduit que $M \leq 0$.

- Enfin, puisqu'on a $u_t \leq v$ sur Ω pour tout t strictement inférieur à 1, par passage à la limite on obtient $u \leq v$. Or, on peut répéter la même démonstration en inversant les rôles de u et v qui vérifient les mêmes propriétés : on en déduit le théorème voulu : $u = v$ sur Ω.

(1) En effet, on a $u - (\frac{1}{t}\phi - \frac{1-t}{t}\overline{u})$ qui admet un maximum en x_0 par hypothèse. Or, $\frac{1}{t}\phi - \frac{1-t}{t}\overline{u}$ est une fonction de classe $\mathcal{C}^1$ sur Ω. A la façon de la propriété 2 de la partie précédente, on peut l'approcher par une suite de fonctions lisses ϕ_ϵ (à l'aide d'une suite régularisante par exemple). De la même façon, on construit alors une suite de points x_ϵ de maximums des fonctions $u - \phi_\epsilon$, et par passage à la limite on obtient bien : $H(x_0, q) \leq 0$.

❑

Théorème : Sous les mêmes hypothèses, on dispose de l'existence d'une (unique) solution de viscosité de (E)

On ne donne pas ici les démonstrations de ce théorème dans le cas général (cf [] et []) qui nécessitent des connaissances dont nous ne disposons pas.
Nous remarquons tout de même qu'il existe 2 façon principales d'effectuer cette démonstration :

- La première méthode est de transformer l'équation en $\epsilon\Delta u + H(x, \nabla u) = 0$. On peut alors montrer que cette équation possède une solution u_ϵ. On peut ensuite sous certaines hypothèses extraire de la suite (u_ϵ) une sous suite convergente lorsque ϵ tend vers 0, à l'aide du théorème d'Arzela-Ascoli. Il est enfin possible de montrer que cette limite est bien solution de viscosité de (E).
- La seconde méthode qui semble être plus utilisée en pratique consiste à utiliser la théorie du contrôle afin de transformer l'équation en un problème de contrôle optimal, à l'aide de propriétés des équations de Hamilton-Jacobi. On montre que la solution de ce problème est alors une solution de viscosité.

4.3.4 Application à l'équation Eikonale

On a montré que dans le cas d'un domaine ouvert borné Ω de $\mathbb{R}^n$, et avec des conditions aux bords de Dirichlet nulles, la fonction distance au bord du domaine était solution de viscosité de l'équation Eikonale. On vérifie maintenant les hypothèses d'unicité sur le Hamiltonien :

$$H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

$$(x, p) \mapsto \|p\| - 1$$

- H est Lipschitzien car la norme l'est.
- H est convexe pour la même raison.
- La fonction nulle est une sous-solution de viscosité $\mathcal{C}^1$ stricte de l'équation.

On en déduit que la fonction distance est l'unique solution de viscosité de l'équation Eikonale.

4.3.5 Application à l'équation Shape from Shading

Le Hamiltonien est ici définie par :

$$H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$
$$(x, p) \mapsto I(x, y)\sqrt{1 + \|p\|^2} - \langle(\alpha, \beta), p\rangle - \gamma$$

- On remarque d'abord que H est Lipschitzien si et seulement si I l'est. cela présuppose que le relief ne possède pas de "discontinuité de pente" d'après l'équation SFS. On supposera dans la suite que cette condition forte est vérifiée sur le relief recherché, et que I est lipischitzienne.
- La fonction $x \mapsto \sqrt{1 + x^2}$ étant convexe sur $\mathbb{R}$, la norme l'étant aussi ainsi que le produit scalaire, on en déduit que H est bien convexe en p.
- Enfin, sous la condition $I(x) < 1 \forall x$ et un éclairage non horizontal, on vérifie que la fonction $\underline{u}(x, y) \mapsto \frac{\alpha}{\gamma}x + \frac{\beta}{\gamma}y$ est $\mathcal{C}^1$ et sous-solution stricte de viscosité : $H(x, \nabla\underline{u}(x)) = I(x)\sqrt{1 + \frac{\alpha^2+\beta^2}{\gamma^2}} - \frac{\|(\alpha,\beta,\gamma)\|^2}{\gamma^2} = \frac{I(x)-1}{\gamma^2} < 0$ puisque (α, β, γ) est normé. Cette condition sur I revient à demander à ce que en aucun point du relief l'éclairage n'ait une incidence normale. On peut remarquer que si c'est le cas, il y a effectivement plusieurs solutions de viscosité au problème ce qui n'est alors pas satisfaisant. On suppose dans la suite que cette hypothèse est vérifiée, ce qui n'est pas déraisonnable puisque l'intensité manipulée numériquement n'est presque jamais exactement égale à 1.

Dans la suite, on va maintenant chercher à résoudre cette équation SFS numériquement en construisant un schéma convergent vers la solution de viscosité du problème, en se basant sur le fait que la solution de l'équation Eikonale, donc de SFS avec un éclairage vertical, ait pour solution la fonction de distance aux bords.

5 Résolution d'une équation différentielle par FMM

Dans cette partie, nous nous intéressons à l'algorithme de fast-marching (fast marching method ou FMM) qui est une méthode de parcours de graphe. On peut en effet discrétiser le domaine Ω en un maillage X et calculer de proche en proche les approximations de la fonction u aux points de ce maillage.
Etant donnée une discrétisation de l'équation considérée, le problème est donc de savoir dans quel ordre sélectionner les points sur lesquels on calcul une solution approchée. On a vu précédemment que la solution de l'équation eikonale était la distance au bord, on peut donc voir ce problème comme celui d'un plus court chemin. Une façon intelligente de calculer numériquement cette solution est alors de parcourir le maillage à la manière de l'algorithme de Dijkstra, qui est utilisé pour le calcul d'un plus court chemin dans un graphe pondéré. C'est sur ce principe que repose la FMM, qui consiste à calculer les valeurs approchées de la solution selon un "front d'onde", qui se propage des valeurs les plus faibles vers les valeurs les plus élevées de la solution.
Après avoir décrit le principe de fonctionnement du Fast-Marching, nous commencerons par présenter les résultats de son implémentation pour l'équation Eikonale, puis pour l'équation SFS.

5.1 Description de l'algorithme

On distingue trois types d'états pour les points du maillage sur lequel l'équation est à résoudre : Accepted, Trial et Far. Initialement, tous les points du maillage sont classés "Far", à l'exception de ceux du bord du domaine dont on connait la valeur de u avec les conditions aux bords de l'équation à résoudre. Ces derniers sont classés "Trial". Les points "Accepted" sont ceux sur lesquels la valeur de u a été définitivement calculée. Les points "Trial" sont ceux qui ont déjà été considérés et pour lesquels une valeur de u a déjà été calculée. Les points "Far" sont ceux qui n'ont jamais été explorés et auxquels est associée une valeur infinie de u.
Ci-dessous le pseudo-code de l'algorithme :

Algorithm 1: Fast Marching

```
1 Initialisation (Accepted, Trial, Far);
2 while Trial ≠ {} do
3 |   Trouver l'élément Trial x qui minimise u;
4 |   Ajouter x à Accepted;
5 |   for y ∈ V(x) do
6 |   |   Ajouter y dans Trial;
7 |   |   Actualiser la valeur de u(y)
8 |   end
9 end
```

On précise que la notation $\mathcal{V}(x)$ désigne l'ensemble des voisins de x dans le maillage.
On note X l'ensemble des points du maillages et ∂X l'ensemble des points aux bords du domaine discret.

Actualisation des valeurs de u (ligne 7 dans le pseudo-code)
A chaque étape, un point x minimisant la valeur approchée de u parmi les points de "Trial" est donc considéré et accepté. On actualise alors ses voisins de la façon suivante :

Soit y un voisin de x. Si y est "accepted" ou si $y \in \partial X$ alors on ne modifie pas sa valeur. (on respecte la condition au bord imposée par le problème). Sinon, on calcule la valeur de $u(y)$ en résolvant l'équation associée au problème à résoudre.
Prenons le cas de l'équation eikonale : l'équation à résoudre est $\|\nabla u(x)\|^2 = 1$, $\forall x \in \Omega$. Si e_1 et e_2 désignent les vecteurs de la base canonique de $\mathbb{R}^2$, alors une façon classique de discrétiser la dérivée selon x est de remplacer $(\partial_1 u(x))^2$ par $\frac{1}{h}\max\left(0, u(x) - u(x - he_1), u(x) - u(x + he_1)\right)^2$. On obtient alors le schéma suivant :

$$\frac{1}{h^2}\sum_{i=1}^{2} \max(0, U(x) - U(x - he_i), U(x) - U(x + hei))^2 = 1$$

On remarque que dans le cas où le point est un minimum dans une direction, la discrétisation de la dérivée dans cette direction est nulle, tandis que ce schéma privilégie toujours les directions de plus fortes pentes dans les autres cas. Cette discrétisation est particulièrement adaptée lorsque les dérivées directionnelles sont élevées au carré, ce qui est le cas ici.

Pour appliquer ce schéma dans l'algorithme de fast-Marching, il suffit de choisir le voisin dont la valeur est la plus faible car il va maximiser la quantité $U(x) - U(x \pm he_i)$. Mais si dans une direction les deux voisins sont encore "Far", alors on a $U(x) - U(x - he_i) = U(x) - U(x + he_i) = -\infty$ et dans ce cas on doit ignorer le terme qui discrétise la dérivée dans cette direction. Par exemple si dans la direction e_1 les voisins de x sont encore "Far" et qu'on a $z = \min(U(x - he_1), U(x + he_2))$, on cherche à résoudre l'équation $(\lambda - z)^2 = h^2$.

Remarque : Par principe du Fast Marching et du plus court chemin, si dans une direction les deux voisins de x sont "Far", alors $U(x)$ est un minimum dans cette direction (les points étant traités par valeur de U croissantes). Ainsi, ignorer la dérivée dans cette direction revient à la considérée comme nulle, ce qui se comprend bien.

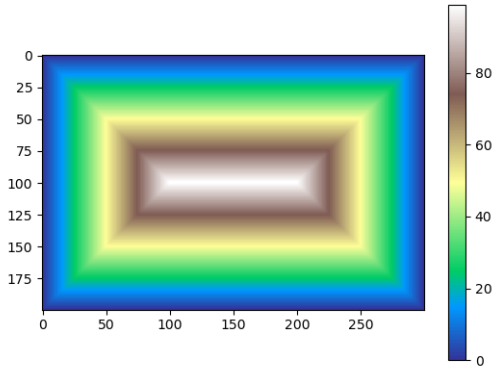
La valeur approchée de u en les voisins de x est donc calculé en résolvant une équation du second degré en $u(y)$.

À chaque équation quadratique à résoudre, on choisit la racine la plus grande car c'est ce qui assure d'obtenir la solution de viscosité (qui est la fonction distance au bord).

On obtient bien ainsi une méthode numérique permettant le calcul de solutions approchées à nos problèmes en estimant la valeur des points de proche en proche. On peut montrer que, dans le cas de l'équation Eikonale, ce schéma est bien convergent, stable et monotone, et donc qu'il est convergent. Au vu de la discrétisation choisie, on peut s'attendre à un ordre 1, cela sera confirmé plus tard.

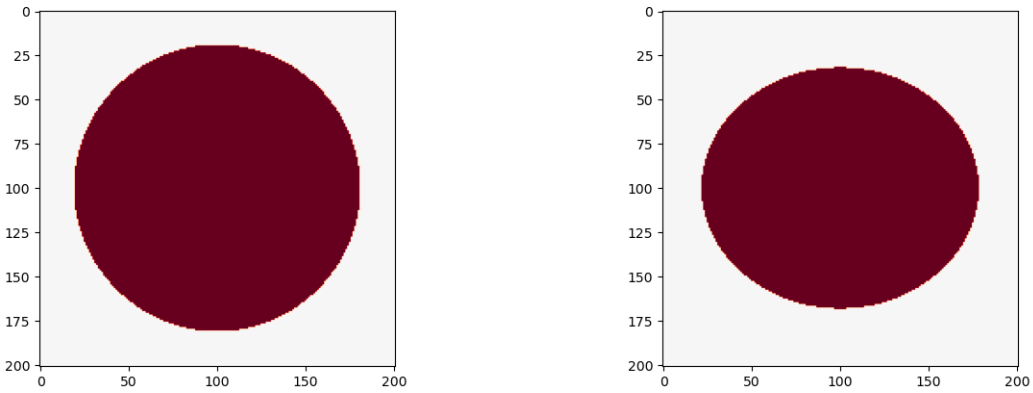
5.2 Implémentation et résultats pour l'équation eikonale

Nous avons implémenté le code correspondant (cf Annexe) et nous l'avons testé pour divers domaines Ω et des conditions de Dirichlet nulle au bords. Voici le résultat pour Ω un rectangle, et X le maillage rectangulaire de taille 300x500 :

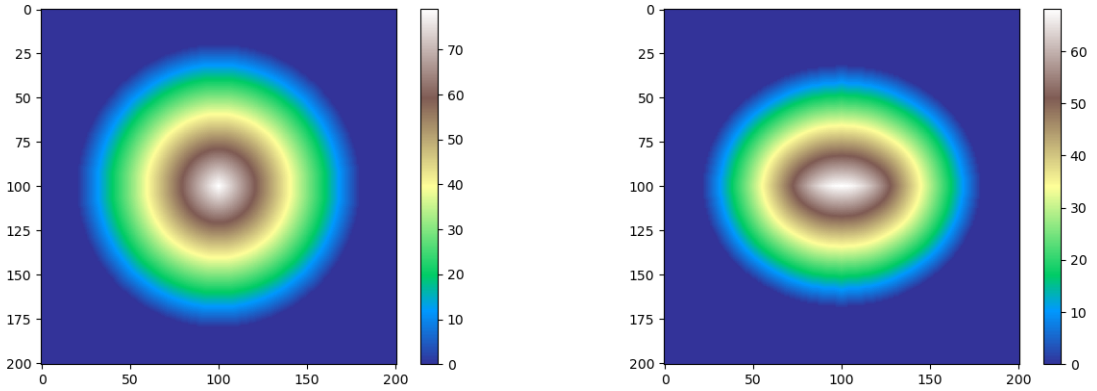


Le résultat correspond bien aux attentes théoriques : nous avons établi que les courbes caractéristiques pour ce problème étaient les lignes orthogonales aux bords du rectangle (jusqu'aux croisement), et que le long de ces courbes $(x(t), y(t))$ la dérivée de la fonction $u((x(t), y(t))$ valait 1. On vérifie bien par exemple que la valeur de u augmente de 1 à chaque pixel quand on part du bas du rectangle vers le haut (puis qu'elle redescend de 1 à chaque pixel). En fait, on reconnait bien dans ce résultat la fonction distance aux bords qui était attendue.

Nous avons alors testé l'algorithme sur différents domaines X, par exemple un disque et une ellipse :

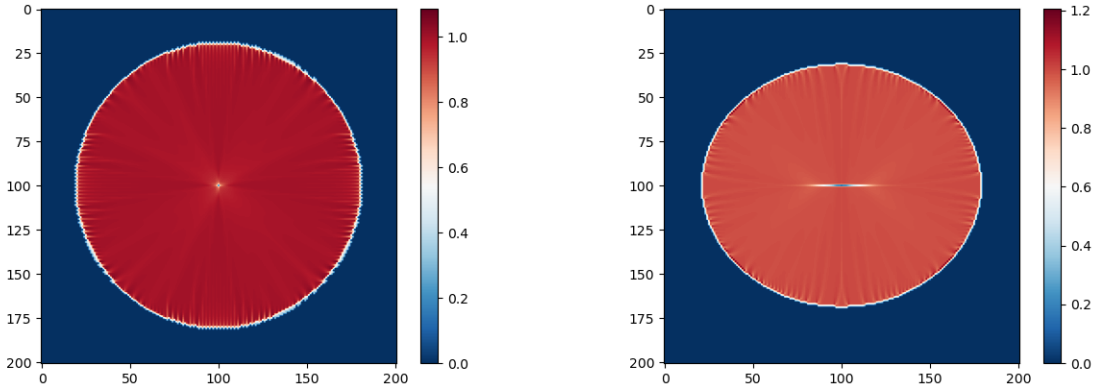


Domaines X de résolution de l'équation eikonale (en rouge).



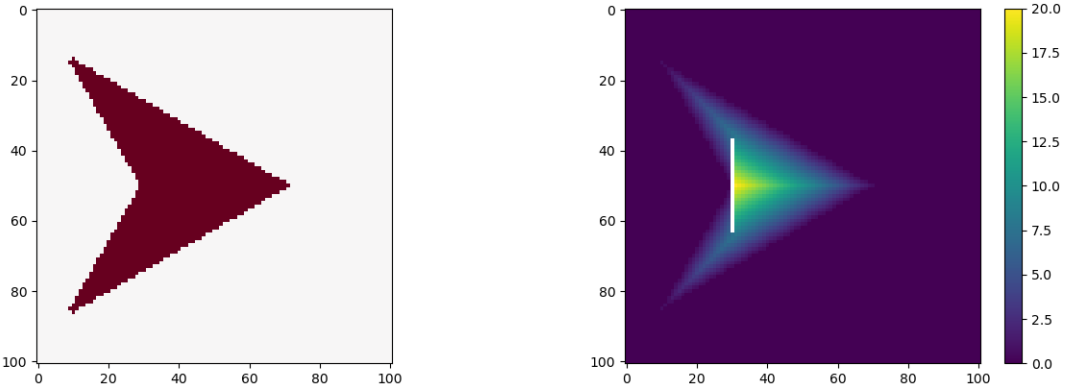
Résultat de la résolution.

On vérifie que les reliefs obtenus sont bien solutions de l'équation eikonale en affichant la norme du gradient des reliefs :



Norme des gradients de chaque relief.

Les valeurs de la norme du gradient de u ne sont pas rigoureusement égales à 1 mais sont bien très proches dans l'intérieur du domaine $X \setminus \partial X$.
Ainsi, dans le cas de formes régulières, l'algorithme fournit bien les résultats voulus. Nous avons aussi testé sur des formes plus irrégulières, comme par exemple ce quadrilatère concave :



Norme des gradients de chaque relief.

Sur l'image de gauche, on précise le domaine X de résolution de l'équation eikonale, qui prend la forme d'un deltoïde. Á droite, la résolution obtenue avec le défaut algorithmique. Les deux pixels blancs aux extrémités de la bande blanche n'ont pas pu être calculés car l'équation quadratique associée à ces points possédait un discriminant négatif. Ils ont pris alors la valeur "nan" de numpy et cette valeur s'est propagée sur la bande blanche. Pour éviter ce problème, nous avons choisi de prendre la partie réelle des racines du polynôme du second degré lorsque ce cas

arrivait, en espérant que les parties imaginaires soient petites pour que cela ne soit pas la source de trop grosses erreurs. Cela était la plupart du temps le cas, mais pouvait néanmoins causer des soucis lors de la résolution. Nous revenons plus tard sur une amélioration de cette partie de la méthode. Le résultat de cette modification pour le cas de la forme du deltoïde est affiché en première figure dans la section **6.3**.

6 Application de la FMM à l'équation SFS

Nous avons vu comment implémenter la FMM pour la résolution de l'équation eikonale simple, nous pouvons à présent la généraliser pour l'équation SFS à éclairage vertical. L'algorithme reste exactement le même dans sa structure mais puisque l'équation différentielle à résoudre est différentes, les équations quadratiques à résoudre à chaque itérations seront différentes.

6.1 Équation quadratique de SFS

Rappelons d'abord l'équation de Shape From Shading (1) :

$$I(x,y) = A\frac{-\alpha u_x - \beta u_y + \gamma}{\sqrt{1+u_x^2+u_y^2}} + B$$

On peut supposer $A = 1$ et $B = 0$ quitte à remplacer $I(x,y)$ par $\frac{I-B}{A}$ puisque A, B, et I sont toutes des données supposées connues. En passant la racine de l'autre côté, on se retrouve à devoir résoudre l'équation suivante :

$$I(x,y)\sqrt{1+u_x^2+u_y^2} = -\alpha u_x - \beta u_y + \gamma \tag{2}$$

Il s'agit à présent de discrétiser cette équation afin de calculer les approximations des valeurs de la solution avec la FFM. On utilise le même schéma de discrétisation de la dérivée à l'ordre 1 que précédemment : en notant v_x (resp. v_y) la valeur minimale des voisins sur l'axe x (resp. y) du point z dont la valeur est à calculer on obtient $u_x \approx \epsilon_x \frac{u(z)-v_x}{h}$ (et l'approximation similaire en y) avec ϵ_x qui vaut 1 si le voisin choisi est à gauche de z et -1 s'il est à droite.
La valeur $u(z)$ à déterminer doit donc vérifier l'équation suivante :

$$I(x,y)\sqrt{1+\left(\frac{u(z)-v_x}{h}\right)^2+\left(\frac{u(z)-v_y}{h}\right)^2} = -\alpha\epsilon_x\frac{u(z)-v_x}{h} - \beta\epsilon_x\frac{u(z)-v_y}{h} + \gamma$$

Pour cela, on résout donc l'équation quadratique en λ suivante (en prenant la racine la plus grande comme pour l'équation eikonale précédente) :

$$I(x,y)^2\left[1+\left(\frac{\lambda-v_x}{h}\right)^2+\left(\frac{\lambda-v_y}{h}\right)^2\right] - \left[-\alpha\epsilon_x\frac{\lambda-v_x}{h} - \beta\epsilon_x\frac{\lambda-v_y}{h} + \gamma\right]^2 = 0 \tag{3}$$

En fonction du nombre de voisins disponibles, on peut comme précédemment décider d'éliminer des termes dans l'équation ou bien de résoudre l'équation plus tard lorsque plus de voisins seront disponibles.

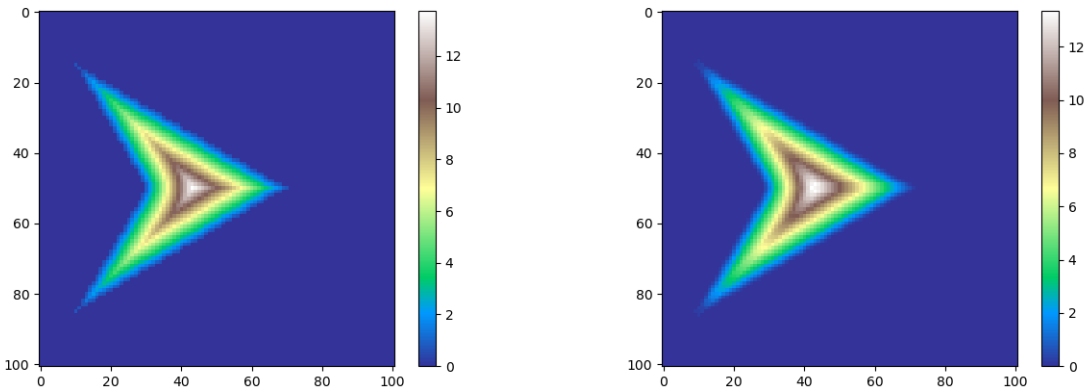
6.2 Méthode de test

Pour tester nos codes, nous allons utilisé un relief connu et enregistré sous forme de tableau numpy (renseignant l'altitude des points de la surface) pour simuler une intensité reçue par un appareil photo. Avec ce deuxième tableau numpy, nous pouvons résoudre l'équation SFS avec la méthode de fast-marching décrite précédemment et que nous avons implémentée nous-même pour obtenir un nouveau relief. Nous voudrions que ce nouveau relief soit le plus proche possible du relief original, mais ce n'est pas exactement cela qui va assurer que le code fonctionne. En effet en cherchant une solution de l'équation SFS, le critère à vérifier est que l'intensité associée au nouveau relief correspond bien à celui qui a été utilisé pour la résolution de SFS. Nous veillerons donc à comparer reliefs et intensités après chaque résolution de SFS. Puisque l'enjeu de la résolution de l'équation SFS est le calcul du volume de la forme étudiée, le calcul de l'erreur L^1 peut aussi être un indicateur pertinent.
Nous testerons notre code sur les reliefs que nous avons pu générer en résolvant les équations eikonales (cf les reliefs obtenus dans la partie **5.2**), mais également sur d'autres reliefs générés par des fonctions mathématiques de $[0,1]^2$ dans $\mathbb{R}$. Nous les mettrons sous forme de tableau *numpy* en les évaluant sur un maillage carré X de pas constant h selon les deux axes. Lorsque cela n'est pas précisé nous prendrons $h = 1/100$.
Expression des reliefs :
`gaussienne_x`$(x,y) = \exp\left(-3x^2\right)$

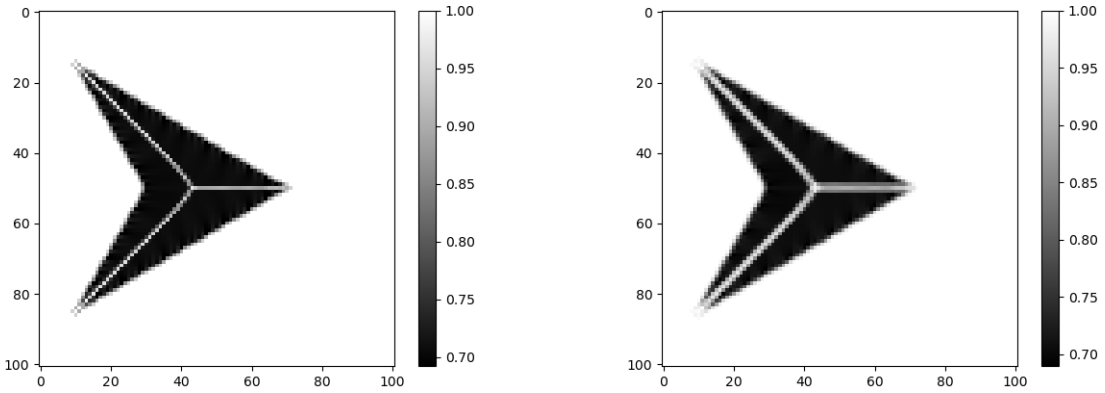
`gaussienne`$(x, y) = \exp\left(-5((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2)\right)$
`Volcano_gauss`$(x, y) = \exp\left(-3((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2)\right) - 0.7\exp\left(-9((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2)\right)$
`OneBump`$(x, y) = \max(0, \frac{1}{2} - 3((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2))$
`ThreeBump`$(x, y) = \max(0, 0.3 - 3((x-0.4)^2 + (y-0.5)^2), 0.25 - 3((x-0.65)^2 + (y-0.6)^2), 0.25 - 3((x-0.6)^2 + (y-0.35)^2))$
Voir en annexe la forme visuelle de ces reliefs.

6.3 Résultat de l'implémentation pour un éclairage vertical (partie à compléter)

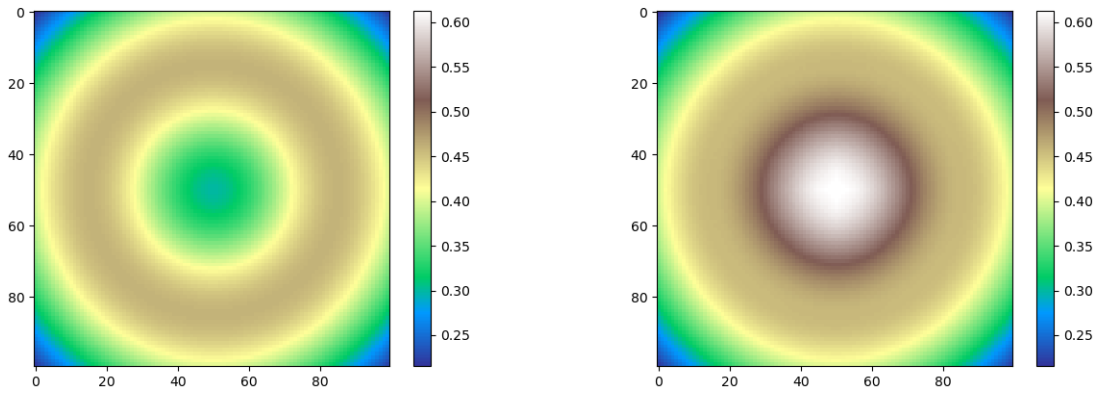
En premier lieu nous testons notre code sur un éclairage vertical puisque ce problème est proche de l'équation Eikonale. On obtient dans ce cas une reconstruction qui semble très correcte (on met les exemples du relief généré par l'équation eikonale sur un deltoïde :



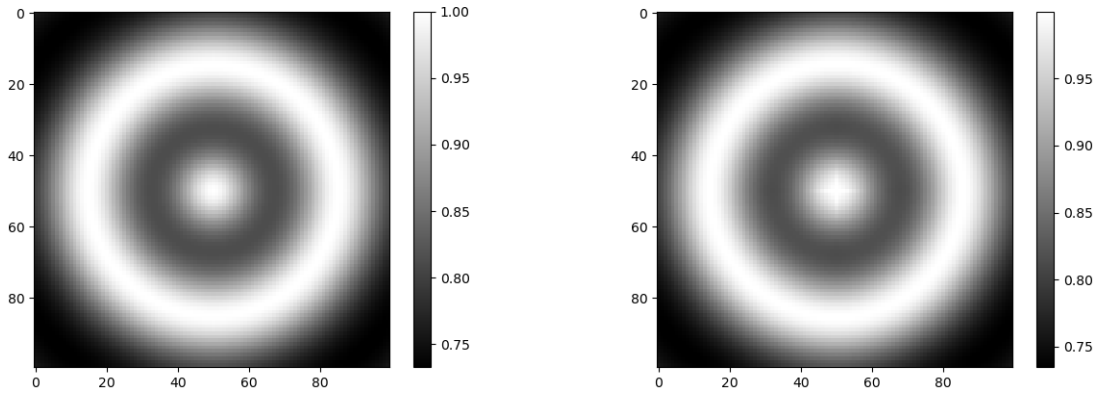
Relief original à gauche et reconstruction du relief après résolution de SFS sur tout le domaine rectangulaire à droite.
Les reliefs sont quasi-identiques et on peut vérifier qu'il en est de même pour les intensités lumineuse associées à chaque relief :



Intensité associée au relief initial à partir de laquelle SFS est résolue à gauche, et intensité associée au relief reconstruit à droite.
Les intensités sont quasiment les mêmes. Les petites différences apparaissent au niveaux des discontinuités de pente.
On obtient également des résultats justes pour les autres formes sauf pour `Volcano_gauss` qui donne un résultat surprenant :



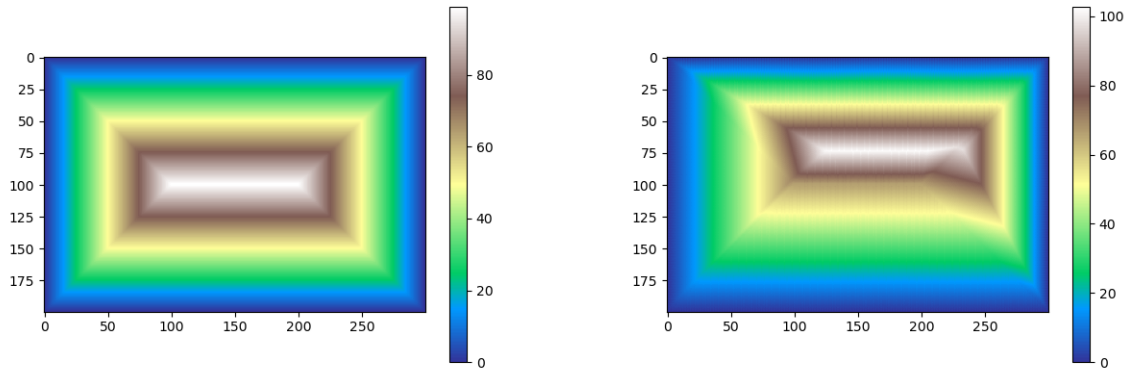
Relief original à gauche et reconstruction du relief après résolution de SFS sur tout le domaine rectangulaire à droite.
Les deux reliefs n'ont rien à voir au centre : au lieu de reconstruire un puits, l'algorithme a reconstruit une bosse. Cependant on peut comparer les intensités des deux reliefs :



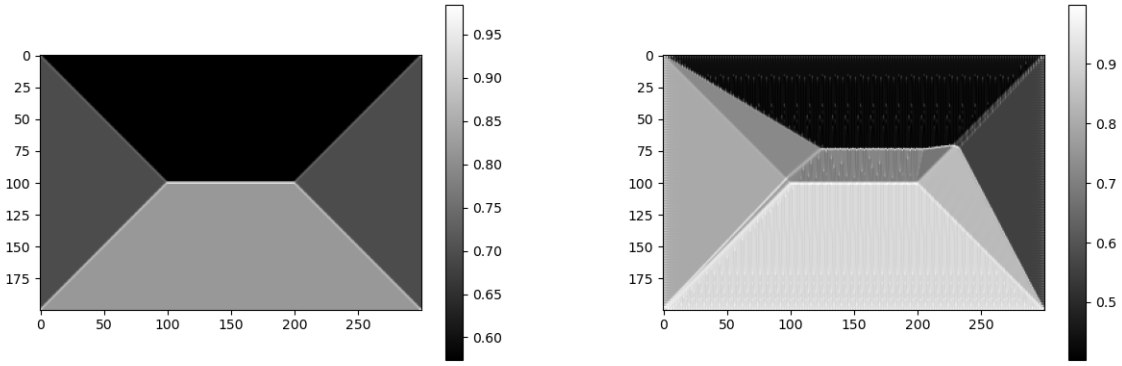
Intensité associée au relief initiale à partir delaquelle SFS est résolue à gauche, et intensité associée au relief reconstruit à droite.
Les deux intensités sont en extrêmement semblables ce qui montre qu'il n'y a en fait pas unicité dans l'équation SFS de ce problème et que l'algorithme a obtenu une autre solution que celle attendue !

6.4 Résultat de l'implémentation pour un éclairage non-vertical

On réalise à présent les mêmes tests mais pour un éclairage non vertical (i.e. $(\alpha, \beta, \gamma) \neq (0, 0, 1)$). Dans ce cas on obtient des résultats décevants en prenant $(\alpha, \beta, \gamma) = (-0.17, 0, 0.95)$ (cela correspond à une élévation du soleil de 80°) :



Relief original à gauche et reconstruction du relief après résolution de SFS sur tout le domaine rectangulaire à droite.



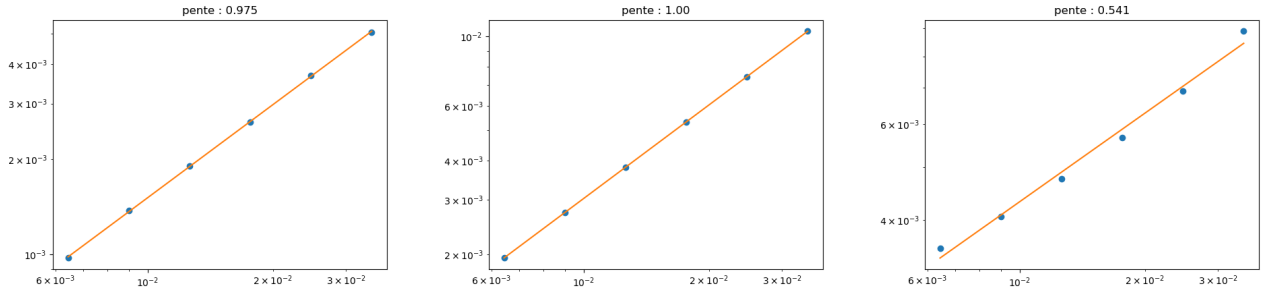
Intensité associée au relief initiale à partir de laquelle SFS est résolue à gauche, et intensité associée au relief reconstruit à droite.
Le relief original est clairement déformé et l'intensité associé à la reconstruction ne correspond plus à l'originale. Notre algorithme n'a donc pas trouvé la solution.
On peut continuer de réduire l'élévation du soleil mais on se rend rapidement compte que l'algorithme diverge (on obtient des valeurs extrêmement grandes par rapport à celles du relief original).
On peut tout de même garder l'espoir que dans le cas des reliefs générés par des fonctions mathématiques l'erreur commise par la résolution numérique tend vers 0 lorsque le pas h du maillage tend vers 0.

6.5 Étude expérimentale de la convergence du schéma

On réalise l'expérience suivante : On se donne une fonction de $[0,1]^2$ dans $\mathbb{R}$ représentant un relief et on l'évalue sur un maillage régulier de pas h (même pas pour chaque axe). On obtient le tableau *numpy* `height`. Après résolution de SFS avec un éclairage vertical pour ce relief, on obtient un nouveau tableau `U`. On peut associer ces tableaux à des fonctions constantes par morceaux de $[0,1]^2$ dans $\mathbb{R}$ et calculer la norme L^1 de leur différence, ce qui s'obtient simplement par la commande *numpy* suivante : `err = np.sum(np.abs(height-U))*h**2`.
En faisant varier le pas h, on peut déterminer expérimentalement l'ordre de convergence de l'algorithme on affichant sur un graphe l'erreur L^1 commise en fonction de h sur une échelle logarithmique (sur les deux axes). En effectuant une régression linéaire, on peut obtenir l'ordre de convergence avec le coefficient directeur de la droite.

6.5.1 Pour un éclairage vertical

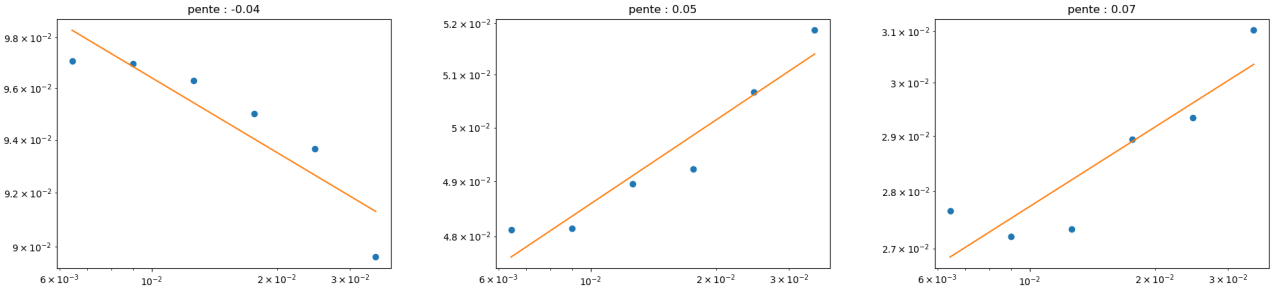
On réalise l'expérience pour les formes suivantes : `gaussienne`, `OneBump` et `ThreeBumps`. On obtient alors les graphes respectifs suivant :



Dans les deux premiers cas on obtient un nuage de points bien alignés sur une droite de pente 1. Cela est rassurant car cela signifie que l'algorithme converge et l'ordre de convergence est bien l'ordre du schéma utilisé pour approcher les dérivées. Pour `ThreeBumps` cependant, les points ne sont pas si bien alignés et l'ordre de convergence est approximativement de 0,5. Ce graphe n'est peut-être pas un bon indicateur car `ThreBumps` est la surface avec le plus de discontinuités de pente, mais cela montre que pour des reliefs moins régulier l'algorithme est moins performant.

6.5.2 Pour un éclairage non vertical

On refait exactement la même expérience mais pour une élévation du soleil de 80°. On obtient alors les résultats suivants (les formes sont traitées dans le même ordre) :

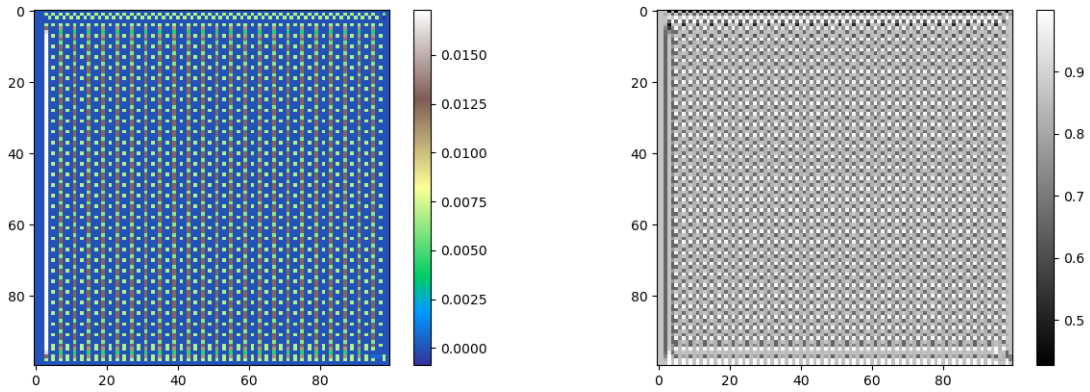


On ne peut donc pas conclure qu'il y a convergence du schéma. Le graphe le plus révélateur est celui de la forme `gaussienne` qui est la plus régulière mais pour laquelle l'erreur augmente avec le nombre de points du maillage.
On conclut donc que notre code n'est pas adapté à l'éclairage non vertical. Nous allons donc essayer de comprendre d'où vient le problème dans l'espoir de le résoudre.

7 Étude des problèmes intrinsèques à l'éclairage non-vertical

7.1 Résolution de SFS pour un relief nul

Prenons le cas le plus simple et voyons si notre algorithme renvoie le bon résultat. Nous considérons le relief constant égal à 0. D'après l'équation SFS (2), on a donc I constant égale à γ. La résolution numérique renvoie alors le résultat suivant pour $(\alpha, \beta, \gamma) = (-0.5, 0, 0.87)$ (élévation du soleil de 60°) :



Relief reconstitué et intensité associée à la reconstitution

On observe que même dans ce cas simple, la reconstitution est mauvaise et on n'obtient pas une solution. (0 est solution $\mathcal{C}^1$ au problème et donc unique solution de viscosité en particulier pour des conditions aux bords nulles, et l'intensité associé à la reconstruction ne correspond pas à l'originale). Les valeurs du relief obtenu sont petites mais bien trop grandes pour considérer qu'il s'agit de "zéros numériques". On peut en fait expliquer d'où vient ce problème :
Le premier point calculé par l'algorithme possède deux voisins aux bords du domaine qui sont nuls et avec de plus $I = \gamma$, l'équation quadratique de notre schéma numérique à résoudre est la suivante :

$$\gamma^2(h^2 + 2\lambda^2) - ((\alpha\epsilon_x + \beta\epsilon_y)\lambda - \gamma h)^2 = 0$$

En réordonnant les coefficients du polynôme par degré on obtient :

$$[2\gamma^2 - (\alpha\epsilon_x + \beta\epsilon_y)^2]\lambda^2 + 2\gamma h(\alpha\epsilon_x + \beta\epsilon_y)\lambda + 0 = 0$$

Et donc finalement les solutions de cette équation sont 0 et $\frac{-2\gamma h(\alpha\epsilon_x+\beta\epsilon_y)}{2\gamma^2-(\alpha\epsilon_x+\beta\epsilon_y)^2}$.
On observe que 0 est toujours racine mais elle n'est pas forcément la plus grande puisqu'*a priori* ϵ_x et ϵ_y sont de signes quelconques et donc le numérateur de la deuxième racine l'est également. Imposer leur signe reviendrai à imposer un sens de parcours des points du maillage ce qui est contraire au principe du Fast Marching et ne conviendrai pas pour de nombreuses formes plus compliquées qu'un relief nul. Il est ainsi inévitable pour un parcours de type fast-Marching qu'une valeur non nulle soit attribué à un point. On a d'ailleurs bien vérifié que les valeurs non nulles obtenues dans le cas précédent étaient celles attendues théoriquement (la deuxième solution

présentée au dessus).
Toutefois, on peut également faire la remarque que pour l'éclairage vertical on a $\alpha = \beta = 0$ et donc 0 est l'unique solution et ainsi l'algorithme résout bien le problème dans ce cas. Nous en tirons que dans le cas non-vertical, la racine de l'équation quadratique à choisir n'est peut-être pas toujours la plus grande contrairement au cas de l'équation eikonale.
On va alors essayer d'autres critères de sélection des racines.

7.2 Choisir la racine régularisante

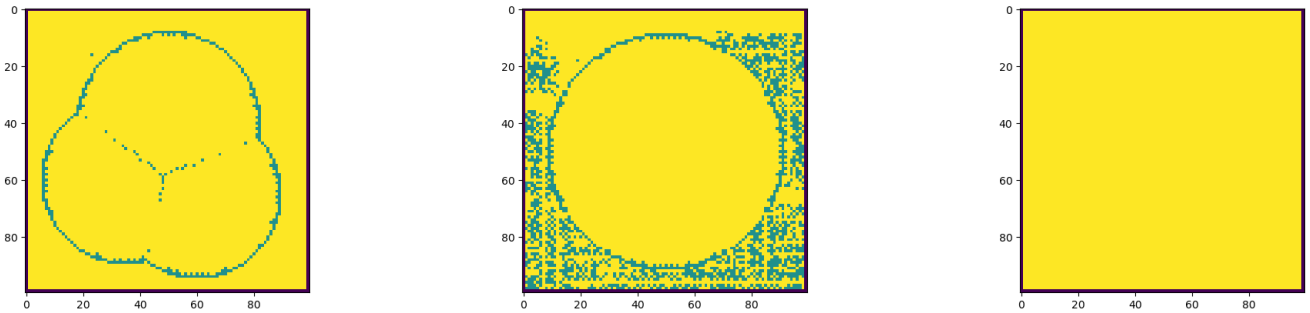
On propose ici une autre façon de choisir la racine associée à l'équation quadratique (3). On va essayer de prendre la racine qui minimise la variation du second ordre du relief en reconstitution. Concrètement, pour avoir une surface u plus lisse, on veut réduire les quantités $|\frac{\partial^2 u}{\partial x^2}|$ et $|\frac{\partial^2 u}{\partial y^2}|$. Pour implémenter cela nous utilisons le schéma classique centré pour discrétiser les dérivées secondes. Nous centrons ce schéma autour du dernier point x accepté et on cherche donc à minimiser la dérivée seconde dans la direction du point à calculer $(x + h\epsilon_i e_i)$. Une difficulté apparaît lorsque le 3ème point intervenant dans la dérivée seconde n'est pas encore calculé. On ne peut alors pas déterminer cette dérivée seconde et par défaut on choisira la racine la plus grande de l'équation quadratique à attribuer au point à calculer. Lorsque le 3ème point est déjà calculé, on choisi alors la racine λ de l'équation quadratique qui minimise la quantité $\frac{1}{h}|\lambda - 2u(x) + u(x - h\epsilon_i e_i)|$. On note qu'on aurait pu décider de centrer la dérivée seconde sur un autre point (le nouveau point à calculer par exemple) mais que l'on ne sera jamais sûr de disposer de trois points pour calculer la variation du second ordre et donc cette méthode n'est pas très adapté au parcours du fast-marching.
On obtient cependant d'abord un résultat satisfaisant dans le cas du relief nul : quelque soit la direction de l'éclairage, on reconstitue bien le relief nul (0 étant toujours racine de l'équation quadratique comme on l'a déjà montré).
Cependant cette méthode donne de très mauvais résultat pour les autres reliefs :

7.3 Choisir la racine la plus proche

Dans cette partie, on ne détaille pas une nouvelle méthode pour choisir la racine correspondant à la réelle solution car on suppose connu le relief initial.
On réalise l'expérience suivante : on se donne un relief `height` connu et on choisi la racine la plus proche du relief original lors de la résolution de l'équation quadratique. On note à chaque fois si la racine choisie est la plus petite ou la plus grande.

ECLAIRAGE VERTICALE

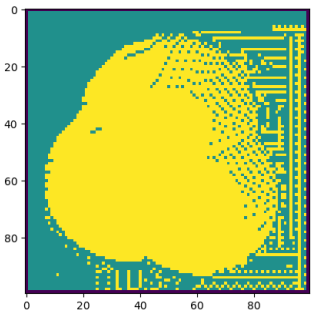
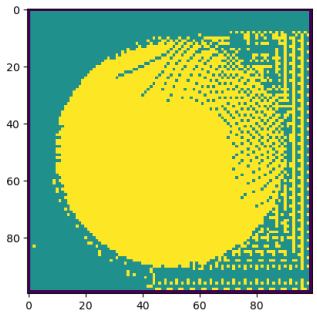
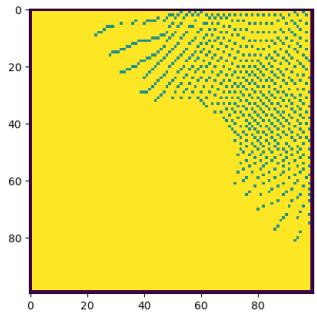
On présente ci-dessous les résultats obtenus avec les formes `ThreeBumps`, `OneBump` et `gaussienne` (dans le même ordre de gauche à droite) :



Pixel bleue : Bord du domaine, pas d'équation à résoudre
Pixel vert : La racine choisie est la plus petite
Pixel jaune : La racine choisie est la plus grande
On observe qu'en dehors des zones d'irrégularités des reliefs et des zones ou le relief est nul (et donc les racines y valent toutes les deux zéro) la racine choisie est toujours la plus grande. Cela confirme que ce choix est correct dans ce cas.

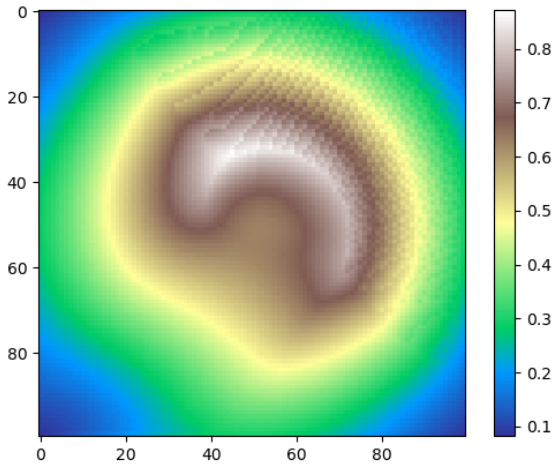
ELEVATION DE 80°

On présente ci-dessous les résultats obtenus avec les formes `ThreeBumps`, `OneBump` et `gaussienne` :

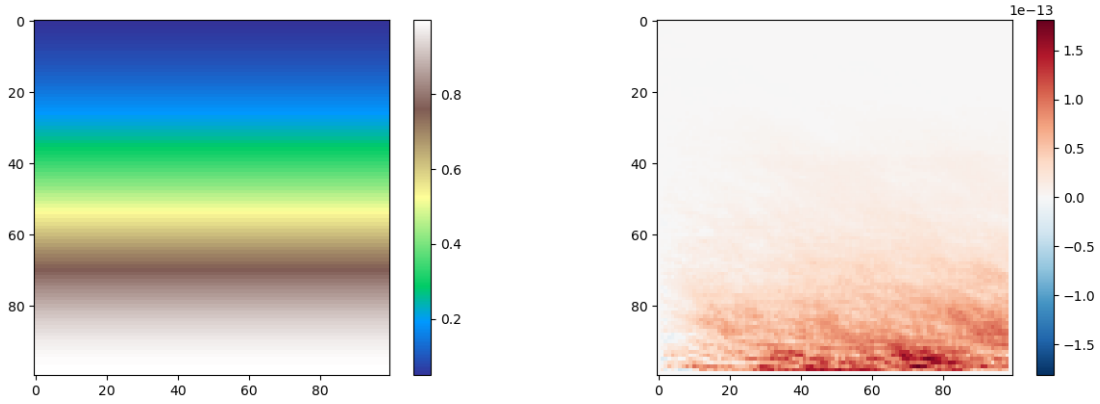




Les racines les plus petites ne sont utilisées que dans le domaine où `height` $= 0$ (on a déjà observé que pour un éclairage non verticale la racine choisie par défaut n'est pas toujours la bonne), mais dans le domaine "intéressant" la racine la plus grande est largement choisie. Dans les deux images on observe un nuage de points où la racine la plus petite a été choisie de façon minoritaire et dont la forme est similaire. Cela est probablement une artefact de l'algorithme pour un éclairage non vertical (contexte qui sort de la convergence du schéma).

Question : En choisissant la racine la plus proche du relief initial retrouve-t-on ce relief dans le cadre de l'éclairage non-vertical (où l'on ne dispose pas de preuve de convergence) ? L'intensité I est calculée à partir d'un relief original en discrétisant l'équations SFS. C'est cette même équation qui est résolue en sens inverse pour retrouver le relief et donc on devrait s'attendre à ce que le bonne solution soit toujours parmis les deux racines de l'équation quadratique.
Cependant, on se rend compte dés 80° d'élévation (ce qui est encore assez vertical) que le résultat retourné pour la forme `gaussienne` est significativement éloigné du relief original :



Cela est possible du fait que les points utilisés pour résoudre SFS localement (équation (3)) ne sont en général pas les mêmes que ceux utilisés pour le calcul de I à partir du relief original à cause du parcours par FMM. On va tout de même vérifier qu'il n'y a pas de problème dans l'implémentation de notre fast-Marching et que les erreurs apparaissent bien naturellement par la méthode de résolution.
Pour cela on prend un relief simple `gaussienne_x` dont on connait d'avance l'ordre de parcours des points du maillage (de gauche à droite puis ligne par ligne). On calcule alors l'intensité avec le schéma (3) où $v_x = u(x - he_1)$ et $v_y(x - he_2)$. Ainsi l'équation quadratique (3) doit toujours avoir une solution exacte puisque l'intensité est calculée avec exactement les mêmes valeurs. On se rend compte que quelque soit la direction de l'éclairage, on retrouve bien le relief original en choisissant la racine de l'équation quadratique la plus proche du relief.



Reconstitution du relief `gaussienne_x` et écart entre le relief original et la reconstitution pour une élévation du soleil de 20° et un azimut de 30°.

Notre code est donc bien correct (les erreurs étant de l'ordre de 10^{-12} ce qui peut être considéré comme 0), cependant pour certaines formes comme `gaussienne` le schéma de résolution du fast-marching ne permet pas de retrouver le bon relief car il arrive que les deux racines de l'équation quadratique soient éloignées du relief original comme le montre la figure précédente. Il n'y a donc finalement pas espoir que notre méthode de fast-marching fonctionne pour un éclairage quelconque et des formes quelconques.

7.4 Tests d'algorithmes pour traiter le cas vertical

Nous avons donc à présent vérifié le fait que notre algorithme de Fast-Marching n'est pas capable de traiter le cas d'un éclairage oblique, même en essayant d'adapter la méthode de choix de la racine à chaque étape au lieu de toujours considérer la plus grande. Nous avons donc essayé de comprendre différentes méthodes d'amélioration du Fast-Marching qui permettent de traiter ce cas plus général. Cependant, nous nous sommes rendus compte qu'il n'existe pas de méthode simple permettant de résoudre efficacement ce problème. Nous essayons tout de même de comprendre le principe des modifications apportées par Les premières approches trouvées[ref].
La première étape de cette méthode consiste à faire une approximation du relief dans le cas d'un éclairage presque vertical (moins de 10° d'écart avec la verticale). Cette première estimation est souvent peu précise, du fait des approximations grossières effectuées.
Afin d'améliorer cela, on peut ensuite appliquer une méthode itérative[ref] permettant, par approximation successive, de faire converger notre suite d'estimations vers la solution voulue. Cette méthode semble plus efficace, cependant elle parait fonctionner de moins en moins bien lorsqu'on considère des éclairages de plus en plus obliques, ou des formes moins lisses.
L'autre méthode trouvée,[ref], bien que semblant plus efficace, s'éloigne beaucoup plus du principe du Fast-Marching en proposant une approche basée sur la théorie du contrôle.
Nous avons essayé d'implémenter la première de ces 3 méthodes, les autres étant bien plus difficiles à comprendre et à implémenter.[explications]
Les résultats obtenues sont mitigés, même pour des éclairages très proches de la verticale, la reconstruction est loin d'être parfaite. Il faudrait, afin d'avoir de meilleurs résultats, implémenter la méthode itérative, chose que nous n'avons pas faite faute de temps.

Nous avons aussi testé notre propre méthode itérative [explications si nécessaire].

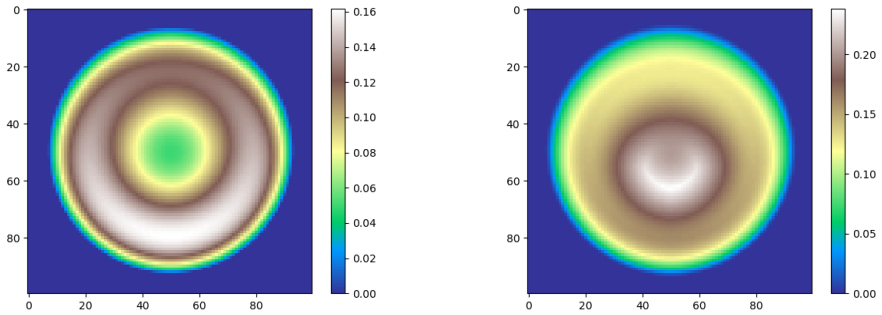
Tous ces tests n'étant pas vraiment concluants, nous abandonnons l'étude du cas oblique pour nous recentrer sur le cas vertical, car l'implémentation d'algorithme issus du Fast-Marching adaptés au cas oblique nous aurait pris un temps trop conséquent pour des résultats trop peu convaincants.

8 Reconstitution de la bonne solution

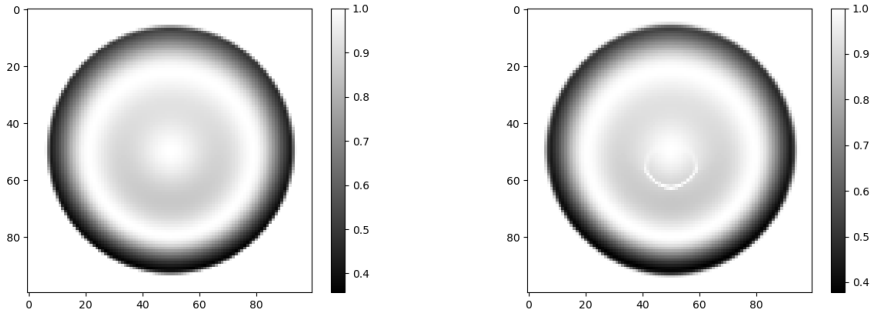
Jusqu'à maintenant, nous n'avons regardé que des cas où notre algorithme reconstituait bien le relief pour un éclairage vertical. Nous nous intéressons maintenant au cas où la solution trouvée par l'algorithme n'est pas celle recherchée.

8.1 Cas où le relief n'est pas solution de viscosité

Le premier cas où apparaît cette erreur est l'étude de la forme *Volcano* (cf annexe). En effet, nous remarquons que, lors de la résolution par notre algorithme de l'équation SFS obtenue avec un éclairage vertical, la solution reconstituée n'est pas le relief voulu.

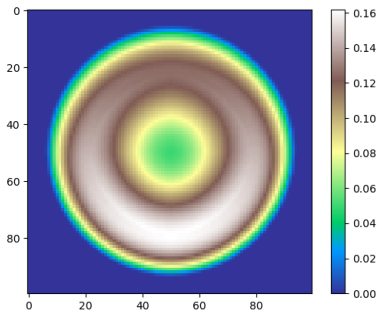


(à gauche le relief initial/ à droite le relief reconstitué) Pourtant, l'éclairage re-simulé sur la forme calculée est bien quasiment identique à l'éclairage initial.



L'explication de ce phénomène est donnée par le fait que la forme *Volcano* n'est pas solution de viscosité du problème, tandis que la solution trouvée par l'algorithme l'est.
On sait en effet que celui-ci converge toujours (au moins pour le cas vertical) vers la solution de viscosité qui est unique, pour peu qu'on ait des conditions de Dirichlet sur toute la frontière du domaine.
Or, d'après ce qui a été dit précédemment sur l'équation eikonale, le relief *Volcano* n'est pas la solution de viscosité.
Ainsi, même si on a vu qu'il était justifié de chercher la solution de viscosité du problème SFS afin de reconstituer le relief, il existe des formes pour lesquelles la méthode ne semble pas adaptée telle quelle.

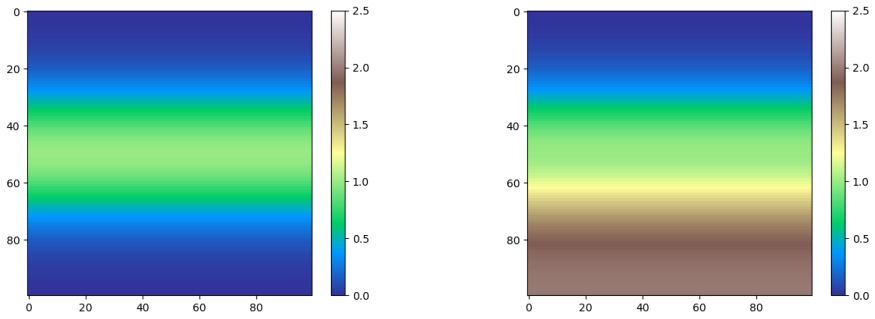
Afin de résoudre ce problème, nous choisissons d'utiliser une méthode classique dans la résolution du problème SFS : en plus des conditions aux bords "classiques", nous ajoutons la valeur au centre du cratère parmis les valeurs initiales. Plus précisément, le valeur dans le tableau représentant la solution correspondant au centre du cratère est fixée comme si elle appartenait au bord, et est directement classée comme *Trial*. Le but avec cette opération est d'enlever l'incertitude liée à l'extremum de la fonction, et donc de forcer notre algorithme à converger vers la bonne solution. Les résultats obtenus sont conformes à ce qui est attendu :



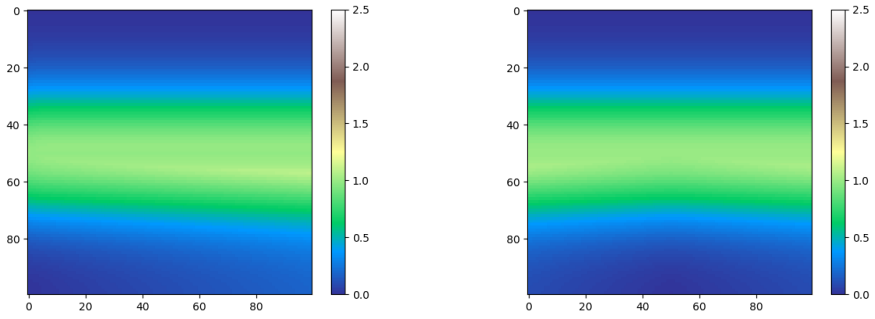
Ainsi, la connaissance des points d'extremum du relief à reconstituer permet dans ce cas de forcer l'algorithme à faire converger la solution approchée vers la solution voulue.

8.2 Conditions aux bords insuffisantes

Nous essayons ici de reconstruire une forme simple (*gaussiennex*, cf annexe), mais en restreignant les conditions aux bords de telle sorte qu'il n'y ait plus qu'un bord du domaine sur lequel les valeurs initiales du relief sont fixées. On sait que pour un éclairage verticale et des conditions aux bords "habituelles", notre algorithme reconstitue parfaitement la forme. Nous testons dans le cas où seul le bord "$y = 0$" est fixé.



On observe alors que la solution trouvée n'est pas le relief voulu. En effet, sur la première moitié du domaine, il est reconstitué fidèlement, mais une fois le maximum de la gaussienne passé, la solution approchée continue d'augmenter au contraire de la solution voulue qui diminue. Le problème est ici qu'on ne dispose pas de l'unicité de la solution de viscosité pour le problème posé : les conditions aux bords ne sont pas suffisantes, ce qui fait que, bien que la gaussienne soit bien solution de viscosité, la méthode numérique converge vers une autre approximation, elle aussi solution de viscosité.
Afin de résoudre ce problème, nous pouvons d'abords simplement remarquer que le fait de fixer un bord supplémentaire suffit pour retrouver la bonne solution, ce qui semble logique au vue de ce que l'on a vu sur l'unicité et le comportement de la gaussienne. En fait, dans ce cas précis, le fait de fixer la valeur d'un unique point sur le bord opposé au bord initial suffit pour obtenir la bonne solution.

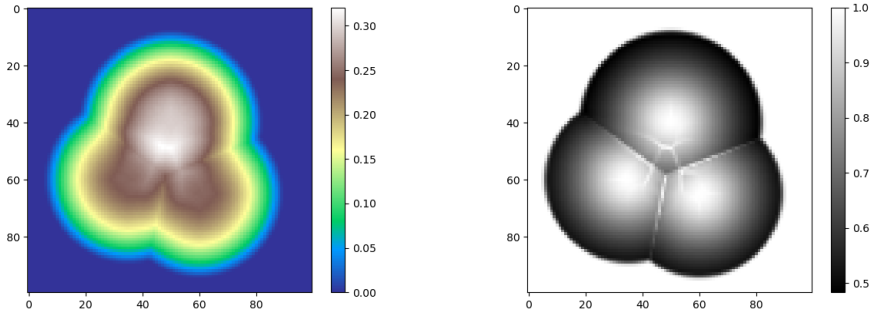


Ainsi, il est nécessaire d'avoir suffisamment de conditions aux bords afin de reconstituer fidèlement le relief. Cela nous resservira plus tard lors du traitement de formes discontinues.

8.3 Irrégularités et solution

Nous nous intéressons finalement dans cette partie à la reconstitution d'une forme plus irrégulière, la forme *Three-Bumps* (cf annexe).

Le résultat obtenu pour cette forme avec un éclairage vertical est le suivant :



On observe que sur chaque bosse, une fois passé le maximum le front d'onde commence à redescendre avant de subitement remonter, créant une discontinuité d'intensité visible sous la forme d'une "petite courbe" passant sur les trois bosses. La reconstruction n'est donc ici pas vraiment fidèle, et cela se ressent au niveau du volume calculé par rapport au volume initial[chiffres].

Nous cherchons à comprendre l'origine de cette erreur. Pour cela,

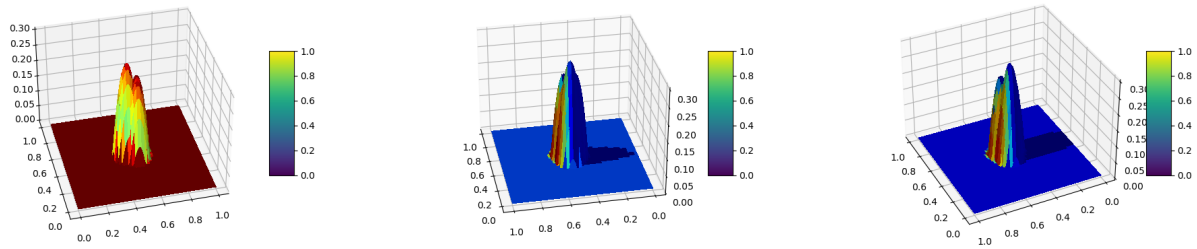
9 Ombres et discontinuités

9.1 Objectifs

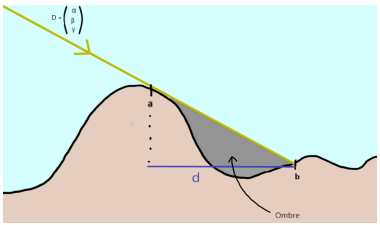
En supposant qu'on peut reconstruire un relief étant donné un éclairage quelconque, on cherche maintenant à reconstruire un relief à partir d'une image comportant des ombres. En effet, pour une image réel, il y a presque toujours des ombres plus ou moins grandes à partir du moment où l'éclairage est un peu décalé de la verticale.
Le but est donc de :

- identifier les zones avec ombres
- résoudre le problème sur le domaine privé de ces zones
- utiliser les nouvelles conditions aux bords données par la modélisation mathématiques des ombres

La présence d'une ombre à une endroit donné doit être visible par le fait que l'intensité sur une image réelle est supposée être nulle (égale à B sur le modèle initial) lorsqu'il y a une ombre. Il est donc nécessaire d'améliorer notre fonction simulant l'éclairage afin que celle-ci puisse créer des ombres (en particulier, lorsque le produit scalaire de la normale à la surface avec le vecteur d'émission de lumière est négatif, la valeur de l'intensité doit être nulle et non négative). Nous avons donc codé cette amélioration en Python, donnant les résultats suivants plutôt satisfaisant:



Ensuite, on peut schématiser la présence d'une ombre comme ceci :



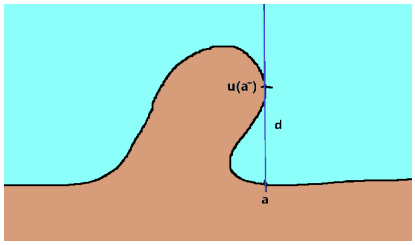
On remarque alors qu'on dispose de deux conditions supplémentaire aux bords de la zone d'ombre :
D'une part, du côté le plus haut (= le plus proche du soleil), la dérivée directionnelle de la fonction u représentant le relief, selon le vecteur $\overrightarrow{n} = (-\alpha, -\beta)$, est égale à $-\gamma$.
D'autre part, puisqu'on connaît la largeur d de l'ombre, et que l'angle θ est donné par $\tan(\theta) = \frac{\gamma}{\sqrt{\alpha^2+\beta^2}}$. On en déduit que la différence de hauteur entre a et b est $\frac{d}{\tan(\theta)}$.
Les conditions sont donc :

$$\frac{\partial u}{\partial n} = -\gamma$$

$$u(a) - u(b) = \frac{d}{\gamma}\sqrt{\alpha^2 + \beta^2}$$

.

Cependant, ne disposant pas de méthode permettant de résoudre le problème pour un éclairage non-vertical, nous cherchons un problème similaire à résoudre pour le cas vertical. Nous choisissons par conséquent de nous intéresser au cas d'une discontinuité dans le relief supposée connue. On considère en effet un cas comme suit :



De la même façon que pour les ombres, on cherche ici à résoudre le problème sur les deux zones de part et d'autre de la discontinuité ; ainsi le relief est toutjours bien une fonction u possèdant une discontinuité en a : $u(a^+) \neq u(a^-)$. Afin de se placer dans un cas similaire à celui des ombres, on suppose connus :

- L'emplacement exact de la discontinuité
- La différence de hauteur $h = u(a^+) - u(a^-)$

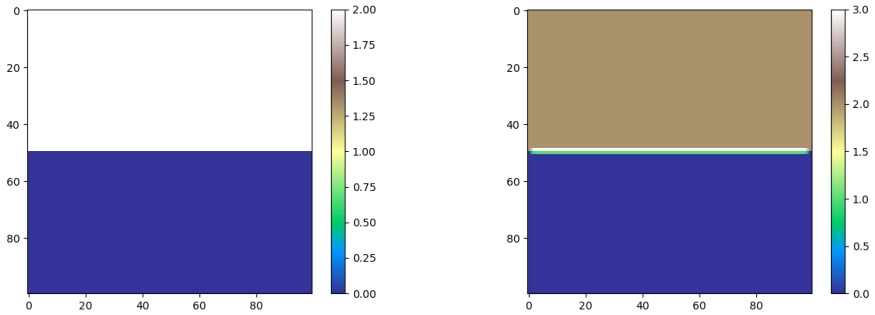
Dans le cas des ombres, ces données peuvent être obtenues en analysant l'intensité, ce qui n'est pas le cas ici, d'où la nécessité d'avoir ces données. On a ainsi un problème ressemblant plus ou moins à celui des ombres dans le cas oblique (les deux ne sont cependant pas exactement les mêmes). Nous de disposons pas de résultats de convergence dans ce cas, et l'approche sera donc exclusivement expérimentale.

9.2 Implémentation

On commence par ne pas prendre en compte la condition de différence de hauteur au niveau de la discontinuité. Afin d'implémenter cela, nous rajoutons un masque sur lequel nous identifions tous les points situés d'un côté ou de l'autre de la discontinuité, en indiquant de quel côté se trouve la discontinuité selon ce point. Nous supposons donc un tel tableau connu. Nous appliquons ensuite l'algorithme de Fast-Marching comme dans le cas usuel, mais lorsqu'on traite un point, si celui-ci et situé au bord de la discontinuité, alors les points situés de l'autre côté ne sont pas pris en compte dans les calculs.

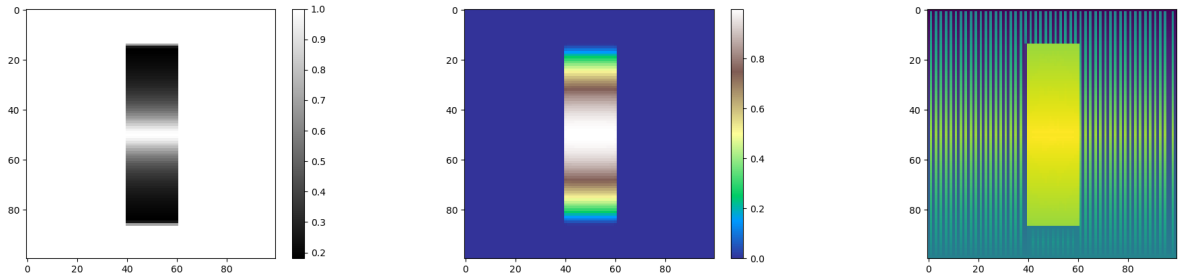
9.3 Résultats

Nous commençons par tester pour un cas simple, celui de deux demi-plans de hauteurs différentes séparés par une droite. Nous obtenons d'abord le résultat suivant :



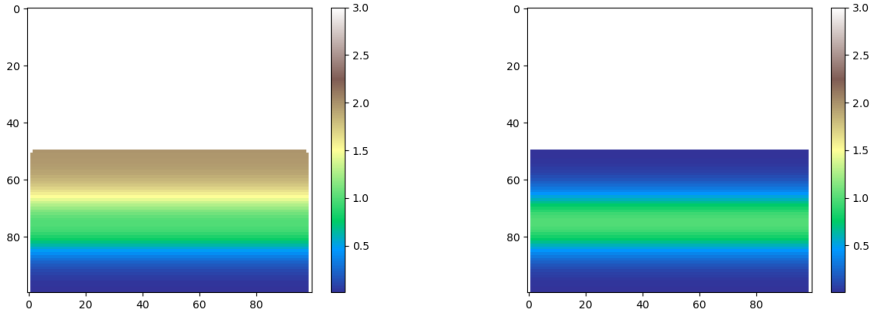
On observe que la reconstruction est fidèle, sauf au niveau de la discontinuité où il y a eu un effet de "lissage". Cela est dû au premier calcul d'intensité lumineuse simulée, fait avec la fonction *numpy.gradient*, qui "régularise" le relief en calculant les dérivées partielles de façon centrée, ce qui n'est pas voulu au niveau de la discontinuité. Nous utilisons donc à présent une simple discrétisation amont/avale au niveau de la discontinuité des dérivées partielles, afin que seul les points du bon côté soient considérés. Cela mène bien au résultat voulu, à savoir le relief initial.

On teste ensuite avec une forme plus complexe, et plus réaliste car formée d'une unique composante connexe : un plan sur lequel on "pose" une parabole, à la façon d'un pont. En utilisant la même méthode, et en prenant garde à ce que les calculs d'intensité soient justes, on obtient le résultat suivant :

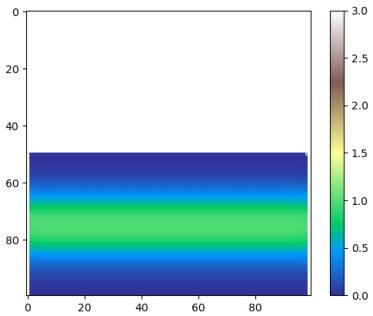


On remarque qu'on obtient alors exactement la reconstruction voulue.

Ainsi, pour des formes simples, un Fast-Marching classique prenant simplement en compte la discontinuité semple a priori suffisant pour résoudre le problème.
On se demande alors dans quelles circonstances la condition de différence de hauteurs pourrait être utile.
On utilise alors ce que nous avons remarqué durant la partie précédente concernant les problèmes de reconstruction de la solution lorsque les conditions aux bords ne sont pas suffisantes. On construit donc la forme discontinue suivante :



Le demi-plan supérieur, prolongé de bandes à droite et à gauche sur le demi plan inférieur, est constant de valeur 2. Sur le demi-plan inférieur, on construit une gaussienne de maximum 1. Lorsqu'on utilise notre algorithme, on n'obtient alors pas une gaussienne, comme on pouvait s'y attendre, faute de conditions aux bords suffisantes.
Il faudrait alors, de la même façon qu'en fixant un point au centre du volcan on retrouve le bon relief, pouvoir fixer les valeurs sur la discontinuité pour retrouver la bonne solution. Le problème est ici qu'on ne dispose pas de la hauteur du relief sur le discontinuité, mais simplement de la différence de hauteur. Du fait de la façon dont notre algorithme est construit, il est donc nécessaire de construire les 2 composantes connexes l'une après l'autre, et d'utiliser les conditions aux bords une fois que la première (le demi plan supérieur) a été reconstruit. Cela nécessite donc de savoir à l'avance dans quel ordre traiter les composantes, ce qui n'est pas vraiment réaliste. Après implémentation, nous obtenons tout de même :



ce qui est exactement le relief initial. Cela n'est pas vraiment surprenant car avec cette méthode, nous effectuons en fait 2 fois la résolution du problème de façon totalement séparée, avec à chaque fois des conditions aux bords suffisantes pour bien reconstruire les formes. Ce n'est donc pas vraiment réaliste de vouloir résoudre de ce problème de cette façon dans le cas général, et il serait préférable de pouvoir prendre en compte les conditions au niveau de la discontinuité en "temps réel" durant la résolution du problème sur tout le domaine, chose que nous n'avons pas pu faire avec notre algorithme.

10 Conclusion