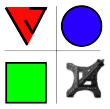

Résolution numérique des équations de Saint-Venant, mise en oeuvre en volumes finis par un solveur de Riemann bien balancé

P.-Y. Lagrée
CNRS & UPMC 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

January 21, 2020

Abstract

Nous montrons comment résoudre par volumes finis les équations de Saint -Venant. Un programme en "C" est expliqué. Des exemples sont ensuite présentés.

page ouèbe de ce fichier `http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/code_C_saintvenant.pdf`

1 Rappel des équations

1.1 Rappel des équations

Le système que nous considérons est celui des Equations de Saint Venant (CRAS 1871 Adhémar Jean-Claude Barré de Saint- Venant, *Shallow Water*):

$$\begin{cases} \partial_t h + \partial_x Q = 0 \\ \partial_t Q + \partial_x \left[\Gamma \dfrac{Q^2}{h} + g \dfrac{h^2}{2} \right] = -gh\partial_x Z - \dfrac{\tau}{\rho} \end{cases} , \tag{1}$$

avec $h(x,t)$ est la hauteur d'eau, $u(x,t)$ la vitesse moyenne, $Q(x,t) = h(x,t)u(x,t)$ le flux, Γ facteur de forme, τ frottement 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 et le frottement au fond dépendent du choix du profil de base.

• Pour les canaux et fleuves, en turbulent, faute de mieux, 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 (Darcy Weissbach) avec $\beta = 7/3$, on a la loi de Manning-Strickler.

• Pour de l'eau en couche mince, en laminaire (autour d'un Poiseuille)

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

• Pour la lave en laminaire (autour d'un Poiseuille)

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

code `http://wwwobs.univ-bpclermont.fr/lmv/perso/Kelfoun_Karim/VolcFlow/VolcFlow.html`

• Pour un granulaire. Les granulaires sont les milieux constitués 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. On s'inspire du frottement de Coulomb liant la force tangentielle τ à la force normale ρgh. On définit un coefficient de friction μ peut être variable et fonction de $I = (d/h)5Q/2h/\sqrt{gh}$ (nombre sans dimension que l'on peut constuire). Il est compris entre 0.2 et 0.4 suivant les matériaux et $\tau = \mu\rho gh$ Le profil de base est calculable (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 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)

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

• Pour des fluides complexes, (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

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

• Panache: Les écoulements de type panache se résolvent avec les équations de couche limite intégrées sur leur épaisseur. On retrouve alors des équatiosn 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 éuqtaions. 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...).

1.2 Equation de bilan locale

On peut écrire (1) sous une forme compacte

$$\partial_t U + \partial_x F(U) = S(U), \tag{2}$$

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

$$U = \begin{pmatrix} h \\ Q \end{pmatrix}, \; F(U) = \begin{pmatrix} Q \\ \frac{Q^2}{h} + g\frac{h^2}{2} \end{pmatrix} \tag{3}$$

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

$$S(U) \quad = \begin{pmatrix} 0 \\ -gh\partial_x Z - \tau \end{pmatrix}. \tag{4}$$

1.3 Linéarisation des équations

Si on résout l'équation avec linéarisée 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 $c_0^2 = gh_0$:

$$\left\{ \begin{array}{l} \partial_t \eta + \partial_x q = 0 \\ \partial_t q + c_0^2 \partial_x \eta = 0 \end{array} \right. , \tag{5}$$

elle se met 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 caratéristique cette équation avec une advection vers la droite et une autre ver la gauche

$$\left\{ \begin{array}{l} \partial_t (c_0\eta + q) + c_0 \partial_x (c_0\eta + q) = 0 \\ \partial_t (c_0\eta - q) - c_0 \partial_x (c_0\eta - q) = 0 \end{array} \right. , \tag{6}$$

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

$$c_0\eta(x,t) + q(x,t) = 2f(x - c_0 t) \text{ et } c_0\eta(x,t) - q(x,t) = 2g(x + c_0 t)$$

et 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 ainsi

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

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

$$f(x - c_0 t) = c_0\eta_0(x - c_0 t) + \int_{x-c_0t}^0 \eta_1(\xi)d\xi \text{ et } g(x + c_0 t) = c_0\eta_0(x + c_0 t) + \int_0^{x+c_0t} \eta_1(\xi)d\xi$$

la somme nous donne la relation

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

mais comme par définition $\partial_t \eta = -\partial_x q$ donc $\eta_1(x)$ est disons $-\partial_x q_0(x)$

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

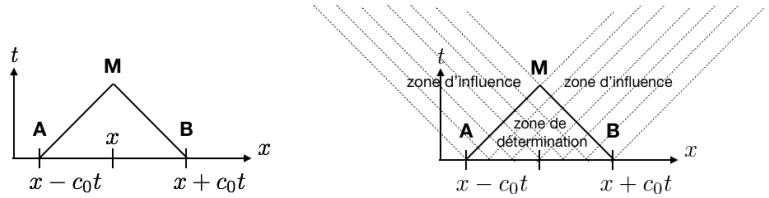

Figure 1: A gauche, la solution au point M en x et $t > 0$ est déterminée par une partie des données, en l'occurence η_0 en A et en B et $\eta_1 = \partial_t \eta_0$ sur le segment AB. A gauche on voit les zones influencées et déterminées par les données η_0 et et η_1 dans le temps futur $t > 0$

ensuite quand on résout

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

on a

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

la forme du flux en fonction des quantités conservées le long de $x \pm c_0t$:

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

Cette forme nous servira pour construire des approximations du flux aux faces de nos domaines.

1.4 "Split": décomposition convection-frottement

Pour résoudre (2) il est d'usage de décomposer en deux étapes (on "coupe" S en plusieurs parties, c'est le *splitting*). Cela renvoit à la discrétisation $\partial_t U = (U^{n+1} - U^n)/\Delta t$

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

est donc

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

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

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

La première étape est une étape conservative,

$$(U_1 - U^n)/\Delta t + \partial_x F(U^n) = 0,$$

la seconde avec le premier terme source,

$$(U_2 - U_1)/\Delta t = S_Z(U_1),$$

la troisième avec le deuxième terme source:

$$(U^{n+1} - U_2)/\Delta t = S_f(U_2),$$

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

$$(U^{n+1} - U^n)/\Delta t = (U^{n+1} - U_2)/\Delta t + (U_2 - U_1)/\Delta t + (U_1 - U^n)/\Delta t = -\partial_x F(U^n) + S_Z(U_1) + S_f(U_2).$$

1.5 Matrice Jacobienne

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

$$\partial_t U + \partial_x F(U) = 0. \tag{7}$$

On dérive $\partial_x F(U) = \partial_U F(U) \cdot \partial_x U$:

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

la matrice Jacobienne $J(U)$ a deux valeurs propres: c'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.

$$\lambda_1 = \frac{Q}{h} - \sqrt{gh} = u - c \quad \text{and} \quad \lambda_2 = \frac{Q}{h} + \sqrt{gh} = u + c. \tag{8}$$

on reconnaît c la vitesse de propagation des ondes :

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

2 Résolution numérique, cas fluide parfait

2.1 Splitting: étape convective

L'espace est découpé en petits segments de longueur quelconque *a priori*, mais ici nous supposerons un pas Δx constant pour alléger les notations, de même pour le temps, le pas de temps sera Δt. Ces petits segments, sont les petits "volumes", car nous sommes en dimension un. Prenons le système (2) (sans source $S(U) = 0$) et intégrons à la fois en x sur un intervalle entre x_i et x_{i+1} et en temps de t^n à t^{n+1}.

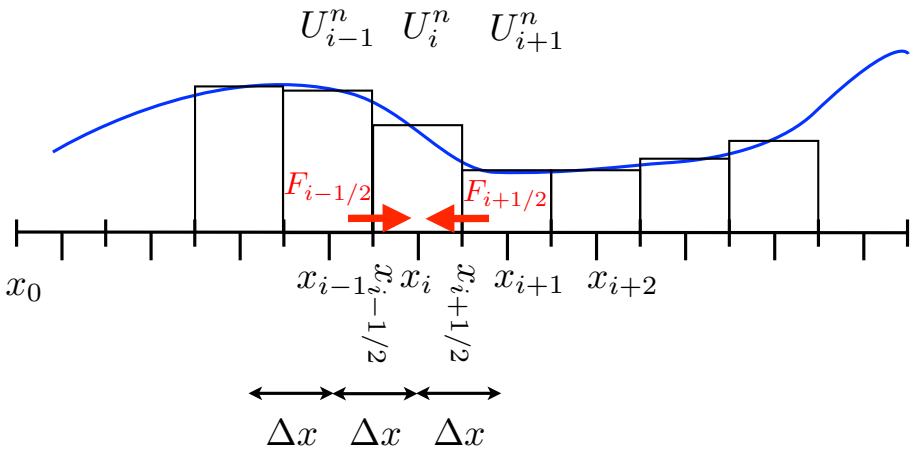

Figure 2: Le signal continu U, approximé par U^n_i en x_i. Le flux aux faces du petit volume centré en x_i et de longueur Δx est $F_{i-1/2}$ à gauche et $F_{i+1/2}$ à droite

Comme, par développement de Taylor:

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

et en posant U^n_i 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^n_i est la valeur moyenne:

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

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 t_n avec $t_{n+1} - t_n = \Delta t$. Et comme par intégration sur le "segment/ Volume" on a:

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

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 (3) à l'interface droite du segment C_i en $i + 1/2$. C'est donc 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 $i + 1$:

$$F_{i+1/2} = \mathcal{F}(U_i, U_{i+1}). \tag{9}$$

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. On obtient le schéma de volumes finis au premier ordre (voir la figure 2) que l'on écrit sous la forme suivante (LeVeque 02 p65 eq (4.4))):

$$\frac{U^{n+1}_i - U^n_i}{\Delta t} + \frac{F^n_{i+1/2} - F^n_{i-1/2}}{\Delta x} = 0, \tag{10}$$

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

2.2 Consistance, stabilité, convergence

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

La méthode doit être "consistante" avec l'équation différentielle, 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, R.; Friedrichs, K.; Lewy, H. (1928), "Über die partiellen Differenzengleichungen der mathematischen Physik", Mathematische Annalen, 100 (1): 32–74, doi:10.1007/BF01448839).

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. P. D. Lax, R.D. Richtmyer, "Survey of the stability of linear finite difference equations", Comm. Pure Appl. Math., vol. 9, 1956, p. 267-293 (DOI 10.1002/cpa.3160090206, (`https://math.berkeley.edu/~wilken/228B.S07/LaxRichtmyer.pdf`)

Malgré l'invocation du mot "théorème", la suite de ces notes est très empirique.

2.3 Implémentation en C

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.

On fait donc une boucle en temps

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

On va donc utiliser des pointeurs h et u tels que leur produit soit le flux: Q[i]=h[i]*un[i]; ces pointeurs sont définis par

```
  double*x=NULL,*h=NULL,*u=NULL;
  double*un=NULL,*hn=NULL;
  nx=160;
  x= (double*)calloc(nx+1,sizeof(double));
  h= (double*)calloc(nx+1,sizeof(double));
  u= (double*)calloc(nx+1,sizeof(double));
  un=(double*)calloc(nx+1,sizeof(double));
  hn=(double*)calloc(nx+1,sizeof(double));
```

h et u est celui en cours, hn et un seront les nouveaux calculé 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 (10):

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

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] sont les deux composantes première et deuxième du flux $F_{i+1/2}$. 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;i++)
    {   h[i]=hn[i];
        u[i]=un[i];
        Q[i]=hn[i]*un[i];
    }
```

2.4 Flux Numérique

Il faut maintenant trouver la bonne approximation de $\mathcal{F}$. Le plus simple serait de prendre la moyenne des valeurs (LeVeque [6] chapitre 4, Finite Volume methods, voir aussi Roe [7])

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} \tag{11}$$

d'où

$$U_i^{n+1} = U_i^n - \Delta t \frac{F(U_{i+1}) - F(U_{i-1})}{2\Delta x} \tag{12}$$

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 le schéma naïf de la partie 1). On va voir une autre technique classique pour stabiliser "Lax-Friedrich". Mais ensuite nous passerons à une méthode plus efficace (flux dépendants des valeurs propres du sytème, Rusanov et HLL).

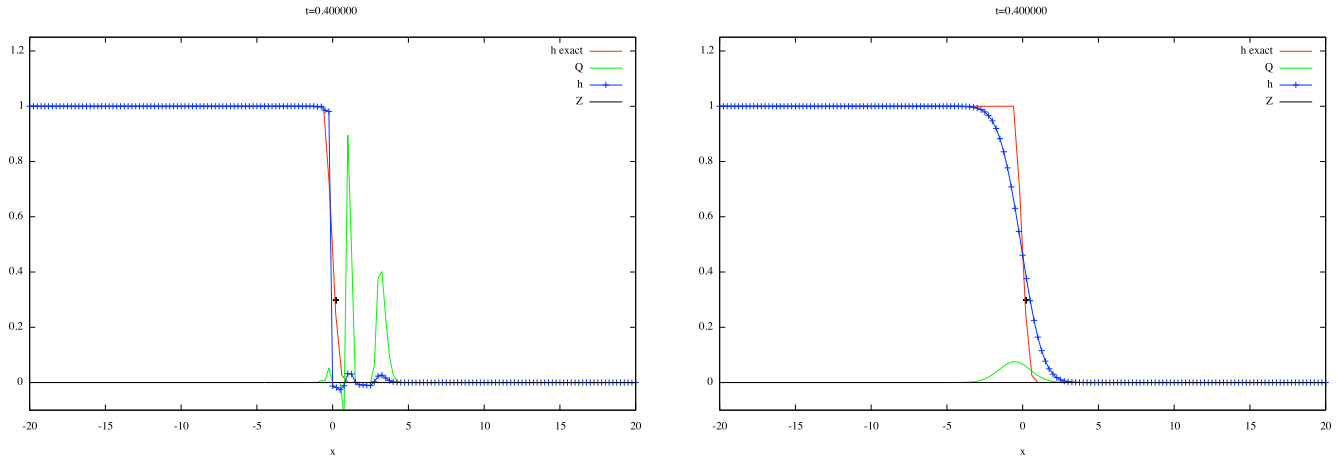

Figure 3: Départ de rupture de barrage au temps sans dimension $t = 0.4$, la solution exacte $h(x, t = 0.4$ est tracée en rouge. A gauche, cas simple du flux $\mathcal{F}(U_{i-1}, U_i) = (F(U_{i-1}) + F(U_i))/2$, on voit des oscillations amplifiées pour Q en vert et h en bleu calculés au même instant. 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).

Si on remplace U_i^n par une moyenne $(U_{i+1}^n + U_{i-1}^n)/2$, on obtient le schéma de Lax-Friedrich

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

qui a le mérite d'introduire une diffusion numérique qui tue l'instabilité, en effet en repassant au développement de Taylor:

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

on obtient un terme de diffusion pour l'équation:

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

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

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} - c_\Delta \frac{((U_i) - (U_{i-1}))}{2} \text{ avec } c_\Delta = \frac{\Delta x}{\Delta t} \tag{13}$$

On va voir que cette forme est presque la bonne si on prend un c_Δ mieux adapté. Plusieurs flux, peuvent être employés, nous utiliserons les plus simples.

justification du flux dans la cas linéaire simple

Si on résout l'équation linéarisée

$$\begin{cases} \partial_t \eta + \partial_x q = 0 \\ \partial_t q + c_0^2 \partial_x \eta = 0 \end{cases}, \tag{14}$$

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 $x \pm c_0 t$:

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

donc la discrétisation en $(x \pm c_0 t)$ va chercher les points en $i \pm 1$ de manière approximative, donc pour la discrétisation

$$U_i^{n+1} = U_i^n - \Delta t \frac{F(U_{i+1}) - F(U_{i-1})}{2\Delta x} \tag{15}$$

l'approximation du flux 13

$$\mathcal{F}(U_{i-1}, U_i) = \frac{F(U_{i-1}) + F(U_i)}{2} - c_\Delta \frac{((U_i) - (U_{i-1}))}{2} \text{ avec ici } c_\Delta = c_0 \tag{16}$$

L'interprétation de cette forme de flux 13 se retrouve aussi dans les relations de ressaut, puisque l'on a bien des discontinuités qui vont se transmettre d'une case à l'autre. Or, on a vu que la conservation de la masse et de quantité de mouvement au travers d'une discontinuité :

$$Q_2 - Q_1 - W(h_2 - h_1) = 0$$

$$\big(Q_2 h_2 - \rho g \frac{h_2^2}{2}\big) - \big(Q_1 h_1 - \rho g \frac{h_1^2}{2}\big) - W(Q_2 - Q_1) = 0$$

2.5 Cas simple $U = h$, $F = ah$

Pour fixer les idées, dans le cas de l'équation d'advection simple

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

la valeur propre est a tout simplement, si on utilise le flux simple (equivalent de 11)

$$F_{i-1/2} = a \frac{h_{i-1} + h_i}{2} \tag{17}$$

d'où

$$h_i^{n+1} = h_i^n - \Delta t \frac{a(h_{i+1} - h_{i-1})}{2\Delta x} \tag{18}$$

la méthode est instable, on peut le vérifier avec la méthode de stabilité de Neuman en écrivant $h = h_n e^{ikx}$ et $h_{n+1} = G h_n$ 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-Friedrich (equivalent de 13):

$$F_{i-1/2} = a \frac{h_{i-1} + h_i}{2} - c_\Delta (\frac{h_i - h_{i-1}}{2}) \tag{19}$$

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

$$h_i^{n+1} = h_i^n - \Delta t \frac{a(h_{i+1} - h_{i-1})}{2\Delta x} + \Delta t (c_\Delta \Delta x) \frac{(h_{i+1} - 2h_i + h_{i-1})}{2\Delta x^2} \tag{20}$$

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

$$\partial_t h + \partial_x(ah) = (a c_\Delta \Delta x) \partial_x^2(h)$$

Une autre manière de stabiliser correspond à un autre choix de c_Δ, si on prend $c_\Delta = a$, c'est le flux *upwind*:

$$F_{i-1/2} = a \frac{h_{i-1} + h_i}{2} - (\frac{a}{2})(h_i - h_{i-1}) \tag{21}$$

d'où

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

et le nouvel h

$$h_i^{n+1} = h_i^n - \Delta t \frac{a(h_i - h_{i-1})}{\Delta x} \tag{22}$$

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

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. La bonne méthode est expliquée dans le cours lorsque l'on parlé des caractéristiques.

2.6 Flux de Rusanov

Le flux le plus simple ayant les bonnes propriétés est le flux de Rusanov [3]

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

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

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

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

D'autres flux seront considérés tels que le flux HLL avec une onde de détente inspirée de l'éventail de détente (par exemple de la rupture de barrage)

2.7 HLL flux

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

$$\mathcal{F}(U_G, U_D) = \begin{cases} F(U_G) & \text{if } 0 \le c_1 \\ \dfrac{c_2 F(U_G) - c_1 F(U_D)}{c_2 - c_1} + \dfrac{c_1 c_2}{c_2 - c_1}(U_D - U_G) & \text{if } c_1 < 0 < c_2 \\ F(U_D) & \text{if } c_2 \le 0 \end{cases}, \tag{23}$$

avec

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

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

2.8 Implémentation en C

L'équation

$$F_{i-1/2} = \mathcal{F}(U_{i-1}, U_i) \tag{24}$$

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

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

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:

$$\mathcal{F}(U_G,U_D) = \begin{cases} F(U_G) & \text{if } 0 \le c_1 \\ \dfrac{c_2F(U_G)-c_1F(U_D)}{c_2-c_1} + \dfrac{c_1c_2}{c_2-c_1}(U_D-U_G) & \text{if } c_1 < 0 < c_2 \\ F(U_D) & \text{if } c_2 \le 0 \end{cases}, \tag{25}$$

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);} }
```

2.9 Exemples de runs

Une mise en oeuvre de ces solutions est le problème de rupture de barrage. On a vu 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 $x = 0$.

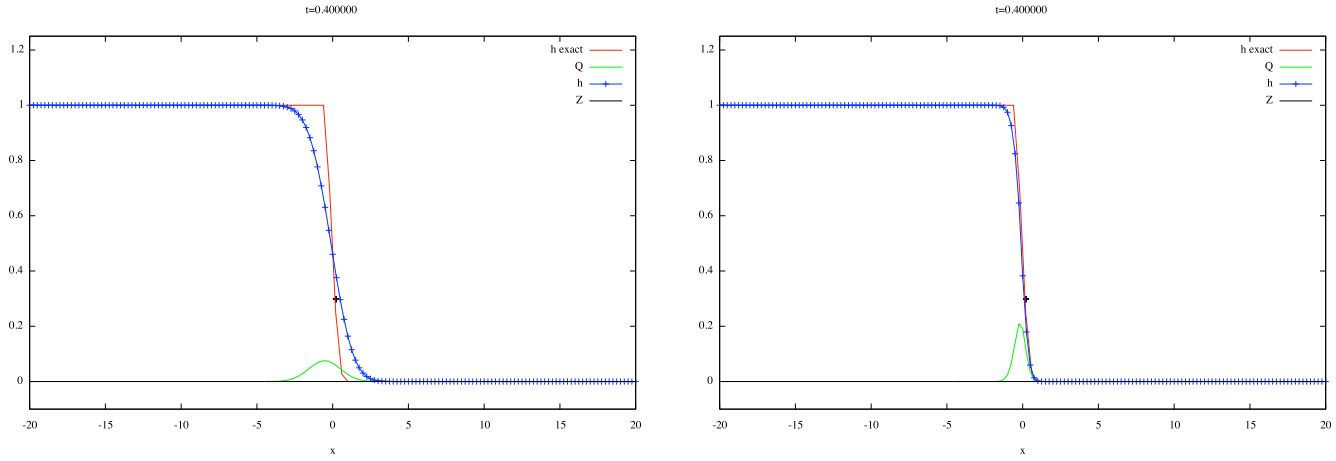

Figure 4: Départ de rupture de barrage. A gauche le schéma de Lax, à droite, au même instant le schéma de Rusanov est plus proche de la solution exacte en rouge de par la caractère trop diffusif de Lax.

Le programme est donné maintenant.

2.10 Programme

Le programme dans le fichier `svdb.c` est

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

//A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows
// Emmanuel Audusse, Franccois Bouchut, Marie-Odile Bristeau, Rupert Klein and Benoıt Perthame
// biblio : Delestre These
// PYL
    double*x=NULL,*h=NULL,*u=NULL,*Q=NULL;
    double t,dt,tmax,dx,Z;
    int nx;

double FR1(double ug,double ud,double hg,double hd)
 { double c;
 //Rusanov
   c=fmax(fabs(ug)+sqrt(hg),fabs(ud)+sqrt(hd));
 // c=1*dx/2/dt;
   return (hg*ug+hd*ud)*0.5-c*(hd-hg)*0.5;
 }
double FR2(double ug,double ud,double hg,double hd)
 { double c;
 //Rusanov
   c=fmax(fabs(ug)+sqrt(hg),fabs(ud)+sqrt(hd));
 //  c=1*dx/2/dt;
   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+1,sizeof(double));
      h= (double*)calloc(nx+1,sizeof(double));
      Q= (double*)calloc(nx+1,sizeof(double));
      u= (double*)calloc(nx+1,sizeof(double));
      fp=(double*)calloc(nx+1,sizeof(double));
```

```
      fd=(double*)calloc(nx+1,sizeof(double));
      un=(double*)calloc(nx+1,sizeof(double));
      hn=(double*)calloc(nx+1,sizeof(double));


 fprintf(stderr,"tmax= %lf \n",tmax);

// initialisation cond init ----------------------------
  for(i=0;i<=nx;i++)
    {  x[i]=(i-nx/2)*dx;
   //dambreak
     Z=0;
     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;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;i++)
    {
      hn[i]=h[i]- dt*(fp[i+1]-fp[i])/dx;   //conservation de la masse
  if(h[i]>0.){                             //conservation qunatité 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]=hn[nx-1];
        un[nx]=un[nx-1];

   //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;
}
```

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` un fichier appelé solxhQt.OUT avec x h Q et t est créé. pour le tracer, on lance gnuplot

```
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.

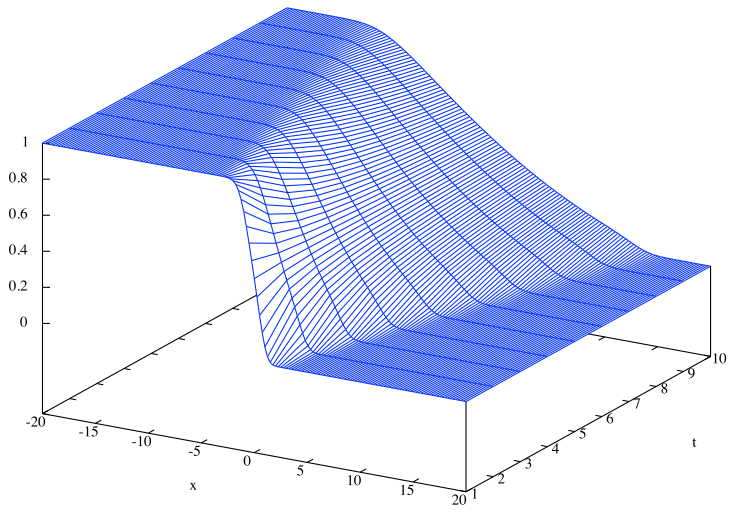

Figure 5: Rupture de barrage. Schéma de Rusanov, la hauteur d'eau en fonction de x et de t.

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.

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

3 Résolution numérique, influence du fond

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

$$\partial_t h = \partial_t u = \partial_t Q = 0. \tag{26}$$

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

$$\left\{ \begin{array}{l} Q = Q_0 \\ \dfrac{{Q_0}^2}{2gh^2} + Z + h = Cst \end{array} \right. , \tag{27}$$

des schémas numériques existent qui préservent cet équilibre, mais ils sont difficiles à mettre en oeuvre (voir [**?**], [**?**], [**?**] et [**?**]). Pour simplifier, on cherche à préserver uniquement les états d'équilibres

$$\left\{ \begin{array}{l} q = u = 0 \\ \partial_x \left(g\dfrac{h^2}{2} \right) + gh\partial_x Z = 0 \end{array} \right. , \tag{28}$$

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 (28) se réduit à

$$H = h + Z = Cst, \tag{29}$$

où H est le niveau d'eaus the water level. Depuis Greenberg et al. [**?**], les schémas qui préservent (28) sont appelés "schémas bien balancés" *well-balanced schemes*.

3.1 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

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = -gh_i \frac{Z_{i+1/2} - Z_{i+1/2}}{\Delta x} \tag{30}$$

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?), s i 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 ([1],[2]) propose

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

ainsi que

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

3.2 Bien balancée

La méthode *well-balanced method* inspirée de la reconstruction hydrostatique ([2] et [3]), le terme de friction est toujours absent, et on construit un schéma qui préserve les états stationnaires (dont celui du lac au repos (29)), comme suggéré par Audusse et al. [2], on définit:

$$\begin{cases} h_{i+1/2G}=max(h_i+Z_i-max(Z_i,Z_{i+1})),\\ U_{i+1/2G}=(h_{i+1/2G},h_{i+1/2G}u_i)\\ h_{i+1/2D}=max(h_{i+1}+Z_{i+1}-max(Z_i,Z_{i+1})),\\ U_{i+1/2D}=(h_{i+1/2D},h_{i+1/2D}u_i) \end{cases}, \tag{31}$$

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:

$$U_i^{n+1}=U_i^n-\frac{\Delta t}{\Delta x}(F_{i+1/2G}^n-F_{i-1/2D}^n), \tag{32}$$

avec

$$\begin{array}{l} F^n_{i+1/2G} = F^n_{i+1/2} + S_{i+1/2G} \\ F^n_{i-1/2D} = F^n_{i-1/2} + S_{i-1/2D} \end{array}, \tag{33}$$

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

$$\begin{array}{l} S_{i+1/2G} = \begin{pmatrix} 0 \\ \mathcal{P}(h^n_i) - \mathcal{P}(h^n_{i+1/2G}) \end{pmatrix} \\ S_{i-1/2D} = \begin{pmatrix} 0 \\ \mathcal{P}(h^n_i) - \mathcal{P}(h^n_{i-1/2D}) \end{pmatrix} \end{array} \tag{34}$$

Le flux numérique $F^n_{i+1/2}$ est reconstruit (31):

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

3.3 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", ((31))

```
 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));
}
```

4 Résolution numérique, Frottement

4.1 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

$$\left(\frac{U_i^{n+1} - U_i^*}{\Delta t}\right) = S_f(U^{n+1}), \tag{35}$$

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 exxemple)

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

et voilà.

4.2 Cas Poiseuille

Dans le cas Poiseuille, $\tau = 3\nu Q/h^2$, 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, attention ne pas oublier de le déclarer!):

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

4.3 Cas Cas trubulent

Dans le cas turbulent , $\tau = C_f u^2$, 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 $C_f = 3\nu$ coefficient associé au frottement, attention ne pas oublier de le déclarer!):

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

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

4.4 Terme Source de frottement en C cas granulaire

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

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

donc

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

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

$$u^{n+1} = u^n - \Delta t g \mu$$

comme la vitesse décroit et ne peut être inférizeure à 0, donc $|u|^{n+1} = max(|u|^{n+1}, 0)$.

5 Les conditions aux limites

Boundary Conditions and Ghost Cells chap7 LeVeque 02 ...

6 Exemples de résolution

6.1 Exemples de Ressaut Fixe

C'est un écoulement de fluide parfait (pas de frottement) sur un fond plat. Le programme calcule des ressauts (figure 6) 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.

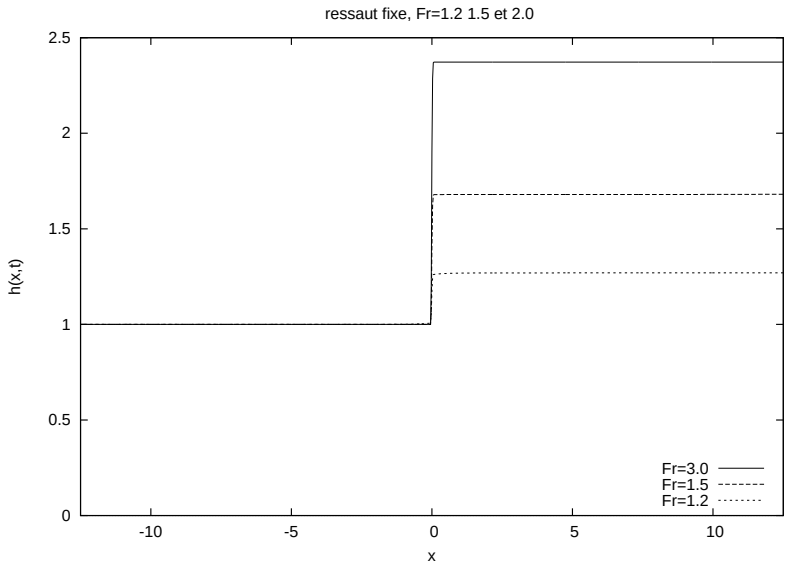

Figure 6: Exemples de ressauts fixes, la solution numérique est confondue avec la solution anlaytique, $h_1 = 1$ $h_2 = (-1+\sqrt{1+8Fr^2})/2-1$

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
    }
```

6.2 Exemple de propagation d'une vague

Calcul de départ d'une vague 7, écoulement de fluide parfait (pas de frottement) sur un fond plat.

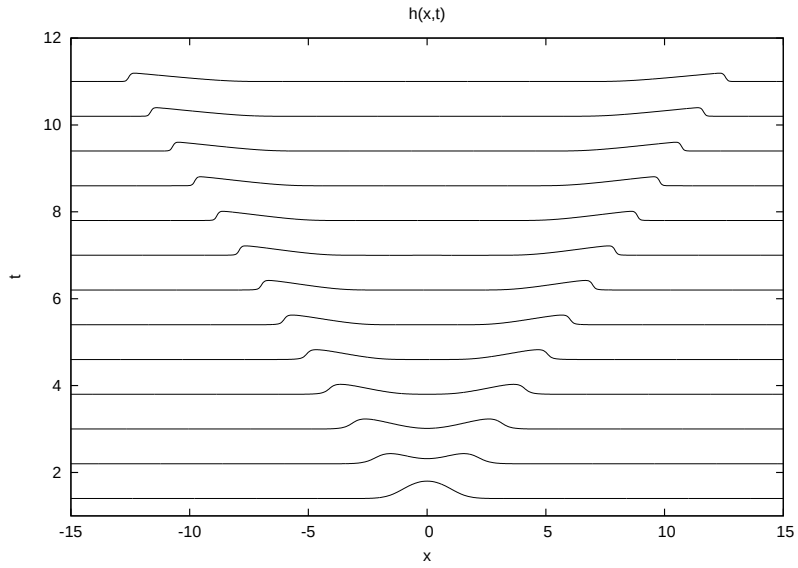

Figure 7: Exemple d'un signal initial: une vague qui se brise en deux, une qui remonte, l'autre qui va à droite. Les non linéarités provoquent un raidissement des vagues.

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

Calcul de départ d'une vague 8, écoulement de fluide parfait (pas de frottement) sur un fond plat. Le fond est

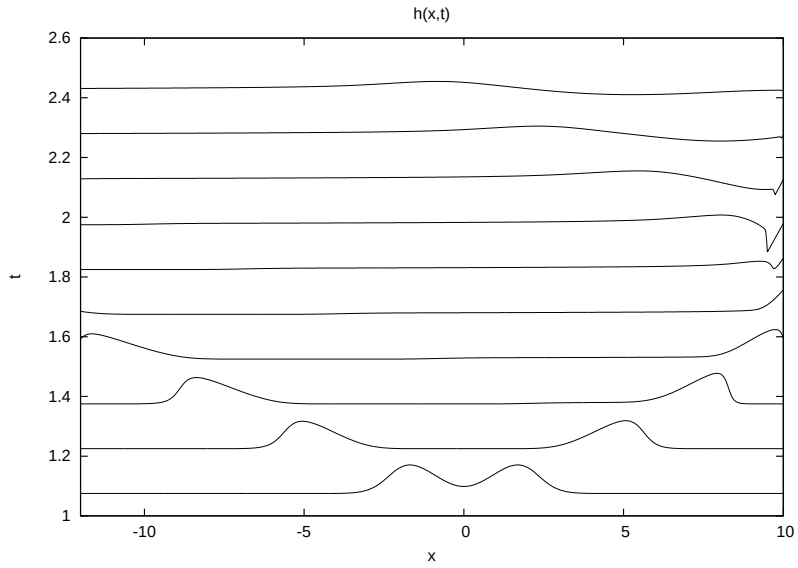

Figure 8: Exemple d'un signal initial: une vague qui se brise en deux, une qui va à gauche, l'autre qui va à droite. Les non linéarités provoquent un raidissement des vagues d'autant plus près de la côte à droite où la vague va mourir.

en 0, 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;
         }
```

6.4 Exemple de rupture de barrage

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 $x > 0$.

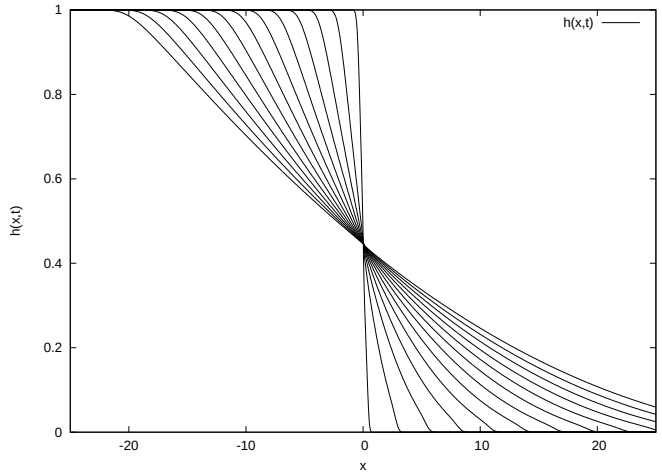

Figure 9: Rupture de barrage, la surface libre à différents temps

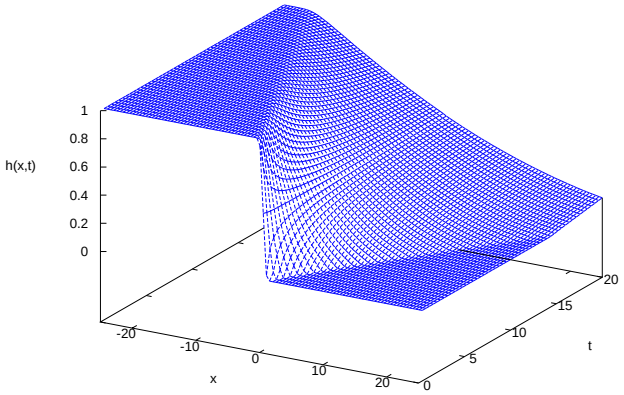

Figure 10: Rupture de barrage

6.5 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.

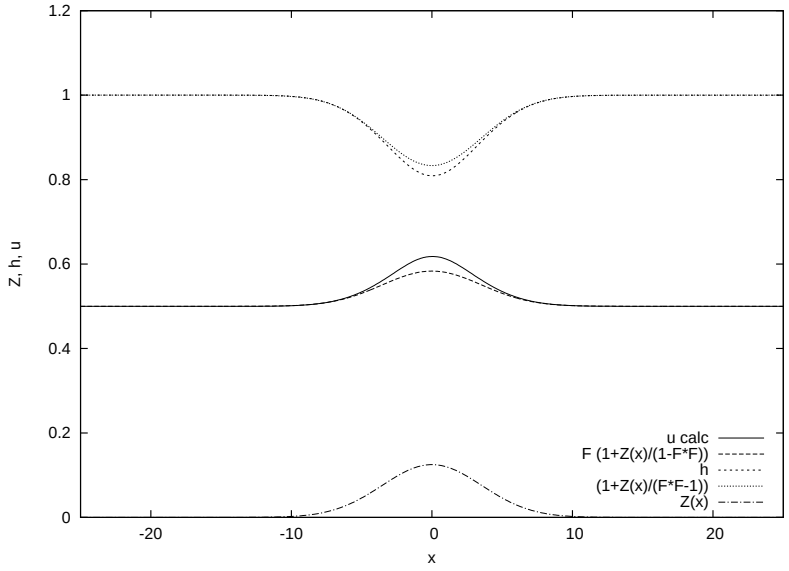

Figure 11: Ecoulement fluvial, sur une bosse. La vitesse augmente sur la bosse puis diminue après.

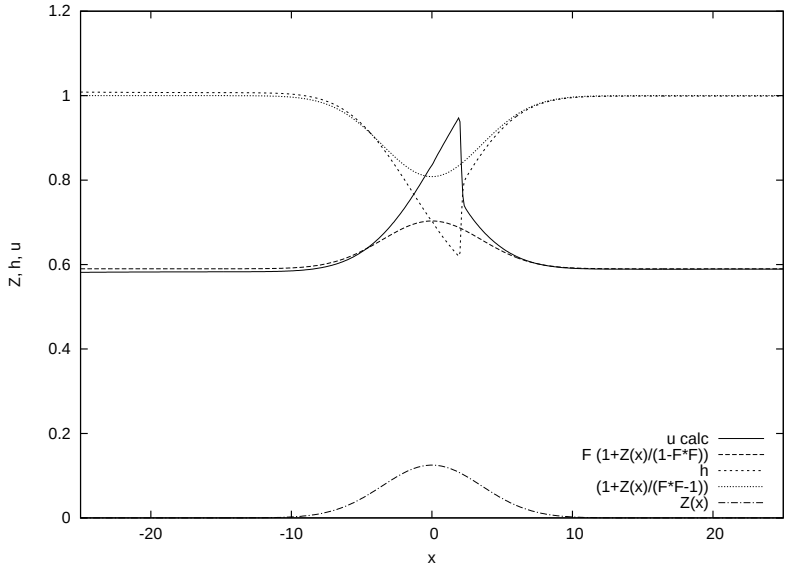

Figure 12: Ecoulement transcritique sur une bosse. Le froude passe à 1 au sommet, l'écoulement devient supercirtique, puis un choc le re fait passer en subcritique

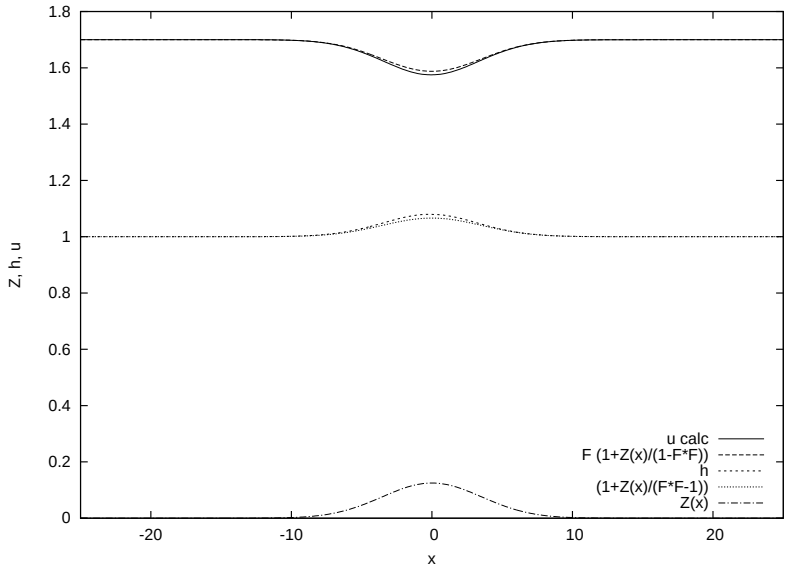

Figure 13: Ecoulement supercritique sur une bosse. La vitesse diminue, la hauteur d'eau augmente.

6.6 Cas frottant: tas de Huppert

Il s'agit de l'effondrement d'un tas visqueux sur un sol horizontal. Comme l'écoulement est lent, on rappelle que l'on peut négliger l'inertie:

$$\left\{ \begin{array}{l} \partial_t h + \partial_x Q = 0 \\ \partial_x \left[\dfrac{h^2}{2} \right] = -C_f \dfrac{|Q|}{h^2} \end{array} \right. , \tag{36}$$

avec $C_f = 3\nu/g$. Ce système se réduit encore en

$$\partial_t h + \partial_x^2 \left[\frac{h^4}{2C_f} \right] = 0$$

On en cherche une solution semblable par invariance $h^*/T^* = (h^*)^3 h^*/X^*/X^*$ et $h^* X^* = 1$, donc

$$(h^*)^{3/2} \sqrt{T^*} = 1/h^*$$

ce qui donne $h^* = T^{*-2/5}$ et $X^* = T^{*2/5}$ la forme est donc

$$h = \frac{H\left(\frac{x}{\sqrt[5]{t}}\right)}{\sqrt[5]{t}}$$

$$-5C_f H(\eta)^3 H''(\eta) - 15C_f H(\eta)^2 H'(\eta)^2 - \eta H'(\eta) - H(\eta) = 0$$

$$\sqrt[3]{\frac{3}{10}} \sqrt[3]{\frac{1}{C_f}} \sqrt[3]{2 - x^2}$$

6.7 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

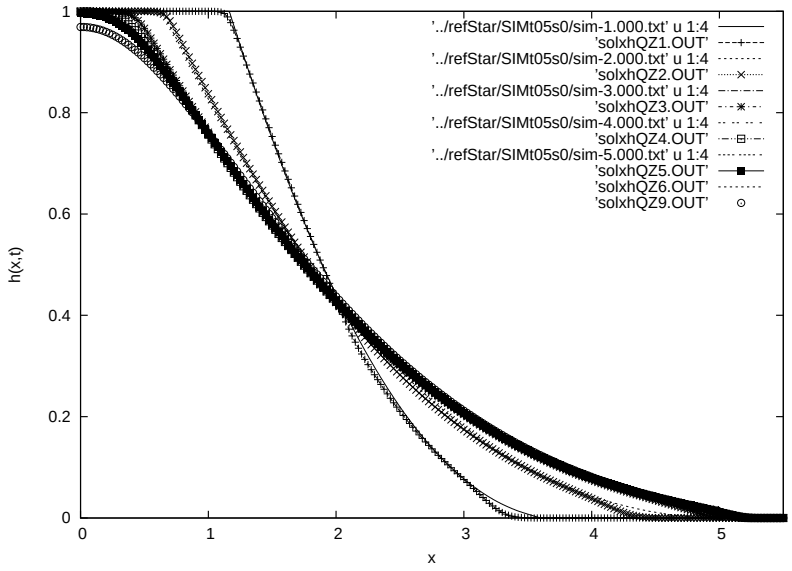

Figure 14: Ecoulement de granulaire, effondrement d'un tas de longueur 2 et de hauteur 1, comparaison du calcul *Gerris* `GfsRiver` et ce code, les résultats sont quasi superposés (la différence provient de la différence de traitement de flux)

```
#####
# 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
##########
```

7 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`).

7.1 Annexe

Dérivée troisième

$$u(x+h) = u(x)+hu'(x)+\frac{1}{2}h^2u'(x)+\frac{1}{6}h^3u'''(x)+\frac{1}{24}h^4u''''(x) \text{ et } u(x+2h) = u(x)+2hu'(x)+2h^2u'(x)+\frac{4}{3}h^3u'''(x)+...$$

donc

$$(u(x-2h) - 2u(x-h) + 2u(x+h) - u(+2h)) = (8-2)/3h^3u''' + ... = 2h^3u''' + ...$$

References

[1] 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`

[2] 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.

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

[4] 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`

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

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

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

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

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

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

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/`

The last version of this file (January 21, 2020) is on:
`http://www.lmm.jussieu.fr/~lagree/COURS/MFEnv/code_C_saintvenant.pdf`

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

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