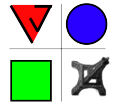


Résolution numérique des équations de Saint-Venant, mise en oeuvre en volumes finis par un solveur de Riemann bien balancé

P.-Y. Lagrée
CNRS & Sorbonne Université (exUPMC Univ Paris 06), UMR 7190,
Institut Jean Le Rond ∂'Alembert, Boîte 162, F-75005 Paris, France
pierre-yves.lagree@upmc.fr ; www.lmm.jussieu.fr/∼lagree

February 8, 2024

Abstract

Nous montrons comment résoudre par volumes finis les équations de Saint-Venant (et l'onde cinématique). Des programmes en "C" sont expliqués (du `python` est proposé aussi). Des exemples sont ensuite présentés (rupture de barrage, ressaut, "déferlement"...). L'ensemble est empirique, les notions sont abordées de manière rapide, allusive, simplifiée et non rigoureuse.

Page ouèbe de ce fichier `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/code_C_saintvenant.pdf`
Ces notes ne sont pas un mode d'emploi de *Gerris* (directive `GfsRiver`) ni de *Basilisk* fichier `http://basilisk.fr/src/saint-venant.h`
Un code en `C` modifiable est proposé et expliqué, il permet de reproduire les exemples classiques indiqués.

1 Rappel des équations

1.1 Rappel de l'établissement des équations

Le système que nous considérons est celui des Équations de Saint-Venant (CRAS 1871 Adhémar Jean-Claude Barré de Saint- Venant, équations *Shallow Water* en anglais ou *depth averaged*), remarquons la date 1871: plus de 150 ans déjà, voir la vidéo `https://www.youtube.com/watch?v=rmAFsEpl7HQ`).

Ces équations sont présentées et discutées dans `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf`, nous les avons établies sur une tranche fixe sous la forme d'un bilan de la masse et d'un bilan de quantité de mouvement, avec $h(x,t)$ qui est la hauteur d'eau, $u(x,t)$ la vitesse moyenne supposée constante sur l'épaisseur (sinon on met un coefficient Γ), τ frottement au fond et $Z(x,t)$ qui est la "topographie" ou forme du fond, g la gravité. Elles s'écrivent

$$\frac{\partial}{\partial t}\int_{x_1}^{x_2} h dx = -[hu]_{x_1}^{x_2} \text{ et } \frac{\partial}{\partial t}\int_{x_1}^{x_2} \rho uh dx = -[\rho hu^2]_{x_1}^{x_2} - [\rho g\frac{h^2}{2}dx]_{x_1}^{x_2} + \int_{x_1}^{x_2} \rho g(-Z')dx - \int_{x_1}^{x_2} \tau dx. \quad (1)$$

Comme le domaine est fixe, on permute $\frac{\partial}{\partial t}\int_{x_1}^{x_2} C dx = \int_{x_1}^{x_2} \frac{\partial}{\partial t} C dx$ et comme $[C]_{x_1}^{x_2} = \int_{x_1}^{x_2} \partial_x C dx$ (pour un C générique), la forme locale associée est :

$$\begin{cases} \partial_t h + \partial_x uh = 0, \\ \partial_t \rho uh + \partial_x \left(\rho u^2 h + \rho g \frac{h^2}{2} \right) = -\rho g h \partial_x Z - \tau. \end{cases} \tag{2}$$

Les mêmes équations de Saint-Venant (eq. 1) dans un domaine qui est cette fois en mouvement $D(t)$ qui est le segment $[x_1(t), x_2(t)]$ transporté dans l'écoulement s'écrivent

$$\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} h dx = 0, \text{ et } \frac{d}{dt}\int_{x_1(t)}^{x_2(t)} hu dx = -[g\frac{h^2}{2}dx]_{x_1(t)}^{x_2(t)} + \int_{x_1(t)}^{x_2(t)} \rho g(-Z')dx - \int_{x_1(t)}^{x_2(t)} \tau dx. \tag{3}$$

Sachant que la dérivée d'intégrale en 1D dans un domaine D qui est le segment $[x_1(t), x_2(t)]$ en mouvement est

$$\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} C dx = \int_{x_1(t)}^{x_2(t)} (\frac{dC}{dt}dx + C\frac{dx}{dt}) = \int_{x_1(t)}^{x_2(t)} ((\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x})dx + C\frac{\partial u}{\partial x}dx) = \int_{x_1(t)}^{x_2(t)} (\frac{\partial C}{\partial t} + \frac{\partial (Cu)}{\partial x})dx$$

on retrouve alors bien sûr à partir de (Eq. 3) la même forme locale (Eq. 2) que celle issue de (Eq. 1).

La forme (Eq. 1) est établie sur un segment fixe, cette forme est adaptée à la résolution numérique en volumes finis car on travaillera justement sur un segment fixe (le "volume"). Ce volume sera de taille petite pour la précision. La forme (Eq. 3 correspond à un segment avec des extrémités qui bougent. Quand les extrémités bougent, on utilise la règle de Leibniz. Dans le cas d'une discontinuité en x_c qui se déplace à la vitesse w dans le domaine, pour la masse et la quantité de mouvement, on montre que les discontinuités sont:

$$\left[|\rho h(w-u)|\right] = 0 \quad \text{et} \quad \left[|\rho uh(w-u) - \frac{1}{2}\rho g h^2|\right] = 0. \tag{4}$$

1.2 Les équations et leur fermeture

Le système que nous considérons est donc:

$$\begin{cases} \partial_t h + \partial_x Q = 0 \\ \partial_t Q + \partial_x \left[\Gamma \frac{Q^2}{h} + g\frac{h^2}{2} \right] = -gh\partial_x Z - \frac{\tau}{\rho} \end{cases}, \tag{5}$$

avec $h(x,t)$ est la hauteur d'eau et $Q(x,t)$ le débit, $u(x,t)$ la vitesse moyenne, $Q(x,t) = h(x,t)u(x,t)$ le flux, Γ facteur de forme (coefficient de Boussinesq), τ frottement (fonction de Q et orienté par $Q/|Q|$) au fond et $Z(x,t)$ est la "topographie" ou forme du fond, g la gravité...

On considère que les hypothèses pour l'établir (couche mince principalement) sont valables dans des cas variés qui ne sont pas uniquement ceux de l'eau.

Le facteur de forme (coefficient de Boussinesq) et le frottement au fond dépendent du choix du profil de base et de la nature de l'écoulement.

Ces équations sont présentées et discutées plus en détail dans `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf`

Note: passage à une équation seulement

Une simplification importante se produit lorsque l'inertie ($\partial_t Q + \partial_x \Gamma \frac{Q^2}{h}$) est négligeable c'est "l'onde de crue diffusante". On a $0 = -gh\partial_x h - gh\partial_x Z - \frac{\tau}{\rho}$, on trouve Q que l'on met dans l'équation de la masse. Cela mène à une équation 1D de transport/ diffusion...).

$$\partial_t h + \partial_x Q(h, \partial_x h, \partial_x Z) = 0 \tag{6}$$

Si la pression $gh\partial_x h$ est aussi négligeable c'est l'"onde cinématique": $0 = -gh\partial_x Z - \frac{\tau}{\rho}$, on trouve Q que l'on met dans l'équation de la masse.

Fermetures pour Saint-Venant

Pour les équations de Saint-Venant (5), le facteur de forme Γ et le frottement au fond τ dépendent du choix du profil de base et s'expriment en fonction de Q et h.

• Pour les canaux et fleuves, en turbulent, faute de mieux, pour résoudre Saint-Venant Eq. 5 on choisit

$$\Gamma = 1 \text{ et } \tau = \rho c |Q| \frac{Q}{h^\beta}$$

le facteur de forme vaut l'unité car on suppose que le profil est assez plat, τ est le frottement au fond et c le coefficient de friction, ce coefficient dépend de la nature du sol. Avec $\beta = 2$ on a la loi de Chézy (dite aussi Darcy Weissbach, frottement turbulent habituel en u^2) avec $\beta = 7/3$, on a la loi de Manning-Strickler.

• Pour de l'eau en couche mince, en laminaire (ruissellement autour d'un Poiseuille), par exemple la lave en première approximation, ou pour un fluide assez visqueux (la pâte à crêpe, la pluis qui ruisselle sur une véranda...)

$$\Gamma = \frac{6}{5} \text{ et } \tau = 3\mu \frac{|Q|}{h^2}$$

• Pour un granulaire (milieux constitué de particules de plus d'une dizaine de micromètres, le sable, le gravier, les rochers.... en dessous, il s'agit des poudres qui ont des propriétés différentes), la modélisation de leur écoulement passe par Saint-Venant (appelé Savage-Hutter dans ce cas). On s'inspire du frottement de Amontons-Coulomb liant la contrainte tangentielle τ à la contrainte normale ρgh. On définit un coefficient de friction μ qui peut être variable $\mu(I) = \mu_s + \Delta\mu/(1 + I_0/I)$ ($\mu_s \sim \Delta\mu \sim I_0 \sim 0.3$ suivant les types de matériaux) et fonction de $I = (d/h)5Q/2h/\sqrt{gh}$ (nombre sans dimension que l'on peut construire avec les quantités en jeu et la taille du grain d). Le coefficient μ est compris au final entre 0.2 et 0.4 suivant les matériaux et : $\tau = \mu\rho gh$. Le profil de base est calculable, c'est le profil de Bagnold `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/mainM2EMN.pdf` (on ne donne pas de détail ici), on peut donc estimer Γ

$$\Gamma = \frac{5}{4} \text{ et } \tau = \mu\rho gh \frac{|Q|}{Q}$$

• Pour des fluides complexes, si on néglige le seuil, (les écoulements d'avalanche/inondation finissent toujours par être très sales, ce sont des mélanges complexes de cailloux, de graviers, de boue et d'eau) des fluides en loi de puissance sont souvent employés. Pour le cas des glaciers en écoulement

$n = 1/3$, c'est la loi de Glen. Pour la boue $0.1 < n < 0.4$

$$\Gamma = \frac{2(1+2n)}{2+3n} \text{ et } \tau = c_n \mu_n \left(\frac{Q}{h^2}\right)^n \frac{|Q|}{Q}$$

• Pour de la neige, on modifie le μ précédent, on garde un μ_0 constant et on ajoute une friction de type fluide en carré de la vitesse et ξ un coefficient. C'est la loi de Voellmy (μ_0 valeur typique 0.2 et ξ valeur typique 500m/s)

$$\Gamma = 1 \text{ et } \tau = \rho g h (\mu_0 + \frac{1}{\xi}\frac{Q^2}{h^3})\frac{|Q|}{Q}$$

• Pour les fluides à seuil (boue, béton...), il faut rajouter une condition, liée au seuil de contrainte. Dans ces cas l'usage est d'utiliser l'onde cinématique, ou l'onde diffusante Eq. 6, voir les écoulements de Bingham `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/mainM2EMN.pdf`

• Panache (et convection naturelle): Les écoulements de type panache se résolvent avec les équations de couche limite intégrées sur leur épaisseur. On retrouve alors des équations de type Saint-Venant.

∗ En pratique, tout le monde prend abusivement $\Gamma = 1$ pour les résolutions, ce que nous ferons par la suite, la raison est que cela permet d'écrire une équation d'énergie et d'assurer l'invariance Galiléenne des équations. Le frottement n'est en revanche pas pris constant mais fonction des variables Q et h; il dépend donc du type d'écoulement (turbulent, laminaire, granulaire...).

1.3 Equation de bilan locale

On peut écrire (5) sous une forme compacte

$$\partial_t U + \partial_x F(U) = S(U), \tag{7}$$

avec une nouvelle définition de U, $F(U)$ et $S(U)$. On identifie U au vecteur des variables conservatives et $F(U)$ le flux

$$U = \begin{pmatrix} h \\ Q \end{pmatrix}, \; F(U) = \begin{pmatrix} Q \\ \frac{Q^2}{h} + g\frac{h^2}{2} \end{pmatrix} \tag{8}$$

le terme source lié à la forme du fond et au frottement:

$$S(U) \quad = \begin{pmatrix} 0 \\ -gh\partial_x Z - \tau/\rho \end{pmatrix}. \tag{9}$$

puis on coupe en autant de parties que de contributions à $S(U)$...

1.4 "Split": décomposition convection-frottement

Pour résoudre (7) il est d'usage de décomposer en étapes (on "coupe" S en plusieurs parties, en commençant par pas de source, puis après chaque source une à une c'est le *splitting*). Cela renvoie à la discrétisation $\partial_t U = (U^{n+1} - U^n)/\Delta t$, on va décomposer l'accroissement $(U^{n+1} - U^n)$ en plusieurs étapes, ici trois étapes (U^n donné, puis $U^{n+1/3}$, puis $U^{n+2/3}$ et enfin U^{n+1}). L'équation de conservation

$$\partial_t U + \partial_x F(U) = S(U),$$

est donc "semi discrétisée" en

$$(U^{n+1} - U^n)/\Delta t + \partial_x F(U) = S(U),$$

dans S il y a deux termes, un lié au fond $S_Z = (0, -gh\partial_x Z)$ et un lié au frottement $S_f = (0, -\tau/\rho)$

$$\partial_t U + \partial_x F(U) = S_Z(U) + S_f(U),$$

La première étape est une étape conservative, pas de S!

$$(U^{n+1/3} - U^n)/\Delta t + \partial_x F(U^n) = 0,$$

la seconde avec le premier terme source,

$$(U^{n+2/3} - U^{n+1/3})/\Delta t = S_Z(U^{n+1/3}),$$

la troisième avec le deuxième terme source:

$$(U^{n+1} - U^{n+2/3})/\Delta t = S_f(U^{n+2/3}),$$

au final, en ajoutant toutes les contributions on retrouve bien:

$$(U^{n+1} - U^n)/\Delta t = (U^{n+1} - U^{n+2/3})/\Delta t + (U^{n+2/3} - U^{n+1/3})/\Delta t + (U^{n+1/3} - U^n)/\Delta t = -\partial_x F(U^n) + S_Z(U^{n+1/3}) + S_f(U^{n+2/3}),$$

les termes étants pris avec des estimations différentes de U, mais tout se perd dans l'erreur Δt.

1.5 Matrice Jacobienne

On va donc commencer par examiner les équations avec $S(U) = 0$: on veut résoudre le problème sans source:

$$\partial_t U + \partial_x F(U) = 0. \tag{10}$$

On dérive la fonction $F(U)$ par rapport à U, en chaîne: $\partial_x F(U) = \partial_U F(U) \cdot \partial_x U$:

$$\partial_x F(U) = \begin{pmatrix} 0 & 1 \\ gh - \dfrac{Q^2}{h^2} & \dfrac{2Q}{h} \end{pmatrix} \cdot \partial_x \begin{pmatrix} h \\ Q \end{pmatrix} = J(U) \cdot \partial_x U,$$

L'équation devient ainsi une équation de type advection dans des vitesses qui dépendent des fonctions:

$$\partial_t U + J(U) \cdot \partial_x U = 0. \tag{11}$$

La matrice Jacobienne $J(U)$ a ici deux valeurs propres λ_1 et λ_2, le fait qu'elles existent est la définition de **l'hyperbolicité**. Si cette matrice n'a pas de valeurs propres, on ne peut pas utiliser la méthode qui suit. Le système est strictement hyperbolique pour $h > 0$ (normal!). Ces valeurs propres vont servir pour construire les solutions approchées du flux, elles ont:

$$\lambda_1 = \frac{Q}{h} - \sqrt{gh} = u - c \quad \text{and} \quad \lambda_2 = \frac{Q}{h} + \sqrt{gh} = u + c. \tag{12}$$

on reconnaît c la vitesse de propagation des ondes :

$$c = \sqrt{gh}.$$

1.6 Linéarisation des équations: pour approximer le flux

1.6.1 Linéarisation de Saint-Venant

Examinons le cas simple d'un fluide au repos sur fond plat avec une hauteur d'eau constante $u = 0$, $h = h_0$. Si on travaille sur l'équation de Saint-Venant sur fond plat sans frottements, on définit les perturbations à l'état de base $Q = 0$ et $h = h_0$ par q et η tels que $Q = 0 + q$ et $h = h_0 + \eta$ alors en posant $c_0^2 = gh_0$ et Saint-Venant Eq. 5 linéarisé :

$$\left\{ \begin{array}{l} \partial_t \eta + \partial_x q = 0 \\ \partial_t q + c_0^2 \partial_x \eta = 0 \end{array} \right. , \tag{13}$$

On veut résoudre ces équations connaissant des conditions initiales, par exemple, on connait $\eta(x,0)$ et $q(x,0)$. On peut aussi remarquer que si on se donne $q(x,0)$, on se donne aussi $\partial_x q(x,0)$ sa dérivée, mais c'est aussi se donner $\partial_t \eta = -\partial_x q$. Donc, on peut aussi dire que l'on veut résoudre ces équations connaissant $\eta(x,0)$ et $\partial_t \eta(x,0)$. C'est le "problème de Cauchy": connaissant l'EDP et ses conditions initiales, on veut résoudre pour tout temps. On note $\eta_0(x) = \eta(x,0)$, $q_0(x) = q(x,0)$ et $\eta_1(x) = \partial_t \eta(x,0)$

1.6.2 solution du "problème de Cauchy"

Les équations (13)) se mettent sous la forme d'une équation des ondes: $\partial_t^2 \eta - c_0^2 \partial_x^2 \eta = 0$. On peut aussi écrire sous forme "caractéristique" cette équation (en manipulant (13)) avec une advection vers la droite et une autre vers la gauche

$$\left\{ \begin{array}{l} \partial_t (c_0 \eta + q) + c_0 \partial_x (c_0 \eta + q) = 0 \\ \partial_t (c_0 \eta - q) - c_0 \partial_x (c_0 \eta - q) = 0 \end{array} \right. , \tag{14}$$

qui donne (les $x \pm c_0 t$ viennent bien de la solution de l'équation d'advection)

$$c_0 \eta(x,t) + q(x,t) = f(x - c_0 t) \text{ et } c_0 \eta(x,t) - q(x,t) = g(x + c_0 t)$$

La solution a la forme bien connue de somme de $f(x - c_0 t) + g(x + c_0 t)$ (le facteur $1/2c_0$ vient de la démarche et simplifie les calculs qui suivent)

$$\eta(x,t) = \frac{1}{2c_0} \left(f(x - c_0 t) + g(x + c_0 t) \right)$$

On a donc respectivement $\eta(x,t)$ et aussi par dérivation directe en temps $\partial_t \eta(x,t)$:

$$2c_0 \eta(x,t) = f(x - c_0 t) + g(x + c_0 t) \text{ resp. } \; 2c_0 \partial_t \eta(x,t) = -c_0 f'(x - c_0 t) + c_0 g'(x + c_0 t)$$

donc si on se donne à $t = 0$ la distribution $\eta(x,t) = \eta_0(x)$ et sa dérivée $\partial_t \eta(x,0) = \eta_1(x)$, alors

$$2c_0 \eta_0(x) = f(x) + g(x) \;\; \text{ et } 2c_0 \eta_1(x) = -c_0 f'(x) + c_0 g'(x)$$

on en déduit que $f(x) - g(x) = -2 \int_0^x \eta_1(\xi) d\xi + K$, constante dont on peut se passer. On en déduit donc les expressions des fonctions f et g en fonction de η_0 et η_1:

$$f(x) = c_0 \eta_0(x) - \int_0^x \eta_1(\xi) d\xi \text{ et } g(x) = c_0 \eta_0(x) + \int_0^x \eta_1(\xi) d\xi$$

et en déplaçant de $c_0 t$ en moins et en plus (en remarquant que $-\int_0^{x-c_0t} \eta_1(\xi)d\xi = +\int_{x-c_0t}^{0} \eta_1(\xi)d\xi$):

$$f(x-c_0t) = c_0\eta_0(x-c_0t) + \int_{x-c_0t}^{0} \eta_1(\xi)d\xi \text{ et } g(x+c_0t) = c_0\eta_0(x+c_0t) + \int_{0}^{x+c_0t} \eta_1(\xi)d\xi$$

la somme nous donne la relation (qui est la solution du "problème de Cauchy")

$$\eta(x,t) = \frac{\eta_0(x-c_0t)+\eta_0(x+c_0t))}{2} + \frac{1}{2c_0}\int_{x-c_0t}^{x+c_0t} \eta_1(\xi)d\xi.$$

Ce que nous venons d'écrire est bien la solution pour tout temps de l'équation aux dérivées partielles de départ avec la donnée de la solution $\eta(x,t=0) = \eta_0(x)$ et de sa dérivée en temps $\partial_t\eta(x,0) = \eta_1(x)$ au temps initial:

$$\eta(x,t) = \frac{\eta_0(x-c_0t)+\eta_0(x+c_0t)}{2} + \frac{1}{2c_0}\int_{x-c_0t}^{x+c_0t} \partial_t\eta(\xi,0)d\xi.$$

Ce premier résultat important étant établi, repassons aux deux variables η et q, comme par définition $\partial_t\eta = -\partial_x q$ donc $\eta_1(x)$ est disons un $-\partial_x q_0(x)$ donné initialement:

$$\eta(x,t) = \frac{\eta_0(x-c_0t)+\eta_0(x+c_0t))}{2} - \frac{1}{2c_0}[q_0(x+c_0t) - q_0(x-c_0t)].$$

Cette expression nous donne la valeur de $\eta(x,t)$ connaissant les champs initiaux η_0 et q_0.

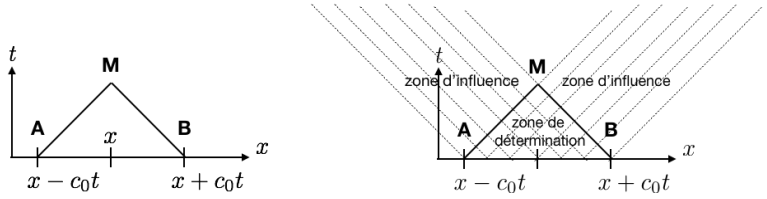


Figure 1: A gauche, la solution au point M en x et $t > 0$ est déterminée par une partie des données, en l'occurence η_0 en A et en B et $\eta_1 = \partial_t\eta_0$ sur le segment AB. A gauche on voit les zones influencées et déterminées par les données η_0 et et η_1 dans le temps futur $t > 0$

1.6.3 Construction d'un flux

Cette construction étant faite, ensuite quand on résout par exemple

$$\partial_t q + \partial_x(F) = 0, \text{ avec } \; F = c_0^2\eta$$

on a l'expression du flux F obtenue avec l'analyse précédente pour $\eta(x,t)$ avec les valeurs en $(\pm c_0 t)$:

$$F(x,t) = \frac{c_0^2\eta_0(x-c_0t)+c_0^2\eta_0(x+c_0t))}{2} - \frac{c_0}{2}[q_0(x+c_0t) - q_0(x-c_0t)],$$

la forme du flux en fonction des quantités conservées le long de $x \pm c_0 t$ et de $F = c_0^2 \eta_0$ est donc:

$$F(x,t) = \frac{F(x - c_0 t) + F(x + c_0 t))}{2} - \frac{c_0}{2}[q_0(x + c_0 t) - q_0(x - c_0 t)], \tag{15}$$

Cette forme nous servira pour construire des approximations du flux aux faces de nos domaines. Gardons la en tête, mais auparavant rappelons les volumes finis.

Nous retrouverons ce résultat et d'autres circonvolutions autour de l'équation des ondes dans
`http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf`

2 Résolution numérique en Volumes Finis

2.1 Splitting: étape convective

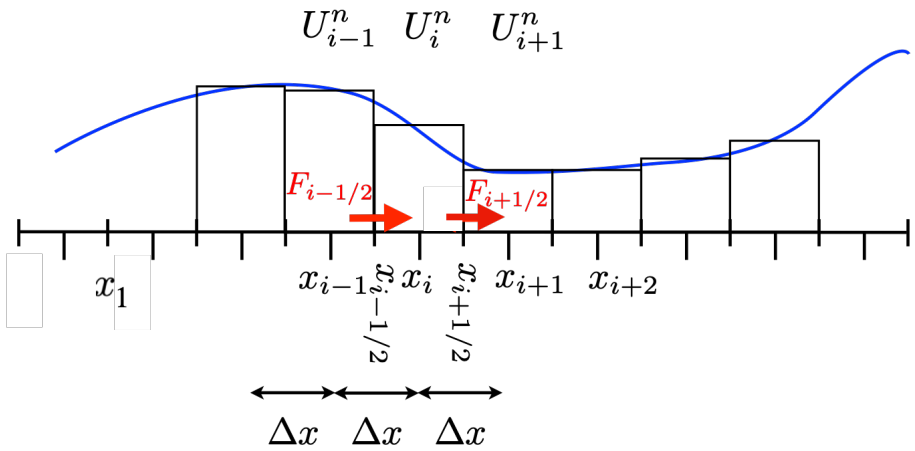


Figure 2: Le signal continu U, approximé par U_i^n en x_i. Le flux aux faces du petit volume centré en x_i et de longueur Δx est $F_{i-1/2}$ à gauche et $F_{i+1/2}$ à droite.

L'espace est découpé en N petits segments de longueur quelconque *a priori*, mais ici nous supposerons un pas Δx constant ($\Delta x = L/N$, voir figures 2 et aussi 11) pour alléger les notations. De même pour le temps, le pas de temps Δt sera supposé constant, assez petit pour assurer la stabilité. Ces petits segments de longueur Δx, sont les petits "volumes", car nous sommes en dimension un. Prenons le système (7) (sans source $S(U) = 0$) et intégrons à la fois en x sur un intervalle entre $x_{i-1/2}$, et $x_{i+1/2}$ centré en x_i (c'est une intégration sur le "segment/ Volume"). L'intervalle étant fixe on peut permuter et donc :

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} = 0, \text{ est intégré } \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial}{\partial t} U(x,t)dx + \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial F(U)}{\partial x} = 0$$

$$\text{Le domaine est fixe et par définition du flux } \frac{\partial}{\partial t} \int_{x_{i-1/2}}^{x_{i+1/2}} U(x,t)dx + F_{i+1/2} - F_{i-1/2} = 0$$

De manière différente de celle des "différences finies", l'approximation de U est une moyenne *average state in an interval* Roe 1986 [15]. On pose U_i^n une approximation de U autour de la position x_i (entre $x_i - \Delta x/2$ et $x_i + \Delta x/2$, notés $x_{i-1/2}, x_{i+1/2}$), ou même plus clairement U_i^n est la valeur moyenne au temps discret t_n:

$$U_i^n = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} U(x,t_n)dx,$$

où i se rapporte au segment $C_i = (x_{i-1/2}, x_{i+1/2}) = (x_{i-1/2}, x_{i-1/2} + \Delta x)$, n se rapporte au temps discrétisé t_n avec $t_{n+1} - t_n = \Delta t$. Le premier terme (dérivée en temps) s'écrit par développement de Taylor de U:

$$U(x, t + \Delta t) = U(x,t) + \Delta t \partial_t U + (1/2)(\Delta t)^2 \partial_t^2 U + O(\Delta t)^3,$$

l'avancée en temps est telle que le temps discret courant est t_n, le temps suivant t_{n+1} est $t_{n+1} = t_n + \Delta t$, donc

$$\frac{\partial}{\partial t}\int_{x_{i-1/2}}^{x_{i+1/2}}(U)dx \simeq \frac{\int_{x_{i-1/2}}^{x_{i+1/2}} U(x,t+\Delta t)dx - \int_{x_{i-1/2}}^{x_{i+1/2}} U(x,t)dx}{\Delta t} = \frac{U_i^{n+1}-U_i^n}{\Delta t}$$

pour le second terme (divergence du flux):

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x F(U)dx = F_{i+1/2}^n - F_{i-1/2}^n.$$

Cette intégration "exacte" du flux sur le "volume" est toute l'essence de la méthode.

Le flux numérique $F_{i+1/2}$ est une approximation de la fonction flux (8) à l'interface droite du segment C_i en $i+1/2$. Ce flux est, on suppose, une fonction de la valeur U_i dans le segment considéré centré en i et de la valeur U_{i+1} dans le segment suivant centré en $i+1$:

$$F_{i+1/2} = \mathcal{F}(U_i, U_{i+1}). \tag{16}$$

Attention à ne pas confondre la fonction $\mathcal{F}$ des deux variables de part et d'autre de l'interface, avec le flux numérique à travers la surface $F_{i+1/2}$ et F le flux physique. Attention aussi, dans les codes numériques, il n'y a pas de demi indices, donc on utilisera `F[i+1]` pour désigner le flux numérique $F_{i+1/2}$.

On obtient le schéma de volumes finis au premier ordre (voir la figure 2) que l'on écrit sous la forme suivante (LeVeque 02 page 65 eq (4.4))):

$$\boxed{\frac{U_i^{n+1}-U_i^n}{\Delta t} + \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = 0,} \tag{17}$$

Le schéma est explicite: on calcule les nouvelles valeurs U_i^{n+1} en fonction des anciennes valeurs U_i^n. Dans la suite de ce document nous implémentons en C ces équations: `http://basilisk.fr/sandbox/M1EMN/BASIC/advecte1c.c`
On peut aussi les mettre dans *Basilisk* `http://basilisk.fr/sandbox/M1EMN/BASIC/advecte1.c`

2.2 Consistance, stabilité, convergence: "théorème de Lax"

Il s'agit du théorème de Lax, fondamental pour la bonne résolution des EDP.

La méthode doit être "consistante" avec l'équation différentielle aux dérivées partielle que l'on étudie, ce qui veut dire que la méthode est une bonne approximation de l'équation de départ.

La méthode doit être "stable", ce qui veut dire que les petites erreurs ne sont pas amplifiées avec les itérations temporelles successives. On introduira la condition CFL (Courant, Friedrichs, Lewy, (1928), "Über die partiellen Differenzengleichungen der mathematischen Physik", [5]).

Le théorème de Lax stipule que lorsque l'on résout numériquement un problème évolutif (supposé bien posé, donc avec la bonne condition initiale) avec un schéma numérique "consistant", la "stabilité" du schéma est une condition nécessaire et suffisante pour assurer la convergence. Lax, Richtmyer [14] 1956.

Ce théorème est un peu le fondement mathématique de l'approche numérique que nous adoptons, remarquons qu'il date de 1928. Ce théorème justifie les calculs que nous faisons, c'est un garde fou. En fait, dans le cas de volumes finis stricts, il faut adapter ce théorème à la convergence des

flux et le sujet est encore en évolution ("Analyse des Volumes Finis" Eymard et al. (2020) MatApli 126).

Malgré l'invocation du mot "théorème", la suite de ces notes est très empirique. On consultera donc les cours des collègues et la littérature sur le sujet pour avoir les vrais détails et non les allusions de plus en plus rapides à partir d'ici.

2.3 Flux Numérique

2.3.1 Flux instable et sa stabilisation

Il faut maintenant trouver la bonne approximation de $\mathcal{F}$. Le plus simple serait de prendre la moyenne des valeurs (LeVeque [13] chapitre 4, Finite Volume methods, voir aussi Roe [15])

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} \tag{18}$$

d'où

$$U_i^{n+1} = U_i^n - \Delta t \frac{F(U_{i+1}) - F(U_{i-1})}{2\Delta x} \tag{19}$$

mais c'est une mauvaise idée... car la méthode est instable. On peut "bricoler" et stabiliser: on calcule la première composante de manière explicite, et en utilisant pour la seconde composante cette approximation, on peut avoir un schéma stable (c'est un schéma naïf, cf Euvrard [12]). On va voir une autre technique classique pour stabiliser qui est la méthode de "Lax-Friedrichs". Mais ensuite nous passerons à une méthode plus efficace (flux dépendants des valeurs propres du sytème, Rusanov et HLL).

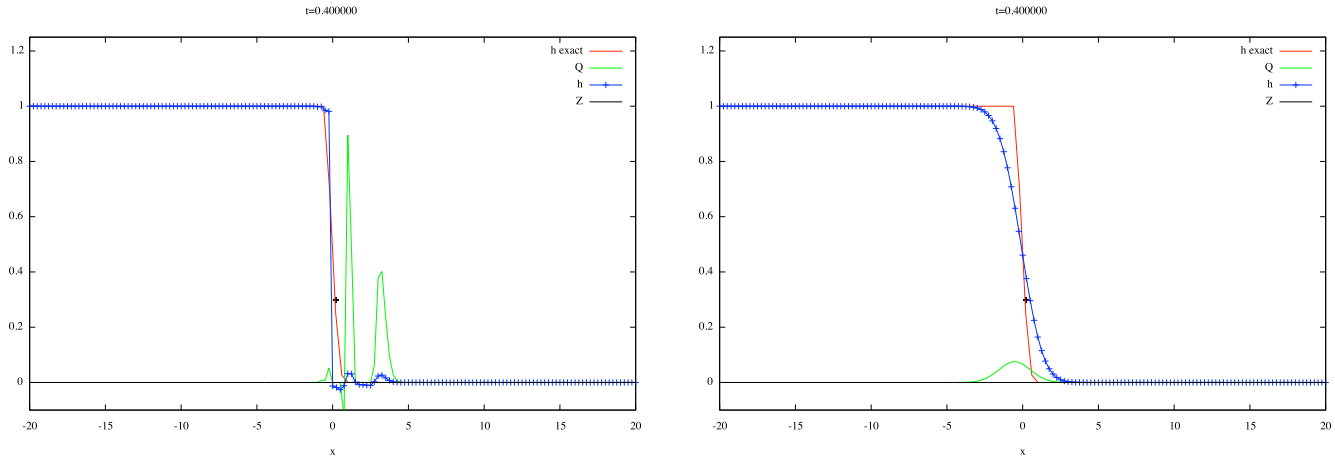


Figure 3: Départ de rupture de barrage au temps sans dimension $t = 0.4$, la solution exacte $h(x, t = 0.4$ est tracée en rouge. A gauche, cas simple du flux $\mathcal{F}(U_{i-1}, U_i) = (F(U_{i-1}) + F(U_i))/2$, on voit des oscillations amplifiées pour Q en vert et h en bleu calculés au même instant. Le schéma simple centré est bien instable. A droite, le schéma de Lax au même instant est stabilisé. En rouge, la solution exacte, on voit que la solution de Lax est plus diffusée qu'il ne faut (c'est normal, car cette méthode introduit des termes de dissipation). La figure 9 montrera une comparaison Lax / Rusanov, ce dernier étant bien meilleur.

justification du flux dans le cas linéaire simple

On a vu à la sous section précédente que

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2}$$

n'est pas un bon choix, nous discutons ici de la forme corrigée

$$\mathcal{F}(U_{i-1},U_i)=\frac{F(U_{i-1})+F(U_i)}{2}-c_\Delta\frac{((U_i)-(U_{i-1}))}{2}\text{ avec la donnée de }c_\Delta$$

Pour l'équation linéarisée de Saint-Venant

$$\left\{\begin{array}{l}\partial_t\eta+\partial_x q=0\\ \partial_t q+c_0^2\partial_x\eta=0\end{array}\right.,\tag{20}$$

Rappelons que l'on a vu dans le cas de l'équation de ∂'Alembert que l'on a exactement pour l'équation: $\partial_t q+\partial_x(F)=0$, la forme du flux en fonction des quantités conservées le long de $x\pm c_0t$ a été vue en équation 15:

$$F=\frac{F(x-c_0t)+F(x+c_0t))}{2}-\frac{c_0}{2}[q_0(x+c_0t)-q_0(x-c_0t)],$$

donc la discrétisation en x avec des cases en $x-\Delta x/2$ et $x+\Delta x/2$, équivalents au $(x\pm c_0t)$. On va donc chercher l'information de part et d'autre de la face pour construire le flux

$$\mathcal{F}(U_{i-1},U_i)=\frac{F(U_{i-1})+F(U_i)}{2}-c_\Delta\frac{((U_i)-(U_{i-1}))}{2}\text{ avec ici }c_\Delta=c_0\tag{21}$$

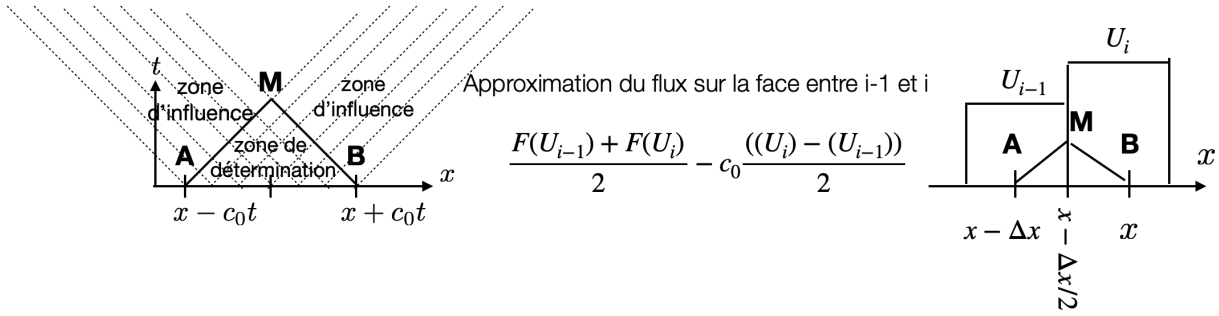


Figure 4: Définition du flux $\mathcal{F}(U_{i-1},U_i)=\frac{(F(U_{i-1})+F(U_i))}{2}-c_0\frac{((U_i)-(U_{i-1}))}{2}$, à la face $i-1/2$ entre les positions des volumes $i-1$ et i inspiré de la notion de zone de détermination.

Autre justification du flux dans le cas 1D

On peut dire que $F_{i-1/2}=\frac{F(U_{i-1})+F(U_i)}{2}$ ne convient pas. Il nous faut en trouver un meilleur dont la structure soit quand même assez proche. La figure 5 nous donne une indication possible, considérons uniquement une face séparant deux familles de cellules de valeurs constantes, soit $i-1/2$ la position de cette face. A gauche on U_G et à droite on a U_D, Nous connaissons la solution exacte dans ce cas: c'est le choc qui se propage, à la vitesse $c=[|F|]/[|U|]$ avec c la vitesse de la discontinuité.

Donc, au temps Δt la solution exacte est $U=U_G$ pour $x<x_{i-1/2}+c\Delta t$ et $U=U_D$ pour $x>x_{i-1/2}+c\Delta t$ avec $c=(F(U_G)-F(U_D))/(U_G-U_D)$ (vitesse de discontinuité Eq. 4). La valeur de U^* (au sens de moyenne des volumes finis) dans la cellule entre cette face $(x_{i-1/2})$ et la face suivante en $(x_{i+1/2}=x_{i-1/2}+\Delta x)$ est une valeur intermédiaire entre U_G et U_D compte tenu de l'avancée du front, on obtient $U^*=U_D+(U_G-U_D)c\Delta t/\Delta x$.

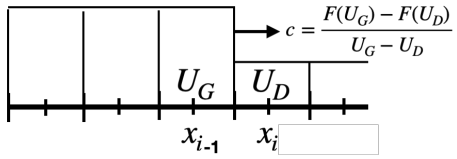


Figure 5: Une discontinuité se déplace de gauche à droite; si $x < x_{i-1/2}$ U_i est constant et vaut U_G, sinon $U_i = U_D$, le choc se déplace à la vitesse $c = (F(U_G) - F(U_D))/(U_G - U_D)$.

La variation temporelle pendant le temps Δt est $(U^* - U_D)/\Delta t = (U_G - U_D)c/\Delta x$. Ce doit être la différence de flux $-(F_{i+1/2} - F_{i-1/2})/\Delta x$. Vérifions le, on a : $F_{i-1/2} = \frac{F(U_G)+F(U_D)}{2} - c\frac{U_D-U_G}{2}$ et on a $F_{i+1/2} = \frac{F(U_D)+F(U_D)}{2} - c\frac{U_D-U_D}{2} = F(U_D)$ simplement et

$$-\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = \frac{F(U_G) - F(U_D)}{2} - c\frac{U_D - U_G}{2} \text{ or } F(U_G) - F(U_D) = c(U_G - U_D) \text{ donc } -\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} = c\frac{U_G - U_D}{\Delta x}$$

c'est effectivement le $(U^* - U_D)/\Delta t = (U_G - U_D)c/\Delta x$ obtenu. Donc avec le choix

$$F_{i-1/2} = \frac{F(U_{i-1}) + F(U_i)}{2} - c\frac{((U_i) - (U_{i-1}))}{2}$$

on retrouve donc bien la solution exacte de propagation du choc.

Conclusion

Le flux de la forme

$$\boxed{F_{i-1/2} = \frac{F(U_{i-1}) + F(U_i)}{2} - c_\Delta\frac{((U_i) - (U_{i-1}))}{2}}$$

va être utilisé par la suite en choisissant le ou les "bons" c_Δ.

C'est Roe dans les années 1980 [15] qui a proposé cette forme en la reliant aux vecteurs propres et valeurs prropres de la matrice du problème hyperbolique.

2.3.2 Flux trop stable de Lax-Friedrichs

On a vu à la sous section précédente que

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2}$$

n'est pas un bon choix, écrivons maintenant comme suggéré dans les points précédents une forme corrigée avec un c_Δ que nous choisissons égal à $\frac{\Delta x}{\Delta t}$ (qui est un choix possible parmi les nombreux autres...):

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} - c_\Delta\frac{((U_i) - (U_{i-1}))}{2} \text{ avec } c_\Delta = \frac{\Delta x}{\Delta t} \tag{22}$$

et mettons ce flux dans le schéma en volumes finis

$$U_i^{n+1} = U_i^n - \Delta t \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = U_i^n - \Delta t \frac{\mathcal{F}(U_i^n, U_{i+1}) - \mathcal{F}(U_{i-1}^n, U_i^n)}{\Delta x}$$

donc en substituant cette expression

$$U_i^{n+1} = U_i^n - \Delta t \frac{\frac{F(U_i)+F(U_{i+1})}{2} - c_\Delta \frac{((U_{i+1})-(U_i))}{2} - \frac{F(U_{i-1})+F(U_i)}{2} + c_\Delta \frac{((U_i)-(U_{i-1}))}{2}}{\Delta x}$$

en simplifiant (comme $c_\Delta \Delta t/\Delta x = 1$) on obtient tout simplement avec le flux discret (Eq. 22) le schéma dit de "Lax-Friedrichs" (proposé par Lax en 1954)

$$U_i^{n+1} = \frac{1}{2}(U_{i+1}^n + U_i^n) - \frac{\Delta t}{2\Delta x}(F(U_{i+1}) - F(U_{i-1}))$$

En définitive, le flux de Lax-Friedrichs s'écrit bien de la forme suivante (Toro p 329):

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1})+F(U_i)}{2} - c_\Delta \frac{((U_i)-(U_{i-1}))}{2} \text{ avec } c_\Delta = \frac{\Delta x}{\Delta t}$$

Habituellement on définit le schéma de Lax-Friedrichs en disant que l'on remplace U_i^n dans (Eq. 19) par une moyenne $(U_{i+1}^n + U_{i-1}^n)/2$, pour obtenir le schéma de Lax-Friedrichs

$$U_i^{n+1} = \frac{1}{2}(U_{i+1}^n + U_{i-1}^n) - \frac{\Delta t}{2\Delta x}(F(U_{i+1}) - F(U_{i-1})).$$

LF a le mérite d'introduire une diffusion numérique qui tue l'instabilité, en effet en repassant au développement de Taylor de la demi somme et de l'accroissement :

$$\frac{1}{2}(U_{i+1}^n + U_{i-1}^n) = U_i^n + \frac{\Delta x^2}{2}\partial_x^2 U^n + ... \text{ et } -\frac{\Delta t}{2\Delta x}(F(U_{i+1}) - F(U_{i-1})) = -\Delta t \partial_x F(U) + ...$$

donc le schéma de Lax-Friedrichs devient

$$U_i^{n+1} = U_i^n + \frac{\Delta x^2}{2}\partial_x^2 U^n + ... - \Delta t \partial_x F(U) + ...$$

et re remplaçant $U_i^{n+1} = U_i^n + \Delta t \partial_t U + ...$ on obtient que le schéma de Lax -Friedrichs est en fait la résolution de l'équation de départ mais avec un terme de diffusion en $\frac{\Delta x^2}{2\Delta t}$ pour l'équation modifiée:

$$\partial_t U + \partial_x F(U) = \frac{\Delta x^2}{2\Delta t}\partial_x^2 U.$$

La solution diffuse avec un coefficient de viscosité $\frac{\Delta x^2}{2\Delta t}$, ce n'est pas toujours très bon si on veut être proche d'une solution avec de fortes variations en x...

Ceci étant rappelé, on va continuer à voir que la forme

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1})+F(U_i)}{2} - c_\Delta \frac{((U_i)-(U_{i-1}))}{2} \tag{23}$$

est presque la bonne si on prend un ou des c_Δ mieux adaptés. Plusieurs flux, peuvent être employés, nous utiliserons les plus simples.

2.4 Cas simple $U = h$, $F = ah$, $a = 1$

Pour fixer les idées, on va appliquer cela dans le cas d'une équation, ce qui arrive lorsque l'on fait des ondes cinématiques (6). L'équation d'advection simple s'obtient en linéarisant (6). On lui applique les flux discrets précédents (18), (22) et (21). Soit l'équation d'advection à a constant

$$\partial_t h + \partial_x(ah) = 0,$$

la valeur propre est a tout simplement.

La solution de $\partial_t h + \partial_x(ah) = 0$, avec $h(x,0) = h_0(x)$ est bien sûr $h(x,t) = h_0(x - at)$.

• Si on utilise le flux simple (equivalent de 18)

$$F_{i-1/2} = a\frac{h_{i-1} + h_i}{2} \tag{24}$$

d'où

$$h_i^{n+1} = h_i^n - \Delta t\frac{a(h_{i+1} - h_{i-1})}{2\Delta x} \tag{25}$$

la méthode est instable, on peut le vérifier avec la méthode de stabilité de Neumann en écrivant $h = h_n e^{ikx}$ et $h_{n+1} = Gh_n$ d'où le gain $G = 1 - ia\sin(k\Delta x)$, toujours plus grand que 1 en module, donc si on part d'une perturbation sinusoidale, elle est amplifiée d'une étape à l'autre.

• Si on utilise le flux de Lax-Friedrichs (equivalent de 22):

$$F_{i-1/2} = a\frac{h_{i-1} + h_i}{2} - c_\Delta(\frac{h_i - h_{i-1}}{2}) \tag{26}$$

avec $c_\Delta = \dfrac{\Delta x}{\Delta t}$ on a stabilisé par diffusion pour $\Delta t < \Delta x/a$ (condition CFL). On peut calculer le gain $G = \cos(k\Delta x) - ia\sin(k\Delta x)$. Si on développe le flux, on obtient

$$h_i^{n+1} = h_i^n - \Delta t\frac{a(h_{i+1} - h_{i-1})}{2\Delta x} + \Delta t(c_\Delta\Delta x)\frac{(h_{i+1} - 2h_i + h_{i-1})}{2\Delta x^2} \tag{27}$$

la méthode est stable mais diffusive car elle correspond à l'équation

$$\partial_t h + \partial_x(ah) = (ac_\Delta\Delta x/2)\partial_x^2(h)$$

• A titre d'exercice on retrouvera les flux de Lax-Wendroff, Beam Warning etc.

• Si maintenant on utilise le flux (21), on prend un autre choix de c_Δ: ici $c_\Delta = a$, c'est le flux *upwind*:

$$F_{i-1/2} = \frac{ah_{i-1} + ah_i}{2} - (\frac{a}{2})(h_i - h_{i-1}) \tag{28}$$

d'où après simplification, cette expression devient simplement

$$F_{i-1/2} = ah_{i-1}$$

et le nouvel h se calcule par

$$h_i^{n+1} = h_i^n - \Delta t \frac{a(h_i - h_{i-1})}{\Delta x} \tag{29}$$

on peut montrer que la méthode est stable, (on pourrait montrer que le flux downwind $F_{i-1/2} = ah_i$ est instable.

On remarque que cette construction est exactement la même que celle développée plus haut pour le déplacement d'un choc, la valeur en $n+1$ est une extrapolation par un retour en arrière de la caractéristique, pour trouver son intersection dans l'intervalle $(i, i-1)$ puis en faisant une interpolation linéaire pour trouver la valeur de h sur cette caractéristique $h_i^{n+1} = h_i^n + (h_i^n - h_{i-1}^n)a\frac{\Delta t}{\Delta x}$

2.5 Mise en œuvre pratique en C du cas simple $U = h$, $F = h$

Cette méthode simple est codée en C dans `http://basilisk.fr/sandbox/M1EMN/BASIC/advecte1c.c` et aussi en *Basilisk* dans `http://basilisk.fr/sandbox/M1EMN/BASIC/advecte1.c`

Les lignes qui suivent sont le code C dans `http://basilisk.fr/sandbox/M1EMN/BASIC/advecte1c.c`
On y résout

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} = 0$$

avec $F = U$, et $U(x, 0) = e^{-x^2}$.

La solution analytique de cette équation de transport, ou équation d'advection est $U(x, t) = e^{-(x-t)^2}$. On compare la solution numérique avec cette solution exacte.

On explique ici les différentes instructions. Le code C commence par des déclarations obligatoires

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
```

puis les déclarations des tableaux (pointeurs, avec une étoile devant) et les variables

```
double*x=NULL,*U=NULL,*F=NULL;
double dt;
double cDelta;
double L0,X0,Delta;
double t;
int i,it=0;
int N;
```

il y a ensuite le `main`, le "principal" avec l'affectation des valeurs des variables, et l'affectation mémoire des pointeurs `tcalloc` (allocation dynamique)

```
int main() {
  L0 = 12.;
  X0 = -L0/4;
  N = 128;
  t=0;
  dt = (L0/N)/2;
  Delta = L0/N;
```

```
  x= (double*)calloc(N+2,sizeof(double));
  U= (double*)calloc(N+2,sizeof(double));
  F= (double*)calloc(N+2,sizeof(double));
```

On fait une boucle en i pour remplir les tableaux précédents pour la valeur initiale en temps

```
    for(i=0;i<N;i++)
    {   x[i]=X0+(i-1./2)*Delta;
        U[i] = exp(-x[i]*x[i]);
  }
```

Ensuite on commence une boucle en temps avec `while`, au début de chaque itération on sort les valeurs de x, U et t dans le terminal

```
   while(t<=3){
     t = t + dt;
     it++;

if( (it == (int)(1/dt)) || (it == (int)(2/dt)) || (it == (int)(3/dt))  ){
 for(i=0;i<N;i++)
    fprintf (stdout, "%g %g %g  \n", x[i], U[i], t);
  fprintf (stdout, "\n\n");
}
```

Puis on calcule le flux

$$F_{i-1/2} = \frac{F(U_{i-1}) + F(U_i)}{2} - c_\Delta \frac{((U_i) - (U_{i-1}))}{2}$$

(i est $i - 1/2$ car les indices de tableaux sont entiers)

```
 for(i=1;i<=N+1;i++)
   {
    cDelta = 1;
    F[i] = (U[i]+U[i-1])/2.  - cDelta *(U[i]-U[i-1])/2;
    }
```

On fait l'update $U_i^{n+1} = U_i^n - \Delta t \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x}$ et pour mémoire, on traite les conditions aux limites par du Neumann, la dernière accolade } correspond à la fin de la boucle d'itération en temps

```
 for(i=0;i<=N;i++)
    U[i] +=  - dt* ( F[i+1] - F[i] )/Delta;

 U[0] = U[1];
 U[N+1]=U[N];
}
```

Pas indispensable, mais associé au `calloc` on libère la mémoire pour sortir du `main` avec la dernière accolade }.

```
  free(U);
  free(F);
  free(x);
}
```

Compilation

Ce fichier est écrit dans un fichier texte ASCII avec l'extension .c, par exemple `advecte1c.c` . On ouvre un terminal dans le dossier où le fichier a été sauvé, on compile avec la ligne de commande:

`cc -lm advecte1c.c -o advecte1c`

cela crée un exécutable.

Execution
On lance l'exécutable dans ce même terminal en redirigeant la sortie du terminal dans le fichier `out`:
`./advecte1c > out`

Tracé
On ouvre un AUTRE onglet ou terminal, on va dans le même dossier, on y lance `gnuplot`, puis on peut tracer en tapant simplement `p'out'` . Si on veut un dessin avec la comparaison à la solution analytique:

```
set xlabel "x"
U(x,t)= exp(-(x-t)*(x-t))
p'out' u (1) : (2)t'num'w p,'' u 1:(U(1,3)) t'exact' w l
```

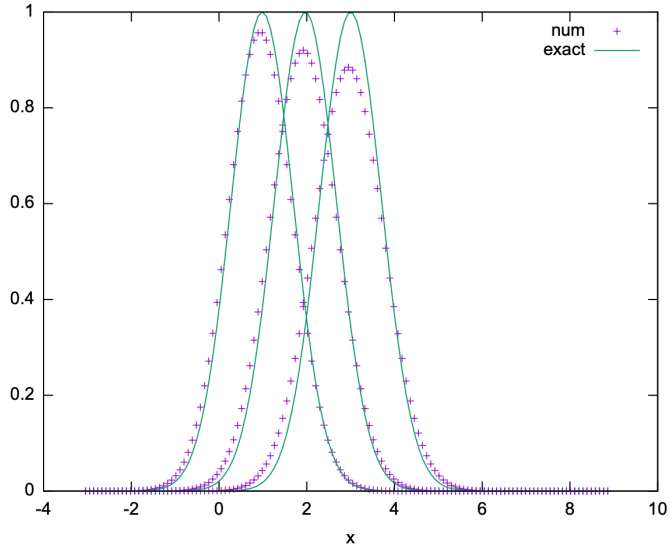


Figure 6: Évolution du signal transporté, comparaison à la solution analytique et numérique, on note que l'on a pas assez de précision car le signal se dissipe

Pour maîtriser `gnuplot`, on consultera `http://basilisk.fr/sandbox/M1EMN/BASIC/gnuplot_examples.c` et *Google is my friend.*

2.6 Cas moins simple *flood wave*: $U = h$, $F = h^{3/2}$

A titre d'exercice, on pourra appliquer cela dans le cas de l'équation d'advection de type "onde de crue" (voir `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf`):

$$\partial_t h + \partial_x(h^{3/2}) = 0,$$

et la condition initiale $h(x,0) = 1/2 + e^{-x^2}$. La solution analytique se fait par les caractéristiques. Il y a apparition d'un choc.
`http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf`.

Pour calculer le flux numérique discret, la vitesse c_Δ à considérer sera liée à $c = \partial_h(h^{3/2})$, donc ici on trouve $3/2\sqrt{h}$. On va coder sur la fazce la moyenne de part et d'autre, et le flux:

$$c_{i-1/2} = 3/2(\sqrt{h_{i-1}} + \sqrt{h_i})/2, \quad F_{i-1/2} = \frac{(h_{i-1}\sqrt{h_{i-1}} + h_i\sqrt{h_{i-1}})}{2} - c_{i-1/2}\frac{(h_i - h_{i-1})}{2};$$

et ainsi le nouvel h au temps suivant est :

$$h_i^{n+1} = h_i^n - \Delta t \frac{F_{i+1/2} - F_{i-1/2}}{\Delta x}$$

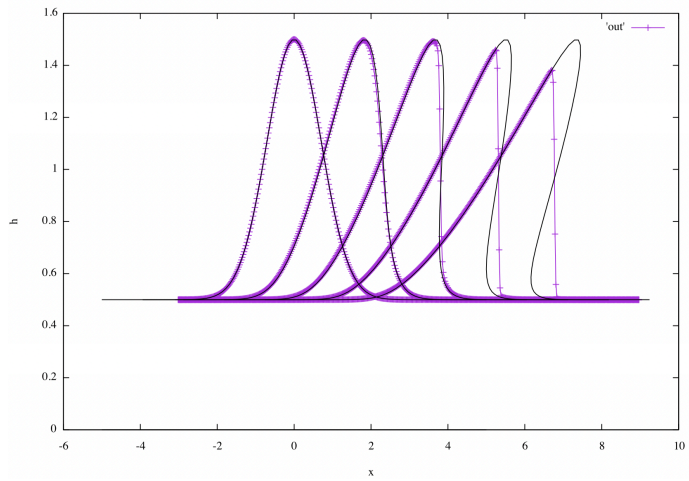


Figure 7: Évolution d'une intumescence par une onde de crue, comparaison à la solution analytique par les caractéristiques. Apparition du choc.

Lorsque l'on résout avec le code ci après, en partant d'une bosse on voit (figure 7) que la bosse avance (elle est transportée), mais on observe un choc. La solution en caractéristiques est superposée.

voir le code C dans
`http://basilisk.fr/sandbox/M1EMN/Exemples/floodwaveC.c`

voir le code python
`http://basilisk.fr/sandbox/M1EMN/PYTHON/flood.py`

voir le code *Basilisk* dans
`http://basilisk.fr/sandbox/M1EMN/Exemples/floodwave.c`

Le code est retranscrit ici, on y reconnait les étapes précédentes. On a codé une fonction `FR1` qui renvoie la valeur du flux compte tenu des paramètres qui sont les valeurs à gauche et à droite de la face.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

double*x=NULL,*h=NULL,*F=NULL;
double dt;
double cDelta;
double L0,X0,Delta;
double t;
int i,it=0;
int N;

double FR1(double hg, double hd){
    double c=1.5*(sqrt(hg)+sqrt(hd))/2;
    return (hg*sqrt(hg)+hd*sqrt(hd))*0.5-c*(hd-hg)*0.5;}


int main() {
  L0 = 12.;
  X0 = -L0/4;
  N = 128*2*2;
  t=0;
  dt = (L0/N)/2;
  Delta = L0/N;

  x= (double*)calloc(N+2,sizeof(double));
  h= (double*)calloc(N+2,sizeof(double));
  F= (double*)calloc(N+2,sizeof(double));

   for(i=0;i<N+2;i++)
    {  x[i]=X0+(i-1./2)*Delta;
       h[i] = 0.5+exp(-x[i]*x[i]);
  }

   while(t<=4){
     t = t + dt;
     it++;

//static  FILE * fp = fopen ("out.txt", "w");

if((it == 1)||(it == (int)(1/dt))
   || (it == (int)(2/dt))
   || (it == (int)(3/dt))
   || (it == (int)(4/dt)) ){
//for(i=0;i<=N;i++)
//     fprintf (fp, "%g %g %g  \n", x[i], h[i], t);
//     fprintf (fp, "\n\n");
//      fclose(fp);

 for(i=0;i<=N;i++)
    fprintf (stdout, "%g %g %g  \n", x[i], h[i], t);
   fprintf (stdout, "\n\n");
}

 for(i=1;i<=N+1;i++)
    F[i] = FR1(h[i-1],h[i]);

  for(i=1;i<=N;i++)
```

```
    h[i] += - dt*( F[i+1] - F[i] )/Delta;

   h[0] = 0.5;
   h[N+1]=h[N];
 }

  free(h);
  free(F);
  free(x);
}
/*
Then compile and run:
 cc  floodwaveC.c -lm
 ./a.out> out
 then  gnuplot
 reset
 set xlabel "x"
 set ylabel "h"
 h(x)=.5+exp(-x*x)
 c(x)=3./2*sqrt(h(x))
 set parametric
 set dummy x

  p[:][:] 'out' w l , x,h(x) not w l lc -1,\
  x+1*c(x),h(x) not w l lc -1,\
  x+2*c(x),h(x) not w l lc -1,\
  x+3*c(x),h(x) not w l lc -1,\
  x+4*c(x),h(x) not w l lc -1,\
  x ,0 not w l lc -1
 which gives $h(x,t)$ plotted here for t=0 1 2 3  4 and $-3<x<9$
```

Cet exemple nous montre bien la mise en œuvre sur des cas à une équation 1D du type $\partial_t h + \partial_x Q(h) = 0$. On va exploiter avec deux équations la même idée de choisir la vitesse c_Δ de manière à rendre le schéma stable et de manière à être dans le bon sens (up ou down) suivant le signe de la vitesse.

2.7 Flux en pratique, Rusanov et HLL

2.7.1 Flux

Nous venons de voir l'importance de bien écrire le flux pour bien résoudre un problème hyperbolique. La construction générale des flux est une tache ardue, on consultera les livres de Bouchut [4] de Toro [16] ou de Randall LeVEque [13] ou Roe [**?**]pour plus de détails; il existe ainsi de nombreux flux ayant les bonnes propriétés, et la Recherche continue sur ce sujet de Mathématiques Appliquées. Citons le flux cinétique, le flux de Kurganov, celui de Rusanov, celui de HLLC.... par exemple dans *Basilisk*: `http://basilisk.fr/src/riemann.h` kinetik, Kurganov, et HLLC sont codés. D'autres méthodes existent, et au final tout le monde est à peu près équivalent, dans un cas proche, celui de l'écoulement du sang dans les artères, nous l'avons vérifié `http://www.lmm.jussieu.fr/~lagree/TEXTES/PDF/wang14.pdf` en comparant Mac-Cormack (prédicteur correcteur avec Lax), Taylor-Galerkin (éléments finis), Volumes Finis, et Discontinuous Galerkin (un mix des deux précédents). Tous les schémas donnent dans les cas doux le même résultat, sauf dans les cas sévères, ou leurs défauts sont exacerbés, trop ou pas assez de diffusion, dispersion, etc....

2.7.2 Flux de Rusanov

Le flux le plus simple ayant les bonnes propriétés est le flux de Rusanov [4], on voit qu'il s'obtient directement à partir de nos observation pour l'équation d'onde ou de l'advection 1D.

(On va voir à la suite qu'il s'obtient à partir de HLL en mettant $c_1 = -c_2 = -c$).

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} - c\frac{U_i - U_{i-1}}{2},$$

ou tel qu'il sera codé

$$\mathcal{F}(U_G, U_D) = \frac{F(U_G) + F(U_D)}{2} - c\frac{U_D - U_G}{2},$$

avec

$$c = \sup_{U=U_G,U_D} (\sup_{j\in\{1,2\}} |\lambda_j(U)|),$$

où $\lambda_1(U)$ and $\lambda_2(U)$ sont les valeurs propres du système.

2.7.3 Observation pour HLL

On va maintenant construire le flux HLL, c'est une sophistication de nos observations précédentes.

On est sur la face $i + 1/2$, donc i est à gauche et $i + 1$ à droite. Soit U_G, la valeur à gauche qui est U_i^n, et U_D, la valeur à droite qui est U_{i+1}^n. Nous avons vu qu'une discontinuité entre les valeurs U_G et U_D pouvait se déplacer à la vitesse $\frac{F(U_G)-F(U_D)}{U_G-U_D}$, nous allons utiliser cette propriété pour construire un "solveur de Riemann". Nous sommes dans le cas où il peut y avoir deux vitesses (deux valeurs propres). En fait il peut y avoir propagation de ce choc, ou inversement de la détente associée (de subtiles conditions d'entropie apparaissent, nous n'en parlerons pas).

Si c_1 et c_2 sont positifs, tout va à droite et on choisit pour valeur du flux $F(U_G)$ (c'est le flux upwind simple), si c_1 et c_2 sont négatifs, tout va à gauche et on choisit pour valeur du flux $F(U_D)$ (l'information va dans l'autre sens). Si c_1 est négatif et c_2 positif, alors une partie va à droite et

l'autre à gauche, voir figure 8. Soit U^* la valeur entre les deux caractéristiques et $F(U^*)$ la valeur du flux par la propagation du "choc" on a de part et d'autre:

$$F(U_G) - F(U^*) = c_1(U_G - U^*) \text{ et } F(U^*) - F(U_D) = c_2(U^* - U_D)$$

par élimination de U^* l'état intermédiaire, on a $c_2F(U_G) - c_1F(U_D) - (c_2 - c_1)F(U^*) = c_1c_2(U_G - U_D)$ donc le flux

$$F(U^*) = \frac{c_2F(U_G) - c_1F(U_D)}{(c_2 - c_1)} + \frac{c_1c_2}{(c_2 - c_1)}(U_D - U_G)$$

Ce $F(U^*)$ est noté plus bas $\mathcal{F}(U_G, U_D)$, puisque l'on va ainsi approximer la valeur du flux sur la face séparant les deux états U_G et U_D.

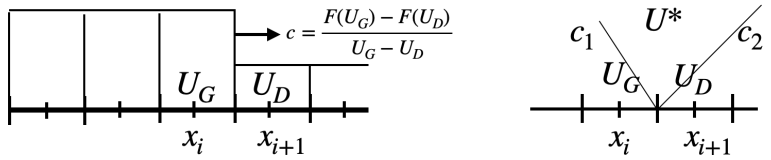


Figure 8: Gauche: une discontinuité se déplace. Droite: deux directions transportant de l'information partant de la face séparant U_G et U_D.

2.7.4 flux HLL

Le flux HLL (Harten, Lax & van Leer 1983) s'écrit

$$\mathcal{F}(U_G, U_D) = \begin{cases} F(U_G) & \text{if } 0 \le c_1 \\ \dfrac{c_2F(U_G) - c_1F(U_D)}{c_2 - c_1} + \dfrac{c_1c_2}{c_2 - c_1}(U_D - U_G) & \text{if } c_1 < 0 < c_2 \\ F(U_D) & \text{if } c_2 \le 0 \end{cases}, \tag{30}$$

avec

$$c_1 = \inf_{U=U_G,U_D} (\inf_{j\in\{1,2\}} \lambda_j(U)) \text{ and } c_2 = \sup_{U=U_G,U_D} (\sup_{j\in\{1,2\}} \lambda_j(U)),$$

où $\lambda_1(U)$ et $\lambda_2(U)$ sont les valeurs propres (*eigenvalues*) du système.

2.7.5 et les autres flux...

D'autres flux seront considérés tels que le flux HLLC, Cinétique, Beam-Warming, Lax-Wendroff, Lax Friedrichs, upwind, MUSCL, Kurganov, Godunov, MacCormack, Richtmeyer, S_α^β, Glimm, etc. etc

3 Convergence, CFL, Stabilité

3.1 CFL

La méthode doit être "stable", ce qui veut dire que les petites erreurs ne sont pas amplifiées avec les itérations temporelles successives. On a vu au détour de ce cours la condition CFL des noms Courant, Friedrichs, Lewy, (1928), [5]: on prend un pas de temps assez petit par rapport à la vitesse des ondes locales c_0 et au Δx.

$$CFL = \frac{c_0 \Delta t}{\Delta x}$$

3.2 Ordre du schéma

Les schémas que nous avons construits sont d'ordre un. C'est à dire que l'erreur est en Δx ou en Δt (au CFL d'ordre un près). Un schéma d'ordre n est tel que l'erreur est en $(\Delta x)^n$.

En pratique, on trace l'erreur à un temps donné t_1 pour un nombre de points donné, puis on multiplie par 2, 4, 8 ... le nombre de points (on résout le problème numériquement pour un nombre N de points donnés au temps t_1 cela donne $U_N(t_1)$). Si on a une solution analytique $U_e(t)$, on trace l'erreur $|U_e(t_1) - U_N(t_1)|$ entre la solution exacte et la solution approchée numériquement en fonction du nombre de points de discrétisation et on vérifie que $|U_e(t_1) - U_N(t_1)|$ décroit avec N. EN traçant en log, on essaye de trouver l'exposant de la décroissance. Ici, ce sera -1.

Si on n'a pas de solution analytique, alors on calcule $U_{Nmax}(t_1)$ pour un $Nmax$ très très grand et on admet que l'on est arrivé à convergence. On trace l'erreur $|U_{Nmax}(t_1) - U_N(t_1)|$ entre la solution à grand nombre de points et la solution approchée numériquement en fonction du nombre de points de discrétisation et on vérifie que $|U_{Nmax}(t_1) - U_N(t_1)|$ décroit avec N.

Il est grand temps de passer à la pratique....

3.3 Implémentation pratique en C

3.3.1 fichiers

Dans la suite on présente les éléments d'un programme simple en C, ce programme est sur la page de *Basilisk* `http://basilisk.fr/sandbox/M1EMN/Exemples/svdb.c`, mais c'est du C. Il est conseillé de regarder ce programme pour comprendre la suite. En effet la suite se focalise sur des extraits de ce programme pour l'expliquer. Ce programme est aussi en section § 3.6

Il sera ensuite modifié pour un fond variable.

Ce programme est aussi en `python` dans les deux fichiers suivants:
- `https://colab.research.google.com/drive/1960Q9Cgu9anAv6wfB9MblsBFo0BvWdrO`
- `http://basilisk.fr/sandbox/M1EMN/PYTHON/svdb.py`

Ce programme est aussi écrit en "langage" *Basilisk*
- `http://basilisk.fr/sandbox/M1EMN/Exemples/damb.c`

3.3.2 lecture du fichier

On ne fait pas tout d'abord une lecture ligne à ligne, mais on repère des blocs principaux. On repère donc dans le code la boucle en temps qui traduit l'évolution temporelle

```
t=0;
while(t<=tmax){    // boucle en temps
      t=t+dt;
     ...   }
```

On va utiliser des pointeurs h et u tels que leur produit soit le flux: Q[i]=h[i]*un[i]; ces pointeurs sont définis auparavant plus haut dans le code (c'est l'originalité par rapport à `python`) par

```
  double*x=NULL,*h=NULL,*u=NULL;
  double*un=NULL,*hn=NULL;
  nx=160;
  x= (double*)calloc(nx+2,sizeof(double));
  h= (double*)calloc(nx+2,sizeof(double));
  u= (double*)calloc(nx+2,sizeof(double));
  un=(double*)calloc(nx+2,sizeof(double));
  hn=(double*)calloc(nx+2,sizeof(double));
```

h et u est celui en cours, hn et un seront les nouveaux calculés à partir de h et u. on fera une boucle pour les points intérieurs,

```
   for(i=1;i<=nx;i++) {    ... }
```

dans cette boucle on calcule le nouveau h appelé hn et le nouveau flux q pour tout de suite exprimer la nouvelle vitesse un[i]=q/hn[i]; on remarque le test sur la positivité de la hauteur qui permet la transition sec mouillé. On voit aussi les pointeurs fp et fd des flux, cette boucle est le codage du

schéma explicite en voumes finis (17):

$$U_i^{n+1} = U_i^n - \Delta t \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} \text{ avec } U = (h, Q) \text{ et } F = (f_p, f_d)$$

ce qui se transcrit en:

```
for(i=1;i<=nx;i++)
  {hn[i]=h[i]- dt*(fp[i+1]-fp[i])/dx;    //cons. de la masse
   if(h[i]>0.){                           //cons. qunatite de mouv.
      q=h[i]*u[i]-dt*(fd[i+1]-fd[i])/dx;
      un[i]=q/hn[i];}
      else{
      un[i]=0.;}
   }
```

où fp[i] (premier) et fd[i] (deuxième) sont les deux composantes première et deuxième du flux $F_{i-1/2}$ et où fp[i+1] et fd[i+1] sont les deux composantes première et deuxième du flux $F_{i+1/2}$. La notation n'est pas fantastique car il n'y a pas de demi indices! fp[i+1] est donc sur la face entre $i+1$ et i.

On remarque que la transition sec/mouillé, est faite simplement: sur un sol sec, pas de vitesse.

On n'oublie pas de swapper h et u et hn et un à la fin, le nouveau devient l'ancien:

```
  for(i=0;i<=nx+1;i++)
    {   h[i]=hn[i];
      u[i]=un[i];
      Q[i]=hn[i]*un[i];
    }
```

3.4 Implémentation en C des flux

L'équation

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) \tag{31}$$

devient les deux lignes suivantes

```
   for(i=1;i<=nx;i++)
    {fp[i]=FR1(u[i-1],u[i],h[i-1],h[i]);
     fd[i]=FR2(u[i-1],u[i],h[i-1],h[i]);  }
```

le flux étant calculé par les fonctions,

```
double FR1(double ug,double ud,double hg,double hd)
double FR2(double ug,double ud,double hg,double hd)
```

la vitesse

$$c = \sup_{U=U_G,U_D} \left(\sup_{j\in\{1,2\}} |\lambda_j(U)| \right),$$

où $\lambda_1(U)$ and $\lambda_2(U)$ sont les valeurs propres du système. est calculée par

```
c=fmax(fabs(ug)+sqrt(hg),fabs(ud)+sqrt(hd));
```

• le flux de Rusanov

$$\mathcal{F}(U_G,U_D) = \frac{F(U_G)+F(U_D)}{2} - c\frac{U_D-U_G}{2},$$

fera apparaître pour h dans FR1

```
(hg*ug+hd*ud)*0.5-c*(hd-hg)*0.5;
```

et pour Q dans FR2

```
(ug*ug*hg + hg*hg/2. + ud*ud*hd + hd*hd/2.)*0.5
                                        - c*(hd*ud-hg*ug)*0.5;
```

• le flux HLL:

$$\mathcal{F}(U_G,U_D) = \begin{cases} F(U_G) & \text{if } 0 \le c_1 \\ \dfrac{c_2F(U_G)-c_1F(U_D)}{c_2-c_1} + \dfrac{c_1c_2}{c_2-c_1}(U_D-U_G) & \text{if } c_1 < 0 < c_2 \\ F(U_D) & \text{if } c_2 \le 0 \end{cases}, \tag{32}$$

a besoin des deux vitesses

```
c1=fmin(ug -sqrt(hg),ud-sqrt(hd));
c2=fmax(ug +sqrt(hg),ud+sqrt(hd));
```

puis, pour les deux flux notes f1 et f2:

```
if (c1>=0.){
f1=hg*ug;
f2=hg*ug*ug+hg*hg/2;
cfl=fabs(c2);}
else{
  if ((c1<0.)&&(0.<c2)) {
f1=(c2*hg*ug-c1*hd*ud)/(c2-c1)+c1*c2*(hd-hg)/(c2-c1);
f2=(c2*(hg*ug*ug+hg*hg/2)  -c1*(hd*ud*ud+hd*hd/2))/(c2-c1) +c1*c2*(hd*ud-hg*ug)/(c2-c1);
cfl=fmax(fabs(c1),fabs(c2));
  } else {
f1=hd*ud;
f2=hd*ud*ud+hd*hd/2.;
cfl=fabs(c1);} }
```

3.5 Exemples de runs

Une mise en oeuvre de ces solutions est le problème de rupture de barrage. On a vu dans [1] la solution exacte de la rupture de barrage, limitée par $u = 0$ à gauche et $h = 0$ à droite.
• pour $-\infty < x < -t\sqrt{gh_0}$ on a $h = h_0$, la hauteur d'eau ne varie pas. Le signal de la perturbation due au barrage détruit se propage vers l'amont à la vitesse c_0.
• pour $-t\sqrt{gh_0} < x < 2t\sqrt{gh_0}$ on a $h = \frac{x^2}{9gt^2} - \frac{4x}{9t}\sqrt{\frac{h_0}{g}} + \frac{4h_0}{9}$ et $u = (2/3)(x/t + \sqrt{gh_0})$. On est dans le régime d'onde simple présenté auparavant.
• pour $2t\sqrt{gh_0} < x < \infty$ on a $h = 0$, on a la deuxième limite correspondant à la hauteur nulle. Remarquons que lors de la rupture d'un barrage, le débit est constant en $x = 0$.

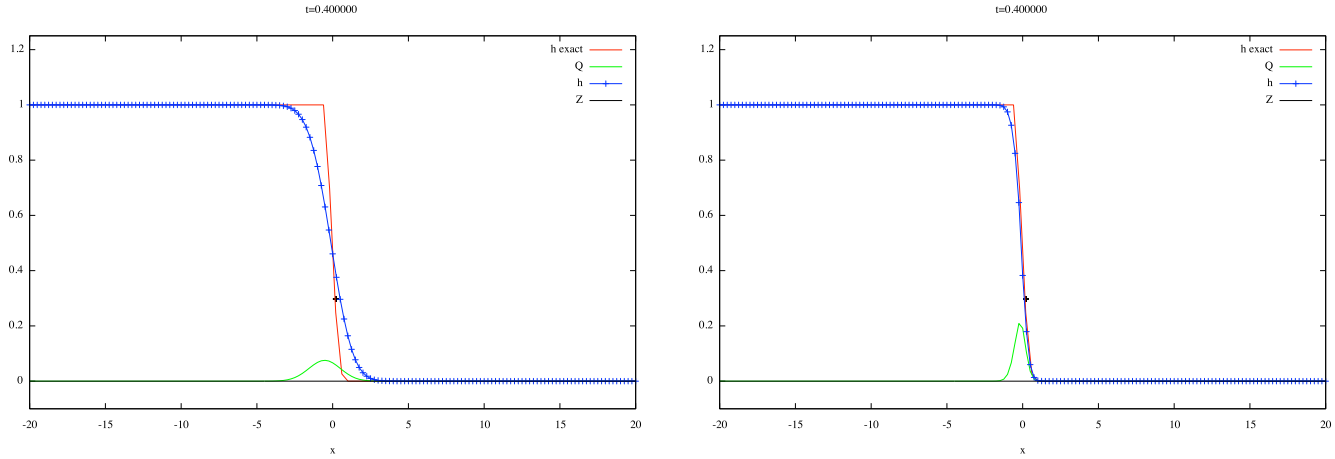


Figure 9: Départ de rupture de barrage. A gauche le schéma de Lax, à droite, au même instant le schéma de Rusanov est plus proche de la solution exacte en rouge de par la caractère trop diffusif de Lax.

Le programme est donné maintenant.

3.6 Programme

Le programme `http://basilisk.fr/sandbox/M1EMN/Exemples/svdb.c` dans le fichier `svdb.c` est retranscrit
Une version python est `http://basilisk.fr/sandbox/M1EMN/PYTHON/svdb.py`

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include <math.h>
#include <string.h>
// biblio : Delestre These,  PYL
     double*x=NULL,*h=NULL,*u=NULL,*Q=NULL;
     double t,dt,tmax,dx;
     int nx;
  //Rusanov
double FR1(double ug,double ud,double hg,double hd)
 { double c;
   c=fmax(fabs(ug)+sqrt(hg),fabs(ud)+sqrt(hd));
   return (hg*ug+hd*ud)*0.5-c*(hd-hg)*0.5;
 }
double FR2(double ug,double ud,double hg,double hd)
 { double c;
   c=fmax(fabs(ug)+sqrt(hg),fabs(ud)+sqrt(hd));
   return (ug*ug*hg + hg*hg/2. + ud*ud*hd + hd*hd/2.)*0.5 - c*(hd*ud-hg*ug)*0.5;
 }
/* ------------------------------     */
int main (int argc, const char *argv[]) {
     int  i,it=0;
     FILE *g;
     double*fp=NULL,*fd=NULL,*un=NULL,*hn=NULL;
     double  q,y1,y2;
// parametres
dt=0.01;
tmax=10;
dx=0.25;
nx=160;
t=0;
  fprintf(stderr," ----------------------                                \n");
  fprintf(stderr,"                        <--                            \n");
  fprintf(stderr,"                           |--->                       \n");
  fprintf(stderr," ------------------------------------------------------\n");
       x= (double*)calloc(nx+2,sizeof(double));
       h= (double*)calloc(nx+2,sizeof(double));
       Q= (double*)calloc(nx+2,sizeof(double));
       u= (double*)calloc(nx+2,sizeof(double));
       fp=(double*)calloc(nx+2,sizeof(double));
       fd=(double*)calloc(nx+2,sizeof(double));
       un=(double*)calloc(nx+2,sizeof(double));
       hn=(double*)calloc(nx+2,sizeof(double));
// initialisation cond init ----------------------------
  for(i=0;i<=nx+1;i++)
    {x[i]=(i-nx/2)*dx;
     h[i]=1*( x[i]<0);
     u[i]=0;
     Q[i]=u[i]*h[i];}
// initialisation du fichier de sortie ----------------------
       g = fopen("solxhQt.OUT", "w");
       fclose(g);

    while(t<tmax){    // boucle en temps
      t=t+dt;
      it=it+1;

   for(i=1;i<=nx+1;i++)
    {   fp[i]=FR1(u[i-1],u[i],h[i-1],h[i]);
        fd[i]=FR2(u[i-1],u[i],h[i-1],h[i]); }

   for(i=1;i<nx+1;i++)
    {hn[i]=h[i]- dt*(fp[i+1]-fp[i])/dx;    //conservation de la masse
  if(h[i]>0.){                              //conservation qunatite de mouvement
      q=h[i]*u[i]-dt*(fd[i+1]-fd[i])/dx ;
      un[i]=q/hn[i];}
      else{
      un[i]=0.;}
    }
 // flux nul en entree sortie
        hn[0]=hn[1];
        un[0]=un[1];
        hn[nx+1]=hn[nx];
```

```
          un[nx+1]=un[nx];
    //swap
   for(i=0;i<=nx;i++)
     { h[i]=hn[i];
       u[i]=un[i];
       Q[i]=hn[i]*un[i];}

   if(it%100==0){
     /* Saving the fields */
     g = fopen("solxhQt.OUT", "a");
     for (i=0; i<=nx;i++) { fprintf(g,"%lf %lf %lf %lf \n",x[i],h[i],Q[i],t);}
      fprintf(g,"\n");
      fclose(g);}
 }
      free(x);
      free(fp);
      free(fd);
      free(un);
      free(hn);
     return 0;
}
```

Bien entendu ceci est sauvé dans un fichier texte ASCII appelé svdb.c et sauvé dans un répertoire clairement indentifié. On revient dans terminal, on compile et on crée l'exécutable db

```
cc  -O3  -ffast-math -std=C99 -lm  svdb.c -o db
```

pour lancer le programme: `./db`
un fichier appelé solxhQt.OUT avec x h Q et t est créé. Pour le tracer, on lance gnuplot dans un AUTRE terminal pour pouvoir jongler avec un terminal de compilation/ exécution et un terminal de tracé. On garde au chaud une fenêtre de l'éditeur de texte pour faire des modifications au fur et à mesure...

```
set ylabel "t"
set xlabel "x"
set hidden3d
sp'solxhQt.OUT' u 1:4:2 not w l linec 3
```

On obtient la courbe suivante hauteur d'eau en fonction de x et de t.

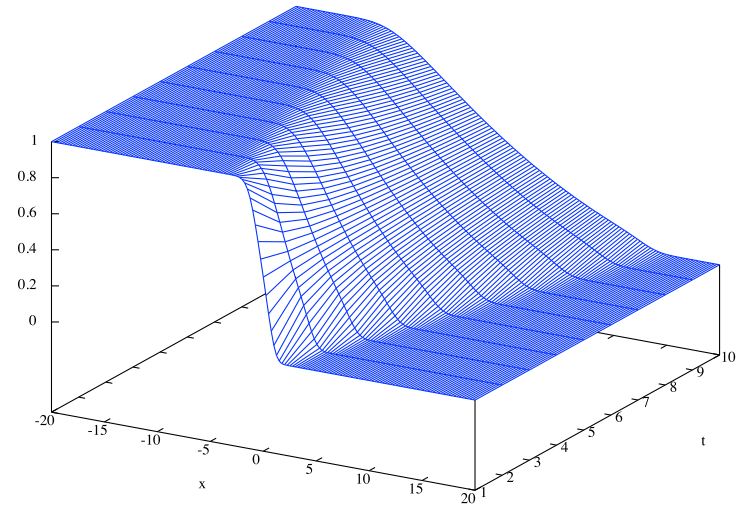


Figure 10: Rupture de barrage. Schéma de Rusanov, la hauteur d'eau en fonction de x et de t.

• jouons avec gnuplot et le terminal pour faire un film en direct
On peut reprendre le fichier, juste après la sauvegarde, on insère une sortie pour gnuplot et on garde la fin:

```
  if(it%100==0){
   /* Saving the fields */
g = fopen("solxhQt.OUT", "a");
for (i=0; i<=nx;i++)
{      fprintf(g,"%lf %lf %lf %lf \n",x[i],h[i],Q[i],t);}
 fprintf(g,"\n");
 fclose(g);
  }

    printf("t=%lf\n",t);
printf("set xlabel \"x\" ; set title \"t=%lf \"\n",t);
     y1=-.1,y2=1.25;
printf("set label \"+\" at 0,.296296296296296 \n");
printf("h(x)=(((x-0)<-t)+((x-0)>-t)*(2./3*(1-(x-0)/(2*t)))**2)*(((x-0)<2*t)) \n");


printf("p[%lf:%lf][%lf:%lf] h(x) t'h exact','-' u 1:2 t'Q'  w l,'-'  t'h' w lp,'-'  t'Z' w l linec -1\n ",
   -nx*dx/2,nx*dx/2,y1,y2);
for (i=0; i<=nx;i++)
{
printf("%lf %lf \n",x[i],Q[i]);}
printf("e \n");
for (i=0; i<=nx;i++)
{
printf("%lf %lf \n",x[i],h[i]+Z);}
printf("e \n");
for (i=0; i<=nx;i++)
{
printf("%lf %lf \n",x[i],Z);}
printf("e \n");
      }

     free(x);
     free(fp);
     free(fd);
     free(un);
     free(hn);
    return 0;
}
```

on le compile et on crée l'exécutable db

```
cc  -O3  -ffast-math -std=C99 -lm  svdb.c -o db
```

pour lancer le programme `./db | gnuplot` une fenêtre graphique s'ouvre et trace x h Q en direct.

Pour plus d'exemple de `gnuplot` voir `http://basilisk.fr/sandbox/M1EMN/BASIC/gnuplot_examples.c` et bien sûr `http://www.gnuplotting.org`

Dans la suite on regarde l'influence de la discrétisation du fond et on met le frottement.

4 Résolution numérique, influence du fond

Cette section sur les schémas bien balancés peut être omise en première lecture
Maintenant que nous avons vu ce qui se passe pour un fond plat, nous allons continuer et étoffer en considérant ce qui se passe pour un fond bombé. Le système de Saint-Venant sans friction admet des solutions stationnaires

$$\partial_t h = \partial_t u = \partial_t Q = 0. \tag{33}$$

Le flux est donc constant, $\partial_x Q(x,t) = 0$, et la quantité de mouvement s'écrit avec Bernoulli

$$\left\{ \begin{array}{l} Q = Q_0 \\ \dfrac{{Q_0}^2}{2gh^2} + Z + h = Cst \end{array} \right. , \tag{34}$$

des schémas numériques existent qui préservent cet équilibre, mais ils sont difficiles à mettre en oeuvre (voir [8], [9], [10] et [11]). Pour simplifier, on cherche à préserver uniquement les états d'équilibres (démarche de Audusse et al., voir aussi Delestre 2010)

$$\left\{ \begin{array}{l} q = u = 0 \\ \partial_x \left(g\dfrac{h^2}{2} \right) + gh\partial_x Z = 0 \end{array} \right. , \tag{35}$$

qui physiquement correspondent à l'état de "lac au repos" (*"lake at rest"*, "équilibre de l'homme mort" pour le cas artériel" Delestre & Lagrée). On a un équilibre dit "hydrostatique" entre la pression hydrostatique et l'accélération due au fond en pente $\partial_x Z$. Le second terme de (35) se réduit à

$$H = h + Z = Cst, \tag{36}$$

où H est le niveau d'eau *the water level*. Depuis Greenberg et al. [7], les schémas qui préservent (35) sont appelés "schémas bien balancés" *well-balanced schemes*.

4.1 traitement naïf, nécessité de "reconstruire"

Dans ce paragraphe nous montrons que le schéma numérique le plus simple ne préserve pas l'équilibre du lac. Un traitement naif du terme source consiste à écrire simplement les dérivées

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = -gh_i \frac{Z_{i+1/2} - Z_{i+1/2}}{\Delta x} \tag{37}$$

avec par exemple la moyenne

$$Z_{i+1/2} = \frac{Z_{i+1} + Z_i}{2}.$$

Nous calculons maintenant le terme de flux et nous le mettons avec le terme de source. Comme le terme de flux est

$$\frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = g\frac{[(h_{i+1}^2/2 + h_i^2/2) - (h_i^2/2 + h_{i-1}^2/2)}{2\Delta x} = g\frac{[(h_{i+1}^2/2 - h_{i-1}^2/2)}{2\Delta x}$$

et que le terme de source est:

$$-gh_i\frac{Z_{i+1/2}-Z_{i+1/2}}{\Delta x}=-gh_i\frac{Z_{i+1}-Z_{i-1}}{2\Delta x}.$$

le schéma à vitesse nulle donne en simplifiant par $g/(2\Delta x)$:

$$[(h_{i+1}+h_{i-1})/(2h_i)](h_{i+1}-h_{i-1})=-(Z_{i+1}-Z_{i-1})$$

on aurait aimé trouver

$$h_{i+1}-h_{i-1}=-(Z_{i+1}-Z_{i-1})$$

pour que l'équilibre soit préservé $h_{i+1}+Z_{i+1}=h_{i-1}+Z_{i-1}$. on constate donc qu'un traitement trop simple des termes sources produit des courants non physiques puisque l'équilibre n'est pas préservé....

Bien sûr, on peut dire que le choix de moyenne n'est pas le bon, que c'était évident (ah?), si on avait écrit le terme source sous la forme, non pas

$$-gh_i\frac{Z_{i+1/2}-Z_{i+1/2}}{\Delta x}$$

mais

$$-g(1/2)(h_{i+1}+h_{i-1})\frac{Z_{i+1/2}-Z_{i+1/2}}{\Delta x}$$

on aurait conservé l'équilibre.

Mais cette forme n'est pas "conservative" dans son expression. Depuis les premiers travaux espagnols de 1994, le but du jeu est donc de trouver une forme conservative pour des variables reconstruites

$$F(U_i^*,U_{i+1}^*)-F(U_{i-1}^*,U_i^*)=0$$

cf Botta 2004. On veut trouver une forme conservative pour le terme de pente qui est $-\int_{x_{i-1/2}}^{x_{i+1/2}}g\partial_x Z dx$. Si on écrit ce terme de pente

$$-\int_{x_{i-1/2}}^{x_{i+1/2}}g\partial_x Z dx=\mathcal{P}(h_{i+1/2}^*)-\mathcal{P}(h_{i-1/2}^*)$$

avec h^* hauteur d'eau reconstruite, on a une forme "conservative" qui sera adaptée aux volumes finis. Cette fonction est $\mathcal{P}(h^*)=\frac{1}{2}gh^{*2}$, mais h^* est la hauteur "reconstruite" à déterminer. Pour ce faire, on part de l'équilibre

$$h_i+Z_i=h_{i+1}+Z_{i+1}$$

en définissant

$$h_{i+1/2G}^*=h_i+Z_i-Z_{i+1/2}^*,\quad h_{i+1/2D}^*=h_{i+1}+Z_{i+1}-Z_{i+1/2}^*$$

l'équilibre est bien

$$h_i+Z_i=h_{i+1}+Z_{i+1}\ \text{ donne } h_{i+1/2G}^*=h_{i+1/2D}^*$$

il nous faut un choix judicieux de $Z_{i+1/2}^*$, pour inégalité entropie, et positivité , Audusse ([2],[3]) propose

$$Z_{i+1/2}^*=max(Z_i,Z_{i+1})$$

ainsi que

$$h_{i+1/2GD}^*=max(h_{i+1/2GD},0)$$

4.2 Bien balancée

La méthode *well-balanced method* inspirée de la reconstruction hydrostatique ([3] et [4]), le terme de friction est toujours absent, et on construit un schéma qui préserve les états stationnaires (dont celui du lac au repos (36)), comme suggéré par Audusse et al. [3], on définit:

$$\left\{\begin{array}{l} h_{i+1/2G} = max(h_i + Z_i - max(Z_i, Z_{i+1})), \\ U_{i+1/2G} = (h_{i+1/2G}, h_{i+1/2G}u_i) \\ h_{i+1/2D} = max(h_{i+1} + Z_{i+1} - max(Z_i, Z_{i+1})), \\ U_{i+1/2D} = (h_{i+1/2D}, h_{i+1/2D}u_i) \end{array}\right. , \tag{38}$$

le max permet d'assurer que les hauteurs d'eau obtenues sont positives. Nous pouvons remarquer qu'avec la reconstruction hydrostatique, les variables vérifient l'équation d'équilibre du lac au repos ainsi que des équilibres plus généraux de la forme (dans la limite $u \ll \sqrt{gh}$):

$$h + z = Cte, \;\; u = Cte.$$

Le schéma s'écrit alors:

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(F^n_{i+1/2G} - F^n_{i-1/2D}), \tag{39}$$

avec

$$\begin{array}{l} F^n_{i+1/2G} = F^n_{i+1/2} + S_{i+1/2G} \\ F^n_{i-1/2D} = F^n_{i-1/2} + S_{i-1/2D} \end{array} , \tag{40}$$

avec pour les S l'expression "conservative" avec $\mathcal{P}(h) = g\dfrac{h^2}{2}$.

$$\begin{array}{l} S_{i+1/2G} = \begin{pmatrix} 0 \\ \mathcal{P}(h_i^n) - \mathcal{P}(h^n_{i+1/2G}) \end{pmatrix} \\ S_{i-1/2D} = \begin{pmatrix} 0 \\ \mathcal{P}(h_i^n) - \mathcal{P}(h^n_{i-1/2D}) \end{pmatrix} \end{array} \tag{41}$$

Le flux numérique $F^n_{i+1/2}$ est reconstruit (38):

$$F^n_{i+1/2} = \mathcal{F}(U^n_{i+1/2G}, U^n_{i+1/2D}).$$

4.3 C bien balancée

On va donc coder le nouveau flux en ajoutant les variations liées à $S_{G,D}$

```
for(i=1;i<nx;i++)
 {
   hn[i]=h[i]- dt*(fp[i+1]-fp[i])/dx;    //cons. de la masse
  if(h[i]>0.){                           //cons. qunatit\'e de mouvement
   q=h[i]*u[i]-dt*(fd[i+1]-fd[i])/dx
    -dt*( hig[i]*hig[i]/2 - hg[i]*hg[i]/2
          + hd[i]*hd[i]/2  - hid[i]*hid[i]/2)/dx;
```

```
        un[i]=q/hn[i];}
        else{
        un[i]=0.;}
    }
```

mais il faut calculer les hauteurs reconstruites, d'où la boucle écrite avant la précédente

```
 for(i=1;i<=nx;i++)
    {
       reconsetat(hd[i-1],hg[i],dZ[i-1],&f1,&f2);
       hid[i-1]=f1;
       hig[i]=f2;
      }
```

avec la fonction qui calcule les "max", ((38))

```
 void   reconsetat(double hl,double hr,double dz,double *hil,double *hir)
{      *hil=fmax(0.,hl-fmax(0.0,dz));
       *hir=fmax(0.,hr-fmax(0.0,-dz));
}
```

un exemple de mise en oeuvre en C est dans `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/svdbWB.c` rupture de barrage avec frottement laminaire, mur à gauche et plage à droite.

5 Résolution numérique, Frottement

5.1 Terme Source de frottement

Jusqu'à présent, on a complètement ignoré le terme de frottement car on a dit que l'on avait "splitté" le problème. Cela veut dire que

$$\partial_t U + \partial_x F(U) = S_Z(U) + S_f(U),$$

avec pour $S(U) = S_Z(U) + S_f(U)$, la partie de pente $S_Z(U)$ réécrite sous forme conservative avec $\mathcal{P}$ mais maintenant il y a en plus le frottement $S_f(U)$. Décomposons en une étape U^* avec reconstruction hydrostatique et sans frottement:

$$U_i^* = U_i^n - \frac{\Delta t}{\Delta x}(F^n_{i+1/2L} - F^n_{i-1/2R})$$

puis une étape de frottement pur

$$\left(\frac{U_i^{n+1} - U_i^*}{\Delta t}\right) = S_f(U^{n+1}), \tag{42}$$

la somme des deux redonne le problème total par disparition du champ intermédiaire U_i^*. En pratique on calcule $(-\tau)$ comme on peut (en explicite par exemple)

$$Q^{n+1} - Q^* = (\Delta t)(-\tau/\rho)$$

et voilà.

5.2 Cas Poiseuille

Dans le cas Poiseuille, $\tau/\rho = 3\nu Q/h^2$, on mettra Q en implicite et h en explicite

$$Q^{n+1} - Q^* = -(\Delta t)3\nu Q^{n+1}/h^{*2}$$

donc on trouve

$$Q^{n+1} = Q^*/(1 + (\Delta t)3\nu/h^{*2})$$

tous les effets sont maintenant réunis. Dans ce cas on écrira (avec $C_f = 3\nu$ coefficient associé au frottement laminaire, attention ne pas oublier de le déclarer!):

```
if((q!=0)&&(Cf>0)) q = q/(1+dt*Cf/h[i]/h[i]);
```

un exemple de mise en oeuvre en C est dans `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/svdbWB.c` rupture de barrage avec frottement laminaire, mur à gauche et plage à droite.

5.3 Cas turbulent

Dans le cas turbulent (on peut mettre le 1/2, mais le ρ disparaît), $\tau/\rho = C_f u^2$, on va donc plutôt corriger la vitesse

$$u^{n+1} - u^* = -C_f u^{n+1} u^*/h^*$$

donc

$$u^{n+1} = u^*/(1 + (\Delta t) C_f/h^*)$$

tous les effets sont maintenant réunis. Dans ce cas on écrira (avec C_f coefficient associé au frottement turbulent, attention ne pas oublier de le déclarer!):

```
if((q!=0)&&(Cf>0)) q = q/(1+dt*Cf/h[i]);

 un[i]=q/hn[i];
```

5.4 Terme Source de frottement en C cas granulaire

Dans le cas des écoulements granulaires $\tau = \rho\mu g h|Q|/Q$, le terme $|Q|/Q$ permet d'orienter le frottement pour freiner.

$$Q^{n+1} - Q^* = -(\Delta t)(g\mu h Q^{n+1}/|Q^*|)$$

donc

$$Q^{n+1} = Q^*/(1 + (\Delta t)\mu g h/|Q^*|)$$

Tous les effets sont maintenant réunis, la boucle s'écrite avec la reconstruction de $-gh\partial_x Z$ et le frottement τ

```
for(i=1;i<nx;i++)
 {
   hn[i]=h[i]- dt*(fp[i+1]-fp[i])/dx;    //conservation de la masse
  if(h[i]>0.){
   q=h[i]*u[i]-dt*(fd[i+1]-fd[i])/dx
    -dt*( hig[i]*hig[i]/2 - hg[i]*hg[i]/2
          + hd[i]*hd[i]/2- hid[i]*hid[i]/2)/dx;
   if(q>0)   sign= 1;
   if(q<0)   sign=-1;
   if(q!=0) q = q/(1+dt*sign*mu*h[i]/q);
   un[i]=q/hn[i];}
   else{
   un[i]=0.;}
 }
```

mise à jour: en fait on peut intégrer exactement puisque la force reste constante opposée à la vitesse

$$u^* = u^n - \Delta t g\mu$$

comme la vitesse décroit et ne peut être inférieure à 0, donc $u^{n+1} = max(u^n - \Delta t g\mu, 0)\frac{u^n}{|u|^n}$.

6 Les conditions aux limites

6.1 Les volumes fantômes

Prenons la convention suivante: le domaine de longueur L entre $x = 0$ et $x = L$ (les deux faces extèmes) est découpé en N volumes de longueur $\Delta x = L/N$. Les cellules sont de longueur Δx, elles sont centrées en x_i, les faces sont en $x_{i-1/2}$ et $x_{i+1/2}$. Le centre est en $x_i = -\frac{\Delta x}{2} + i\Delta x$.

La première cellule est en $i = 1$; sa face de gauche est en $x = 0$ (cette face gauche $x = 0$ est donc $x_{-1/2}$), la première cellule est centrée en $x_1 = \Delta x/2$. Sa face de droite est en $x_{1/2} = \Delta x$.

La dernière en $i = N$. Elle est centrée en $x_N = L - \Delta x/2$. sa face de droite est en $x_{N+1/2} = L$.

Nous allons rajouter des "cellules fantômes", ou "cellules fictives" pour écrire les conditions aux limites. Nous avons donc besoin de ranger $N + 2$ éléments. Nous en rajoutons une de part et d'autre du domaine, à gauche en $i = 0$, avant la cellule 1 et centrée en $x_0 = 0 - \frac{\Delta x}{2}$, et à droite en $i = N + 1$ après la cellule N et centrée en $x_{N+1} = L + \frac{\Delta x}{2}$.

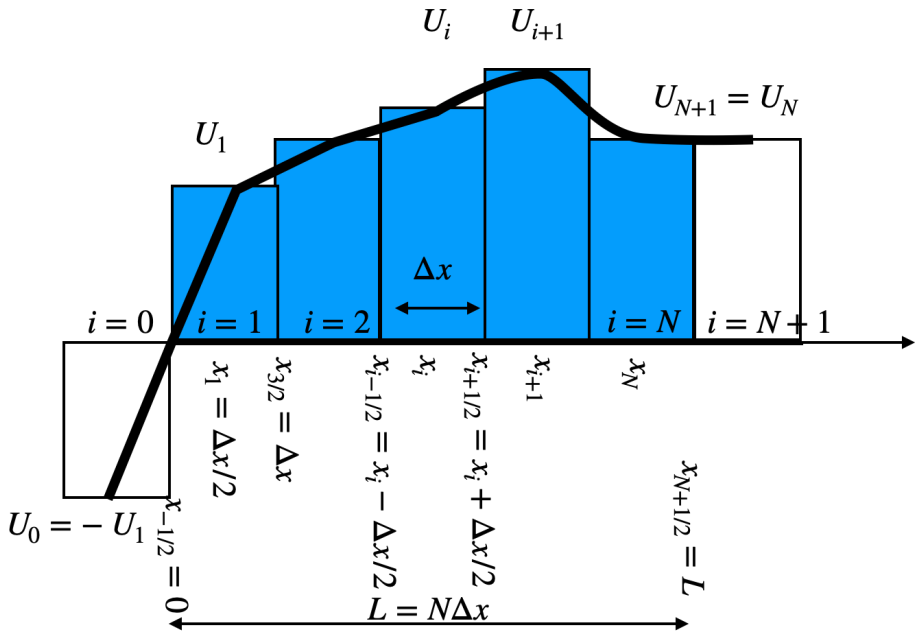


Figure 11: Condition de dirichlet nulle à gauche, en $x = 0$ et de Neumann nulle à droite en $x = L$, ajout de deux cellules supplémentaires, les cellules "fictives" ou "fantôme". A gauche on impose une valeur nulle en $x = 0$, donc la *ghost cell* a pour valeur $U_0 = -U_1$. A droite on impose un flux nul $\partial_x U = 0$ donc la *ghost cell* a pour valeur $U_{N+1} = U_N$.

6.2 Dirichlet/ Neumann

Supposons que l'on veuille imposer en $x = 0$ la valeur U_D (condition de Dirichlet) et en $x = L$ une dérivée f_N (condition de Neumann).

Dirichlet
La position $x = 0$ est une face de gauche du premier volume, où la valeur est U_1. On rajoute un volume fantôme (*ghost*) à gauche. Ce volume est centré en $x_0 = x_1 - \Delta x = -\frac{\Delta x}{2}$. La valeur y est U_0 que nous cherchons à déterminer. On suppose une extrapolation linéaire entre U_1 et U_0 de manière à ce que la face en $x = 0$ ait la valeur U_D:

$$U_D = \frac{U_0 + U_1}{2} \text{ ou } U_0 = 2U_D - U_1.$$

Par exemple si $U_D = 0$, alors il faut donner $U_0 = -U_1$ comme valeur à la cellule fantôme.

Neumann
Si on veut donner une dérivée, par exemple en $x = L$, la dérivée est $\partial_x U = f_N$, alors

$$f_N = \frac{U_{N+1} - U_N}{\Delta x} \text{ ou } U_{N+1} = U_N + f_N \Delta x.$$

Par exemple si $f_N = 0$ (cas d'une sortie libre), alors il faut donner $U_{N+1} = U_N$ comme valeur à la cellule fantôme.
...
Pour plus de détails voir *Boundary Conditions and Ghost Cells* chapitre 7 LeVeque 2002 *fictitious cells* p 224 Toro
...

6.3 Flux subcritique

Dans le cas de la donnée d'une hauteur ou d'un flux en entrée dans le cas subcritique, il faut revenir aux invariants de Riemann: $u - 2\sqrt{gh}$ est constant.
• si on veut imposer une hauteur d'eau à gauche h_D, la cellule fictive sera telle que

$$\left\{ \begin{array}{l} h_0 = h_D \\ u_0 = u_1 - 2(\sqrt{gh_1} - \sqrt{gh_D}) \end{array} \right. ,$$

• si on veut imposer un débit d'eau à gauche Q_D, la cellule fictive sera telle que $u_0 - 2\sqrt{gh_0} = u_1 - 2\sqrt{gh_1}$, il faut résoudre en h_0

$$0 = -Q_D + (u_1 - 2\sqrt{gh_1})h_0 + \sqrt{gh_0^3}. \tag{43}$$

par un Newton, on itère jusqu'à convergence

$$h_0^{k+1} = h_0^k - \frac{2\sqrt{(}gh_1)h_0^k - 2\sqrt{g(h_0^k)^3} + Q - h_0^k u_1}{2\sqrt{(}gh_1) - 3\sqrt{gh_0^k} - u_1}$$

pour obtenir avec assez de précision la valeur.
un exemple est dans `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/bosse_C/svbosse.c`

6.4 Cas supercritique

Soit h_0 et u_0 sont donnés en entrée.

Soit on impose un flux, alors on impose $\mathcal{F}_I = Q_D$

7 Exemples de résolution

7.1 Exemple de rupture de barrage

En C: `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/cours_MFEA_C_OK/index.html`
`http://basilisk.fr/sandbox/M1EMN/Exemples/svdb.c`

En python: `https://colab.research.google.com/drive/1960Q9Cgu9anAv6wfB9MblsBFo0BvWdrO`
`http://basilisk.fr/sandbox/M1EMN/PYTHON/svdb.py`

Cas d'écoulement de fluide parfait (pas de frottement) sur un fond plat, au temps $t = 0$, $h = 1$ pour $x > 0$ et $h = 0$ pour $x > 0$.

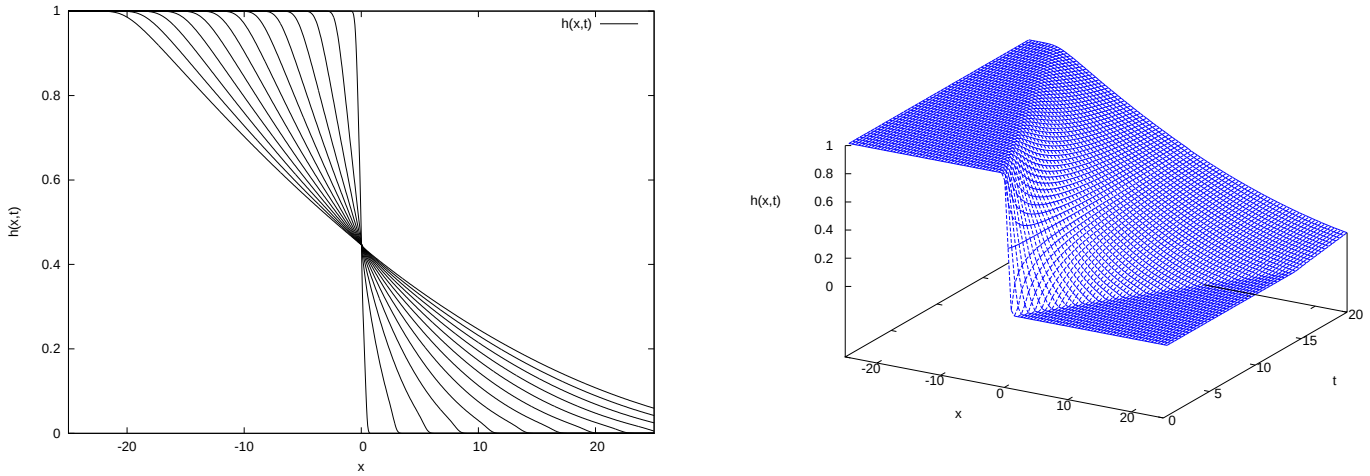
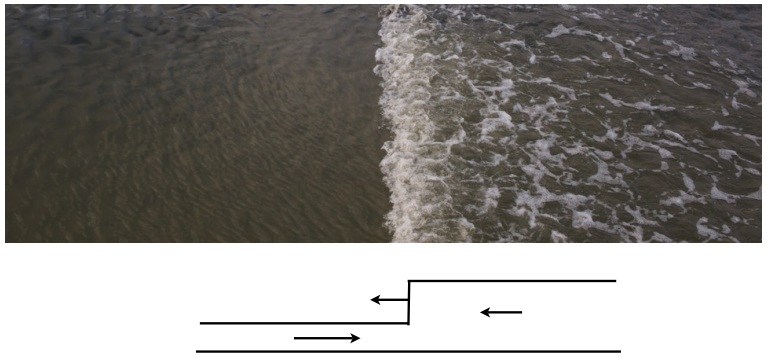


Figure 12: Gauche Rupture de barrage, la surface libre à différents temps, droite Rupture de barrage, vision 3D avec le temps t dans la troisième dimension.

7.2 Exemples de Ressaut Fixe

C'est un écoulement de fluide parfait (pas de frottement) sur un fond plat. Le programme calcule des ressauts (figure 13) si la hauteur en $x < 0$ est 1, la hauteur après le ressaut est on a vu: $(-1 + \sqrt{(}1 + 8Fr^2))/2 - 1$, on le vérifie pour $Fr = 1.2, 1.5$ et 2.



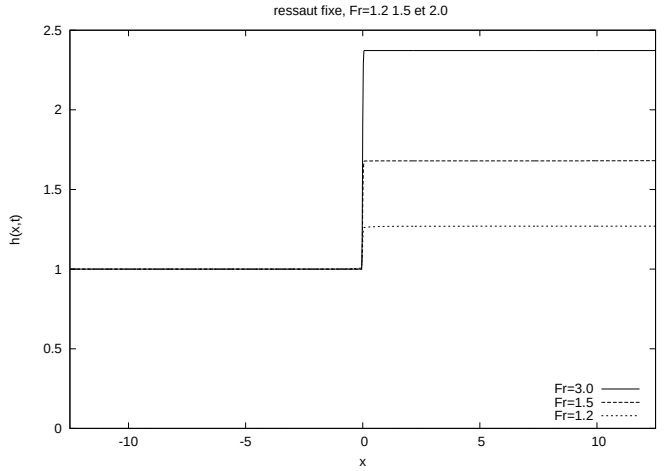


Figure 13: Gauche Un ressaut créé par la montée de la marée à Port à La Duc baie de la Fresnaye, 22, Photo PYL 22/08/09, en dessous sa simplification en deux niveaux. A droite : Exemples de ressauts fixes, la solution numérique est confondue avec la solution anlaytique, $h_1 = 1$ $h_2 = (-1 + \sqrt{1 + 8Fr^2})/2 - 1$

En pratique, on se donne une valeur de Froude, et une perturbation initiale passant de h_1 à h_2 avec une tangente hyperbolique

```
for(i=0;i<=nx;i++)
   {  x[i]=(i-nx/2)*dx;
      // ressaut
         Z[i]=0;
         h[i]=1+(((-1 + sqrt(1+8*Fr*Fr))/2)-1)*(1+tanh(x[i]))/2;
         Q[i]=Fr;
         u[i]=Q[i]/h[i
    }
```

7.3 Exemple de propagation d'une vague

Calcul de départ d'une vague 14, écoulement de fluide parfait (pas de frottement) sur un fond plat.

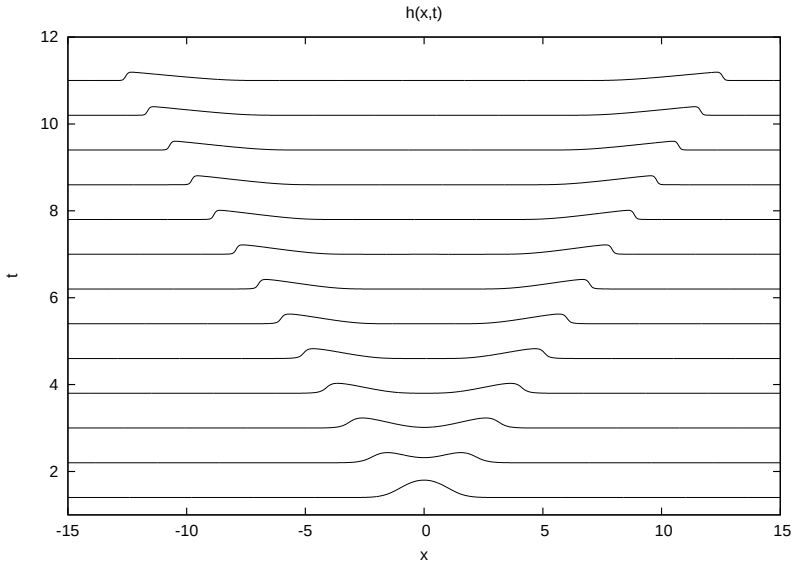


Figure 14: Exemple d'un signal initial: le temps est de bas en haut, une vague qui se brise en deux, une qui remonte, l'autre qui va à droite. Les non linéarités provoquent un raidissement des vagues.

7.4 Cas frottant: tas de Huppert (premier problème de Huppert)

7.4.1 Onde diffusive

Il s'agit de l'effondrement lent d'un tas visqueux sur un sol plat horizontal. On traite ici le problème avec la vision diffusive (6) qui s'écrit ici

$$\partial_t h + \partial_x Q(h, \partial_x h) = 0 \text{ avec le flux } Q = -\frac{1}{3} h^3 \partial_x h \tag{44}$$

On utilise la méthode en 1D avec une équation, ceci est détaillé dans le film suivant :
`https://dropsu.sorbonne-universite.fr/s/6nXz9AnxyqR86E8`
Où on vérifie la solution autosemblable.

Code:
- `http://basilisk.fr/sandbox/M1EMN/Exemples/viscous_collapse_noSV.c`

7.4.2 Saint-Venant

Soit on la considère comme solution de Saint-Venant à deux équations, et on retrouve numériquement aux temps longs la solution de lubrification de Huppert avec les équations de Saint-Venant.

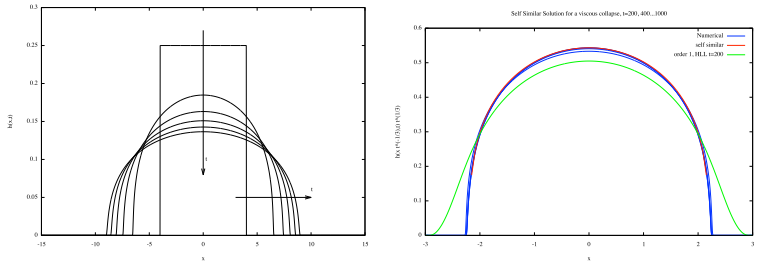


Figure 15: Gauche Plot at t=100,200,300...1000 of $h(x,t)$ with *Basilisk*. Droite Plot at t=100,200,300...1000 of $t^{1/5}h(x/t^{1/5},t)$ with *Basilisk*, with a FV code order 1 HLL and the analytical solution $(3b^2/(10k)(1-(\eta/b)^2))^{1/3}$.

Codes:
- `http://basilisk.fr/sandbox/M1EMN/PYTHON/svvisc.py`
- `http://basilisk.fr/sandbox/M1EMN/Exemples/viscous_collapse.c`

7.5 Cas granulaire

Effondrement d'un tas de longueur 2 et de hauteur 1, sur un fond plat, avec un frottement granulaire constant $\mu = 0.45$ comparaison du calcul *Gerris* avec `GfsRiver` et ce code

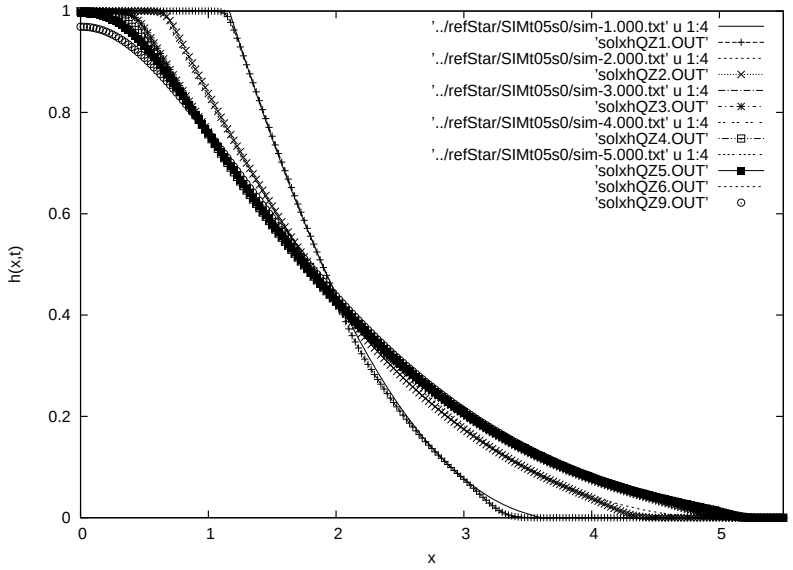


Figure 16: Ecoulement de granulaire, effondrement d'un tas de longueur 2 et de hauteur 1, comparaison du calcul *Gerris* `GfsRiver` et ce code, les résultats sont quasi superposés (la différence provient de la différence de traitement de flux)

```
#####
# Saint Venant-Savage Hutter
Define L0 5
Define eps 0.0000000000001
Define Nrfs 10
Define Htas 1

1 0 GfsRiver GfsBox GfsGEdge { x = 0.5} {
    PhysicalParams { L = L0 }
    RefineSolid Nrfs

# Set a solid boundary close to the top boundary to limit the
# domain width to one cell (i.e. a 1D domain)
```

```
    Solid (y/L0. + 1./pow(2,Nrfs) - 1e-5 - 0.5)

# Set the topography Zb and the initial water surface elevation P
    Init {} {
    Zb = 0
    Zpb = dx("Zb")
    P = {return (x<Ltas)? 0.1*Htas*source + (1.-source)*Htas : 0.0 ;}
    U = 0 }

    Init { istep = 1 } {
        mu = 0.45
        U = (U )/(1. +  dt *mu*P/(fabs(U)+eps))
        dm = source*t*dt*(x<Ltas)*(t<sqrt(2*(Htas-0.1*Htas)))
        P = P + dm
  }
    PhysicalParams { g = 1 }

# Use a first-order scheme rather than the default second-order
# minmod limiter. This is just to add some numerical damping.
    AdvectionParams {
#     gradient = gfs_center_minmod_gradient
    }

    OutputSimulation { istep = 25 } stdout
    OutputTime { step = 0.25 } stderr
# 1:x 2:y 3:z 4:P 5:U 6:V 7:Zb 8:H 9:Px 10:Py 11:Ux 12:Uy 13:Vx 14:Vy 15:Zbx 16:Zby 17:In 18:mu 19:DU
# p'sim.data'  u 1:($7+$4) t'Zb+h',''u 1:8t'eta'

    OutputSimulation { step = 0.25 } SIM/sim-%05.3f.txt { format = text }
    OutputSimulation { step = 0.25 } SIM/ksim-%g.txt { format = text }
    EventStop { istart = 100 istep = 100} U 1e-4 DU
    EventScript { step = 0.25 } { mv SIM/ksim-$GfsTime.txt sim.data}
}
GfsBox {
          left = Boundary { BcDirichlet U 0 }
      right = Boundary { BcNeumann U 0 }
}
# 1 2 right
##########
```

Implémentations de la rhéologie granulaire

- http://basilisk.fr/sandbox/M1EMN/PYTHON/shdb.py
- http://basilisk.fr/sandbox/M1EMN/Exemples/savagestaron.c

7.6 Exemple de propagation d'une vague qui arrive sur un fond

Calcul de départ d'une vague 18, écoulement de fluide parfait (pas de frottement) sur un fond plat puis remontée sur une plage. Le fond est en 0,

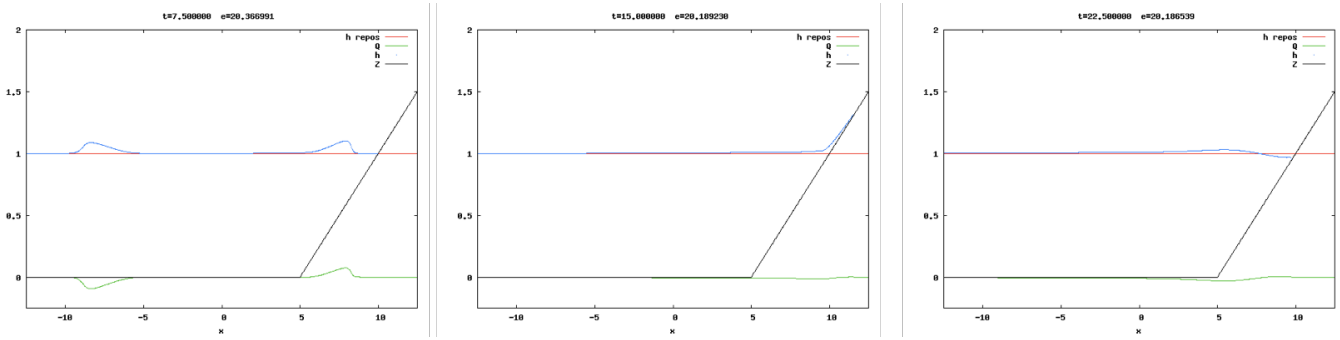


Figure 17: Fond, surface libre et surface de repos. Exemple d'un signal initial: une vague qui se brise en deux, une qui va à gauche, l'autre qui va à droite. Les non linéarités provoquent un raidissement des vagues d'autant plus près de la côte à droite où la vague va mourir.

puis remonte pour $x > x_{bosse}$: on a $Z = \alpha(x - x_{bosse})$, on se donne un niveau initial $1 + a_{bosse}e^{-x^2}$

```
for(i=0;i<=nx;i++)
    {   x[i]=(i-nx/2)*dx;
        Z[i]=alpha*(x[i]>xbosse)*(x[i]-xbosse);
    //   if(x[i]<=xbosse)Z[i]=0;
        h[i]=fmax(1+abosse*exp(-x[i]*x[i])-Z[i],0);
         u[i]=0;
         Q[i]=0;
         }
```

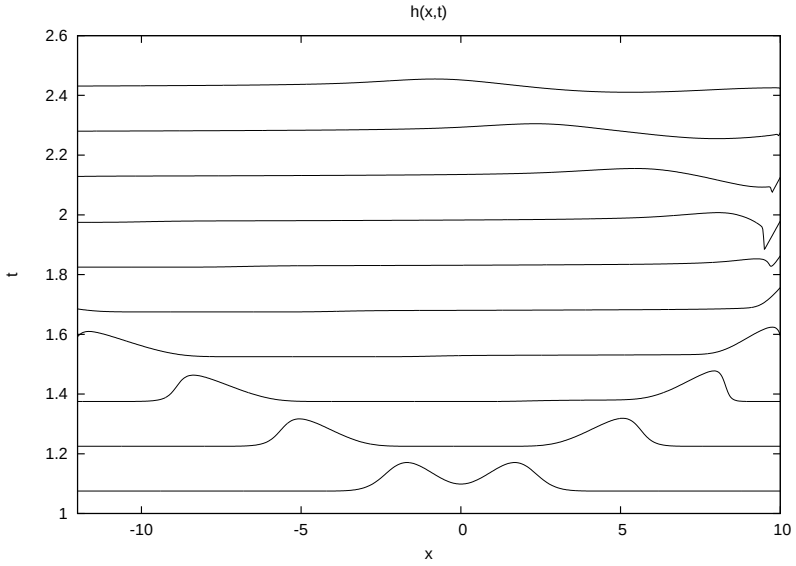


Figure 18: Surface libre, temps de bas en haut. Exemple d'un signal initial: une vague qui se brise en deux, une qui va à gauche, l'autre qui va à droite. Les non linéarités provoquent un raidissement des vagues d'autant plus près de la côte à droite où la vague va mourir.

7.7 Exemple sur une bosse

Cas d'écoulement de fluide parfait (pas de frottement) sur une bosse Z. On peut comparer la solution numérique à la solution analytique linéarisée.

Ce cas est fait dans `http://basilisk.fr/sandbox/M1EMN/Exemples/bump_trans.c`

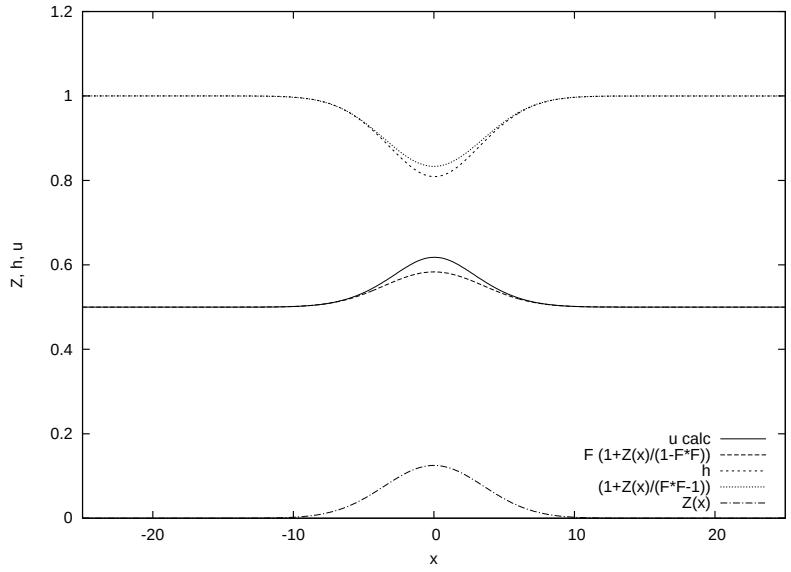


Figure 19: Ecoulement fluvial, sur une bosse. La vitesse augmente sur la bosse puis diminue après.

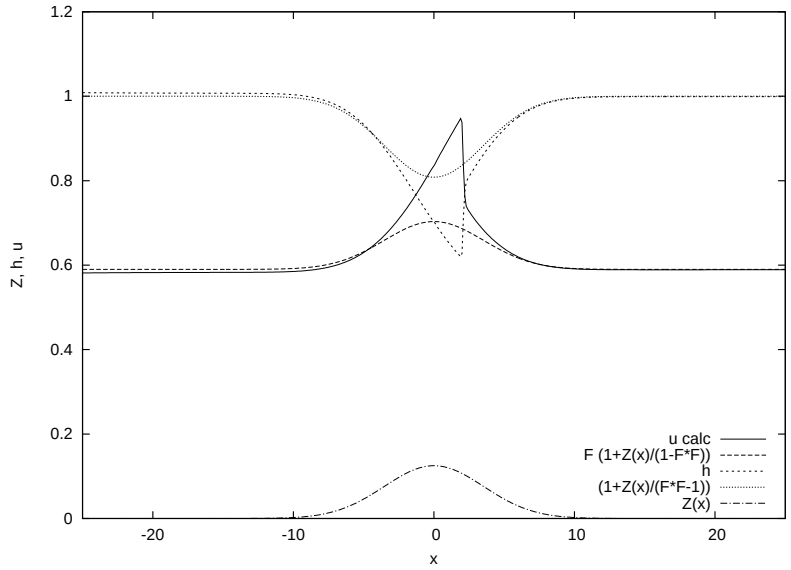


Figure 20: Ecoulement transcritique sur une bosse. Le froude passe à 1 au sommet, l'écoulement devient supercirtique, puis un choc le re fait passer en subcritique

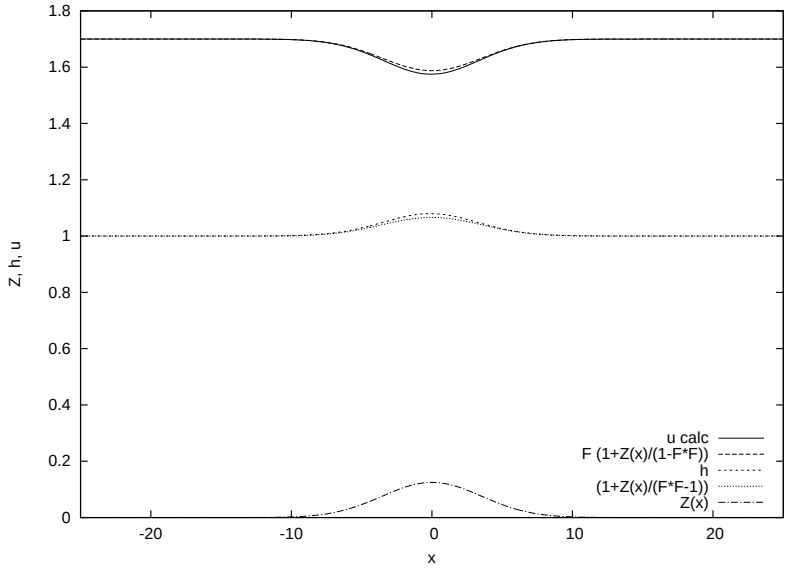


Figure 21: Ecoulement supercritique sur une bosse. La vitesse diminue, la hauteur d'eau augmente.

8 Terme dispersif: Equations de Boussinesq

Il s'agit en fait de Saint-Venant avec un ajout de terme dispersif, elles sont écrites ici sous forme non conservative:

$$\begin{cases} \dfrac{\partial h}{\partial t} + \dfrac{\partial (hu)}{\partial x} & = 0 \\ \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} & = -g\dfrac{\partial h}{\partial x} + \dfrac{h^2}{3}\dfrac{\partial^3 u}{\partial x^2 \partial t} \end{cases}$$

Ces équations permettent de retrouver KdV et le soliton (`http://www.lmm.jussieu.fr/~lagree/COURS/M2MHP/kdv.pdf`) et sont donc plus générales que Saint-Venant.

La forme semi discrétisée du système Boussinesq en conservatif est (en approximant le terme source) :

$$\frac{h^{n+1}-h^n}{\Delta t} + \frac{\partial Q}{\partial x} = 0 \text{ et } \frac{Q^{n+1}-Q^n}{\Delta t} + \frac{\partial}{\partial x}(\frac{Q^2}{h} + \rho g \frac{h^2}{2}) = (\frac{1}{3})h^3 \frac{\partial}{\partial t}\frac{\partial^2 u}{\partial x^2}.$$

On fait un "split": par Saint-Venant on estime h^* et Q^* sans dispersion:

$$\frac{h^*-h^n}{\Delta t} + \frac{\partial Q}{\partial x} = 0$$

$$\frac{Q^*-Q^n}{\Delta t} + \frac{\partial}{\partial x}(\frac{Q^2}{h} + \rho g \frac{h^2}{2}) = 0$$

donc comme $u^* = Q^*/h^*$ et $u^n = Q^n/h^n$, l'équation précédente de la quantité de mouvement $\frac{Q^*-Q^n}{\Delta t} + ...$ peut s'écrire formellement pour la forme non conservative:

$$u^* - u^n = F$$

On ajoute maintenant, par le second split, le terme de dispersion, terme source:

$$u^{n+1} - u^* = \frac{1}{3}(h^2 \frac{\partial}{\partial t}\frac{\partial^2 u}{\partial x^2})\Delta t$$

La variation $(\frac{\partial u}{\partial t}\Delta t)$ y est remplacée par l'accroissement $(u^{n+1}-u^n)$. Ainsi ce terme le dispersion est devenu approximativement $(\frac{1}{3})(h^*)^2\frac{\partial^2}{\partial x^2}(u^{n+1}-u^n)$ l'accroissement final au terme de ce second splitting est

$$u^{n+1} - u^n = F + (1/3)(h^*)^2 \frac{\partial^2}{\partial x^2}(u^{n+1} - u^n)$$

On trouve ainsi de manière implicite l'accroissement temporel de u

$$(1 - (\frac{1}{3})(h^*)^2 \frac{\partial^2}{\partial x^2})(u^{n+1} - u^n) = F,$$

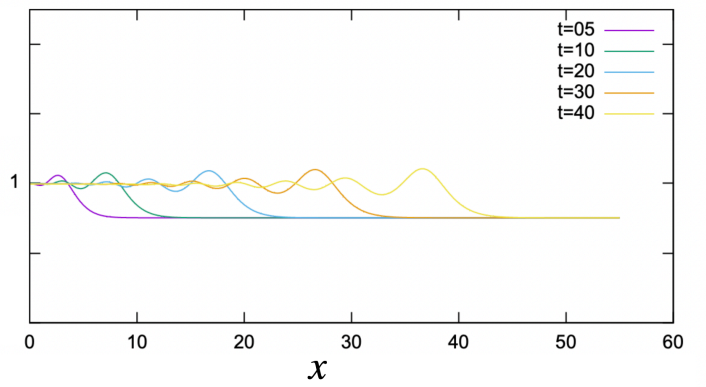


Figure 22: Évolution d'un ressaut en mascaret de gauche à droite: il faut ajouter à Saint-Venant un terme dispersif en terme source $(\frac{1}{3})h^3\frac{\partial}{\partial t}\frac{\partial^2 u}{\partial x^2}$ (ce qui donne les équations de Boussinesq) et résoudre en plus avec frottement $-C_f|u|\frac{u}{h}$.

c'est une dérivée seconde qui se résout par l'algorithme de Thomas pour obtenir $(u^{n+1}-u^n)$ donc u^{n+1}.

Pour tracer la figure 22 on a en plus rajouté un terme de frottement turbulent en rajoutant un "split" (le 1/2 est dans le coefficient):

$$\frac{\partial u}{\partial t} = -C_f|u|\frac{u}{h}$$

Il s'agit donc de la modélisation du mascaret, elle nécessite les termes non linéaires, les termes dispersifs et un peu de dissipation.

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} = 0 \text{ et } \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -g\frac{\partial h}{\partial x} + \frac{h^2}{3}\frac{\partial^3 u}{\partial x^2 \partial t} - C_f|u|\frac{u}{h}$$

La mise en oeuvre numérique est dans `http://basilisk.fr/sandbox/M1EMN/Exemples/ressaut_mascaret.c` elle montre un ressaut qui se transforme en mascaret par la dispersion. Sur la figure 22 on voit le mascaret se propager de gauche à droite, remonter le courant et générer son train de vagues.

9 conclusion

Nous avons présenté une résolution des équations de Saint-Venant en volumes finis bien balancés (telle que présentée par Audusse et Delestre). Cette méthode permet de bien tenir compte de variations de topographie dans la résolution des équations en couche mince en volumes finis... Un code en `C` modifiable est proposé et expliqué, il permet de reproduire les exemples, à noter que c'est cette méthode qui est codée dans le code *Gerris* (directive `GfsRiver`) et dans *Basilisk* fichier `http://basilisk.fr/src/saint-venant.h`.

References

[1] Cours M1 http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf

[2] E. Audusse (2004) Thèse. Modélisation hyperbolique et analyse numérique pour les écoulements en eaux peu profondes http://tel.archives-ouvertes.fr/docs/00/04/75/79/PDF/tel-00008047.pdf

[3] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM J. Sci. Comput., 25(6):2050–2065, 2004.

[4] F. Bouchut. Nonlinear stability of finite volume methods for hyperbolic conservation laws, and well-balanced schemes for sources, volume 2/2004. Birkhäuser Basel, 2004.

[5] Courant, R.; Friedrichs, K.; Lewy, H. (1928), "Über die partiellen Differenzengleichungen der mathematischen Physik", Mathematische Annalen, 100 (1): 32–74, doi:10.1007/BF01448839).

[6] Olivier Delestre (2010) Thèse. Simulation du ruissellement d'eau de pluie sur des surfaces agricoles http://tel.archives-ouvertes.fr/docs/00/56/16/76/PDF/olivier.delestre_1878.pdf

[7] Greenberg, J. M. and LeRoux, A.-Y., A well-balanced scheme for the numerical processing of source terms in hyperbolic equation, SIAM Journal on Numerical Analysis, 1996, v 33, pp 1–16,

[8] Castro, M. J. and Pardo, A. and Parès, C., Well-balanced numerical schemes based on a generalized hydrostatic reconstruction technique, Mathematical Models and Methods in Applied Sciences, 2007, volume = 17, p 2065–2113, n 12,

[9] Noelle, Sebastian and Xing, Yulong and Shu, Chi -Wang, High-order well-balanced finite volume WENO schemes for shallow water equation with moving water, Journal of Computational Physics, 2007, v 226, p 29 - 58, n 1, DOI: 10.1016/j.jcp.2007.03.031, http://www.sciencedirect.com/science/article/B6WHY-4NG3TH4-3/2/546ff3d247ae9db8ed41e3de935ee892

[10] Thanh, M. D. and Fazlul Karim, Md. and Ismail, A. I. Md., Well-balanced scheme for shallow water equations with arbitrary topography, Int. J. Dynamical Systems and Differential Equations, 2008, v 1, p 196-204, n 3

[11] Bouchut, F. and Morales, T., A subsonic-well-balanced reconstruction scheme for shallow water flows, http://www.math.ntnu.no/conservation/2009/032.html, year = 2009, note = Preprint,

[12] D. Euvrard "Résolution numérique des équations aux dérivées partielles" Masson 1988

[13] Randall J. LeVeque. Finite volume methods for hyperbolic problems. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2002.

[14] P. D. Lax, R.D. Richtmyer "Survey of the stability of linear finite difference equations", Communications on Pure and Applied Mathematics · Volume 9, Issue 2 May 1956. https://doi.org/10.1002/cpa.3160090206. (https://math.berkeley.edu/~wilken/228B.S07/LaxRichtmyer.pdf)

[15] Roe Characteristic-Based Schemes For The Euler Equations Ann. Rev. Fluid Mech.1986. 18 : 33745

[16] Toro

[17] http://farside.ph.utexas.edu/teaching/329/lectures/node90.html

[18] http://www.iecn.u-nancy.fr/ sokolows/support/node117.html

[19] http://irfu.cea.fr/Projets/COAST/amr_lecture1.pdf

Liens

https://en.wikipedia.org/wiki/MUSCL_scheme

This course is a part of a larger set of files devoted on Shallow Water and waves in fluids by $\mathscr{P}$.-$\mathscr{Y}$. $\mathscr{L}$agrée

/Users/pyl/macintoshHD/DOKUMENTS/Documents/2011/coursXgranul/codeC_aval/SVSH/RUN_SVSH

The web page of this text is:
http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/cours_MFEA_C_OK/index.html

The web page of the Saint-Venant lecture is:
http://www.lmm.jussieu.fr/∼lagree/COURS/MFEnv/

The last version of this file (February 8, 2024) is on:
http://www.lmm.jussieu.fr/∼lagree/COURS/MFEnv/code_C_saintvenant.pdf

Concepts are in http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/MFEnv.pdf

The C file are on:
http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/cours_MFEA_C_OK/index.html
http://www.lmm.jussieu.fr/∼lagree/COURS/MFEnv/svdb.c
http://basilisk.fr/sandbox/M1EMN/Exemples/svdb.c

The Python files:
http://basilisk.fr/sandbox/M1EMN/PYTHON/shdb.py
http://basilisk.fr/sandbox/M1EMN/PYTHON/svdb.py
http://basilisk.fr/sandbox/M1EMN/PYTHON/flood.py

photo pyl
Raymond Subes "Sans Titre" 1961 (entrée de Jussieu Quai Saint Bernard)