Ecriture d'un code C++ pour la simulation en hydrologie

Olivier Delestre

Master "Compétence Complémentaire en Informatique"

Ecriture d'un code C++ pour la simulation en hydrologie

Olivier DELESTRE

29 août 2008

Encadrant : Frédéric DARBOUX
Enseignant référent : Frédéric LOULERGUE

Stage à l'INRA Orléans – Science du Sol du 14 avril au 29 août 2008

Table des matières

Introduction

Le ruissellement est le phénomène d'écoulement des eaux à la surface des sols(par opposition au phénomène d'infiltration).L'eau de ruissellement entraîne avec elle des particules plus ou moins grosses en fonction de la quantité d'eau en mouvement et de la pente, ce qui peut avoir un effet abrasif sur le terrain. La généralisation des sols imperméabilisés (routes, stationnement automobile, zones bâties, etc.) augmente le ruissellement aux dépens de l'infiltration, ce qui peut conduire à des crues violentes et augmente les risques de saturation des collecteurs d'eau et d'inondation en aval. On doit donc prendre en compte ce phénomène dans l'aménagement urbain. Le ruissellement est aussi un facteur d'aggravation des pollutions liées à l'agriculture : les engrais et autres produits de traitement sont entraînés vers les cours d'eau, puis vers la mer.

Le ruissellement mérite donc que l'on s'y intéresse. Nous essayons de simuler le ruissellement d'eau de pluie sur des surfaces agricoles. Pour ce type d'écoulement, où la couche d'eau est peu épaisse, un modèle classique est un système d'équations aux dérivées partielles : les équations dites de Saint-Venant, dont les inconnues sont la hauteur d'eau et la vitesse d'écoulement de l'eau.

Le système de Saint-Venant est un système hyperbolique, introduit à la fin du dix neuvième siècle par Jean-Claude Adhémar Barré comte de Saint-Venant.

Ce dernier fut admis à l'âge de 15 ans à l'Ecole Polytechnique, et entra ensuite à l'Ecole des Ponts et Chaussées dont il sortit premier en 1825. Il présenta en 1834 deux études à l'Académie des Sciences, sur la mécanique théorique et la dynamique des fluides. Après une carrière au Service Technique de la Ville de Paris, il consacra sa retraite à ses recherches scientifiques et entra à l'Académie des Sciences en 1868. Il mourut en janvier 1886. Les équations désormais appelées "de Barré de Saint-Venant", publiées en 1871, sont encore aujourd'hui d'une extrême importance en hydraulique maritime ou fluviale, elles régissent les écoulements à surface libre en eaux peu profondes, d'où leur appellation anglaise "shallow water equations". On les obtient à partir des équations de Navier-Stokes, à l'aide d'hypothèses simplificatrices.

Du fait de leur validité expérimentale et de leur efficacité numérique largement reconnues, les équations de Saint-Venant sont aujourd'hui très utilisées pour la simulation de nombreux phénomènes d'actualité : protection de l'environnement, pollution environnementale, catastrophes naturelles, évolution climatique, rupture de barrage, calcul des marées, étude des crues, sédimentologie...

Pour ce phénomène, toute expérimentation en vraie grandeur reste difficile, voire impossible à réaliser. Pour comprendre ces phénomènes, une maîtrise de la mécanique des fluides à surface libre est donc indispensable. Cette maîtrise a d'abord été assurée par des maquettes et des modèles physiques ; mais ces dernières années, ils ont cédé du terrain face à la résolution numérique de ces équations.

Ces derniers temps, la simulation numérique s'est imposée comme un outil fondamental pour l'hydraulique de l'environnement, favorisée par une progression très importante des capacités des ordinateurs.

Les équations de Saint-Venant posent des problèmes numériques spécifiques, dont voici deux exemples :

- "Sèchage", la hauteur d'eau s'annule, les équations dégénèrent, et certains schémas calculent alors des hauteurs d'eau négatives ;

– "Termes sources", provenant du modèle de topographie, mais aussi de l'alimentation en eau par la pluie et de l'infiltration dans le sol.

Pour résoudre ce système, nous utilisons des schémas équilibre (ou "well-balanced") qui préservent les états d'équilibre. Ces schémas ont été codés en fortran et les codes ont été validés sur de nombreux cas test. Cependant le langage fortran ne permet pas une grande modularité. Ainsi dans le cadre de mon stage effectué à L'INRA d'Orléans, j'ai développé un code c++ pour simuler l'écoulement d'eau.

1 Contexte du stage

1.1 Présentation de l'entreprise

L'INRA, Institut National de la Recherche Agronomique, est un établissement à caractère scientifique et technologique. Créé en 1946, il est sous la tutelle de deux ministères : le ministère chargé de la Recherche et le ministère chargé de l'Agriculture.

Les missions de l'INRA consistent à :

– œuvrer au service de l'intérêt public tout en maintenant l'équilibre entre les exigences de la recherche et les demandes de la société ;
– produire et diffuser des connaissances scientifiques et des innovations, principalement dans les domaines de l'agriculture, de l'alimentation et de l'environnement ;
– contribuer à l'expertise, à la formation, à la promotion de la culture scientifique et technique, au débat science/société.

L'INRA s'organise en :

– 14 départements de recherche touchant l'agriculture, l'alimentation et l'environnement ;
– 21 centres régionaux répartis en près de 200 sites dans toute la France ;
– 470 unités dont 260 unités de recherche, 80 unités expérimentales et 130 unités d'appui et de service.

Le centre INRA d'Orléans a été inauguré en 1977 sur le site d'Ardon. Le personnel se compose d'environ 200 agents titulaires répartis de la manière suivante : 38% de chercheurs et ingénieurs, 49% de techniciens et 13% d'agents administratifs. A cela, s'ajoute environ 50 agents non titulaires : doctorants, stagiaires ...

Situé à proximité du quartier d'Orléans-La Source sur les communes d'Ardon et de St Cyr en Val, le Centre de Recherche d'Orléans s'étend sur 63 ha dont 35 ha de pépinière, et compte près de 8500 m^2 de bureaux et laboratoires (site `http:/www.orleans.inra.fr`).

L'INRA d'Orléans est composé de 5 unités dont l'unité de Recherche de Science du Sol (UR0272) où j'ai effectué mon stage.

L'Unité de Recherche de Science du Sol d'Orléans est constituée d'une seule équipe de recherche composée de 26 membres permanents, dont 7 chercheurs et 6 ingénieurs. Elle accueille en moyenne 5 doctorants ou post-doctorants chaque année et des chercheurs associés. L'Unité a été créée en 2000 à partir du Service d'Etude des Sols et de la Carte Pédologique de France (SESCPF).

Ses recherches sont actuellement organisées en trois axes

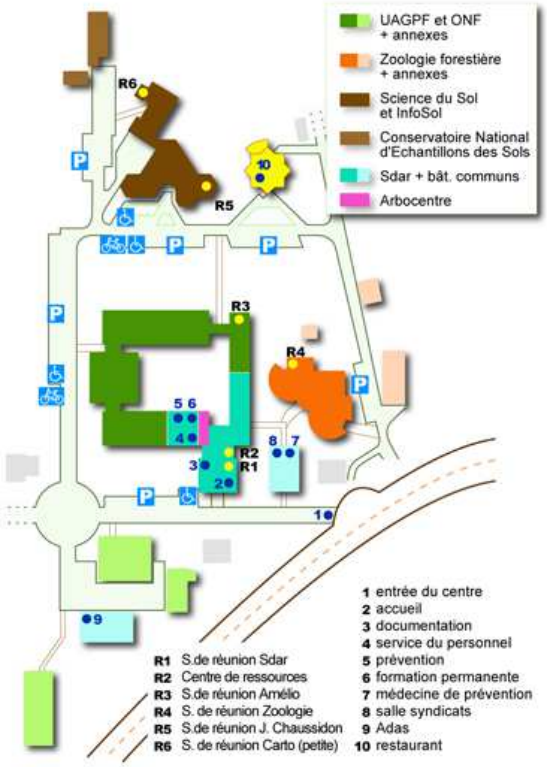


FIG. 1: Plan du site - Dessin © Inra, O. Bertel

FIG. 2: Vue aérienne - Photo : © INRA J.-C. Bastien & O. Bertel

- pédogenèse et évolution des sols en fonction des changements d'usage et de climat ;
- structure, propriétés et fonctionnement actuel des sols ;
- valorisation et protection des sols.

L'Unité de recherche combine des approches d'observations et de mesures sur le terrain avec des sites choisis comme représentatifs de grands systèmes pédologiques, des expérimentations sous conditions contrôlées au laboratoire et de la modélisation informatique. Elle a en charge un simulateur de pluie équipé de bacs de grande taille (10 m^2) unique en France. Elle met en place, sur un territoire en grandes cultures (10 km^2), un site-atelier de suivi spatialisé de l'évolution et du fonctionnement des sols (hydro-thermie et émissions de gaz à effet serre) en fonction des pratiques agricoles. Elle utilise et développe des modèles de fonctionnement spatialisé des sols, à la fois pour formaliser son savoir, tester de nouvelles hypothèses et transférer ses connaissances. Elle a un savoir-faire important en matière de systèmes d'informations géographiques et de géostatistiques.

1.2 Contexte scientifique

Le ruissellement sur les sols cultivés pose des problèmes de conservation des ressources environnementales : diminution des épaisseurs de sol par érosion, pertes en nutriments, ... Pour améliorer l'aménagement des bassins versants, il est nécessaire de prédire correctement les écoulements de surface. Or, les modèles hydrologiques opérationnels sont assez inefficaces sur ce point.

Dans le domaine agricole, des travaux empiriques ont montré que l'interaction sillons-topographie était déterminante sur la géométrie du réseau d'écoulement : pour des faibles écoulements le ruissellement suit la direction des sillons (fig. 3 – cas 2), alors que pour de forts écoulements le ruissellement suit la direction de la plus grande pente (fig. 3 – cas 1). Par manque de connaissance sur cette interaction, ce phénomène n'est pris en compte dans les modèles hydrologiques opérationnels qu'à travers de modèles heuristiques du type loi de tout ou rien.

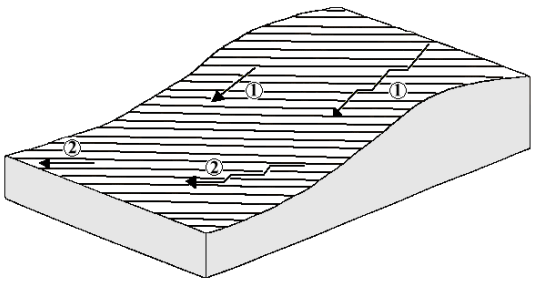


FIG. 3: Exemple de modèle de tout ou rien

Dans le projet ANR METHODE[1](né en 2007), nous souhaitons modéliser le ruissellement afin de comprendre et de prédire l'effet de la morphologie de la surface sur l'écoulement. L'objectif final est de prendre en compte l'effet de l'interaction topographie-sillons dans des modèles couramment utilisés en hydrologie.

[1]Modélisation de l'Ecoulement sur une Topographie avec des Hétérogénéités Orientées et des Différences d'Echelle

Parallèlement à ce projet, je prépare une thèse en mathématiques au MAPMO[2] (UMR CNRS 6628), en collaboration avec l'INRA. L'objectif de cette thèse est de réaliser la simulation de ruissellement d'eau de pluie sur des surfaces agricoles. Le modèle que j'utilise pour ceci est le système de Saint-Venant. Ce système n'admet pas de solution exacte dans le cas général. De nombreuses méthodes numériques ont été utilisées pour résoudre ses équations de manière approchée. Certaines méthodes, telles que le schéma de Mac Cormack, peuvent présenter des instabilités numériques et ne conservent pas toujours la positivité de la hauteur d'eau. Nous utilisons un schéma volumes finis, dit "équilibre" ("well-balanced") avec reconstruction hydrostatique. Outre la préservation de la positivité de la hauteur d'eau, ce schéma permet de calculer correctement les états d'équilibre.

1.3 Objectifs du stage

L'objectif de ce stage consiste à mettre au point deux codes en C++ pour la simulation du ruissellement d'eau de pluie sur des surfaces agricoles. Le premier permet de faire des calculs en une dimension d'espace et le second en deux dimensions d'espace. Les algorithmes permettent différents paramètrages : plusieurs flux numériques, diverses conditions aux limites ... leur programmation nécessite une programmation objet tel que le permet le langage C++.

Dans la première partie de ce rapport, je décrirai le modèle utilisé, la méthode numérique mise en place ainsi que la description des codes existants. Dans la seconde partie, je décrirai la solution choisie, sa mise en oeuvre ainsi que les résultats obtenus.

2 Etat de l'art

2.1 Le modèle de Saint-Venant

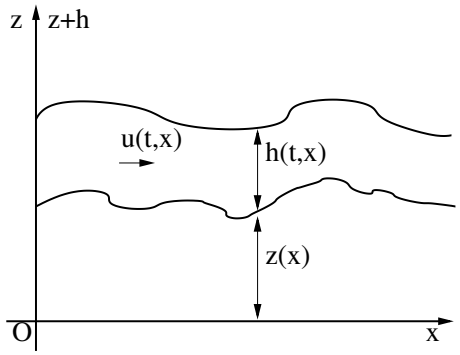


FIG. 4: Schématisation des variables du système de Saint-Venant

Nous nous intéressons à la simulation de ruissellement d'eau de pluie sur des surfaces agricoles. Le modèle que nous considèrons pour cela est le système de Saint-Venant (dérivé dans [Gerbeau00]). C'est un système d'équations aux dérivées partielles qui a déjà été utilisé pour ce type de problème [Esteves00, Fiedler00]. Dans le cadre bidimensionnel, il décrit l'écoulement de l'eau par l'intermédiaire de

[2]Mathématiques et Applications, Physique Mathématique d'Orléans

la hauteur d'eau $h(t,x,y) \geq 0$ et du vecteur vitesse moyen $\vec{u}(t,x,y) = \binom{u}{v} \in \mathbb{R}^2$ (avec x et y les coordonnées spatiales et t le temps).

$$\begin{cases} \partial_t h + \partial_x(hu) + \partial_y(hv) = P \\ \partial_t(hu) + \partial_x(hu^2 + gh^2/2) + \partial_y(huv) = S_{f_x} - gh\partial_x z \\ \partial_t(hv) + \partial_x(huv) + \partial_y(hv^2 + gh^2/2) = S_{f_y} - gh\partial_y z \end{cases}, \tag{1}$$

où g est la constante de gravité, P le terme de pluie, $\vec{S_f}$ un terme de frottement et $z(x,y)$ la topographie.

Il convient de noter que dans ce modèle, la vitesse verticale est négligée. Pour simplifier la description de la méthode numérique, nous nous plaçons dans le cadre monodimensionnel (fig. 4)(le cas bidimensionnel n'en est qu'une généralisation). Le système de Saint-Venant prend alors la forme

$$\begin{cases} \partial_t h + \partial_x(hu) = P \\ \partial_t(hu) + \partial_x(hu^2 + gh^2/2) = S_f - gh\partial_x z \end{cases}, \tag{2}$$

Notons $U = \binom{h}{hu}$ et $F(U) = \binom{hu}{hu^2 + \frac{gh^2}{2}}$ pour le flux.

Ce système peut s'écrire

$$\partial_t U + A(U)\partial_x U = S, \tag{3}$$

avec

$$A(U) = \begin{pmatrix} 0 & 1 \\ -u^2 + gh & 2u \end{pmatrix},$$

donc

$$\det(A(U) - \lambda I) = \lambda^2 - 2u\lambda + u^2 - gh = (\lambda - u)^2 - gh.$$

Les valeurs propres sont

$$\lambda_1(U) = u - \sqrt{gh} \text{ et } \lambda_2(U) = u + \sqrt{gh}.$$

Si $h > 0$, nous avons $\lambda_1(U) < \lambda_2(U)$. Donc en dehors des zones sèches, le système de Saint-Venant est un système strictement hyperbolique.

Soit la vitesse critique $c = \sqrt{gh}$, nous avons alors

$$\lambda_1(U) = u - c \text{ et } \lambda_2(U) = u + c.$$

Ces valeurs propres sont aussi appelées les vitesses caractéristiques, elles permettent de caractériser un écoulement.

Nous avons trois cas possibles :
- si $|u| < c$, l'écoulement est dit fluvial ;
- si $|u| > c$, l'écoulement est dit torrentiel ;
- si $|u| = c$, l'écoulement est dit critique.

A l'aide du système sous forme homogène (c.à.d. $S = 0$ dans (3)), nous obtenons le système de Saint-Venant suivant

$$\begin{cases} \partial_t h + u\partial_x h + h\partial_x u = 0 \\ \partial_t u + g\partial_x h + u\partial_x u = 0 \end{cases}, \tag{4}$$

Comme $h = c^2/g$, nous avons

$$\partial_t(2c) + u\partial_x(2c) + c\partial_x u = 0, \tag{5}$$

et

$$\partial_t u + c\partial_x(2c) + u\partial_x u = 0. \tag{6}$$

La somme de (5) et de (6) donne

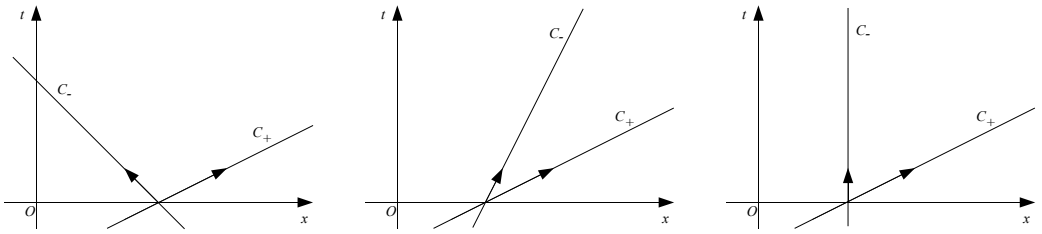
$$\partial_t(u + 2c) + (u + c)\partial_x(2c + u) = 0,$$

donc nous avons $\dfrac{d(u+2c)}{dt} = 0$ ou encore $u + 2c = Cte$ sur la courbe C_+ d'équation $\dfrac{dx}{dt} = u + c$, la différence de (5) et de (6) donne

$$\partial_t(2c - u) + (u - c)\partial_x(2c - u) = 0,$$

donc nous avons $\dfrac{d(u-2c)}{dt} = 0$ ou encore $u - 2c = Cte$ sur la courbe C_- d'équation $\dfrac{dx}{dt} = u - c$.
$u + 2\sqrt{gh}$(respectivement $u - 2\sqrt{gh}$) est donc l'invariant de Riemann correspondant à l'onde $u + \sqrt{gh}$ (respectivement à $u - \sqrt{gh}$).

Nous représentons les caractéristiques dans un repère (O, x, t). On se place dans la situation où l'axe des x est orienté dans le même sens que l'écoulement (donc $u > 0$). Comme nous avons $u + c > 0$, alors quel que soit le type de courant, nous avons $u > 0$, donc $u + c > 0$. La caractéristique C_+ descend donc le courant.



(a) (b) (c)

FIG. 5: Illustration des caractéristiques pour les cas (a) fluvial, (b) torrentiel et (c) critique

Dans le cas d'un écoulement fluvial ($u < c$), nous avons $u - c < 0$. La caractéristique C_- remonte le courant (fig. 5a). Donc une perturbation de la surface libre remonte et descend le courant.
Pour un écoulement torrentiel ($u > c$), nous avons $u - c > 0$. La caractéristique C_- descend le courant (fig. 5b). Donc la perturbation ne peut pas remonter le courant.
Enfin, dans le cas d'un écoulement critique ($u = c$), nous avons $u - c = 0$. La caractéristique C_- est stationnaire (fig. 5c). Donc ici encore, la perturbation ne peut remonter le courant.
Comme nous le verrons plus loin, les caractéristiques et les invariants de Riemann vont être utiles pour prescrire les conditions aux limites si les bords sont liquides.

Pour ce qui va suivre, nous considérons le système sans frottement et sans pluie. Il faut noter que ce système préserve les états d'équilibre au repos

$$h + z = Cte,\ u = 0.$$

C'est une propriété fondamentale pour le problème qui nous intéresse (la flaque d'eau est un exemple d'équilibre au repos).
Depuis les travaux de Greenberg et Leroux [Greenberg96], les schémas numériques satisfaisant cette propriété sont appelés schémas équilibre (ou well-balanced). A cela s'ajoute la difficulté à conserver la quantité d'eau totale et à calculer des hauteurs d'eau positives en particulier aux interfaces sec/mouillé. Le schéma numérique que nous utilisons vérifie toutes ces propriétés, il s'agit du schéma équilibre avec reconstruction hydrostatique décrit dans [Audusse04b], [Bouchut04] et [Marche05].

Dans ce qui va suivre, nous allons voir les différents codes à notre disposition avant le stage.

2.2 Codes disponibles

Depuis le début de ma thèse, je développe un code en fortran 77 permettant la résolution du système de Saint-Venant monodimensionnel. Ce code a été validé sur de nombreux cas tests aussi bien analytiques qu'expérimentaux. Dans le cadre du projet METHODE, deux stages ont eu lieu au BRGM. Les stagiaires (Mohamed El Bouajaji en 2007 [ElBouajaji07] et Marie Rousseau en 2008 [Rousseau08]) ont généralisé mon code monodimensionnel, en réalisant un code de résolution bidimensionnelle en fortran 90. Dans ce dernier, ils ont ajouté la prise en compte de l'infiltration de l'eau. Le code bidimensionnel a aussi été soumis à une batterie de tests. La compilation de ces codes s'est faite à l'aide du compilateur gfortran. Ces codes ont l'avantage d'être assez simples à comprendre. Cependant, ils ont de gros problèmes de modularité. En effet, pour la résolution du système, nous pouvons jouer sur différents paramètres : l'ordre du schéma, le flux numérique... Pour ajouter un composant, nous sommes obligés de le faire dans le corps du programme. A cela s'ajoute des tests redondants (le flux numérique à utiliser...). L'implantation en c++ devra permettre de s'affranchir de ces limites.

Dans la partie suivante, nous allons décrire et analyser la méthode numérique utilisée afin de voir les contraintes que cela pose au niveau informatique.

3 Méthode numérique

3.1 Schéma avec reconstruction hydrostatique

Aspects mathématiques

Considérons le système de Saint-Venant sous la forme suivante

$$\begin{cases} \partial_t h + \partial_x(hu) = 0 \\ \partial_t(hu) + \partial_x(hu^2 + gh^2/2) = -gh\partial_x z \end{cases}, \tag{7}$$

nous approchons ses solutions

$$U(t,x) \in \mathbb{R}^+ \times \mathbb{R} \text{ avec } x \in \mathbb{R} \text{ et } t > 0,$$

par des valeurs discrètes

$$U_i^n = \begin{pmatrix} h_i^n \\ h_i^n u_i^n \end{pmatrix} \text{ avec } i \in \mathbb{Z} \text{ et } n \in \mathbb{N}.$$

Nous considèrons un pas de temps $\Delta t > 0$, constant et définissons les temps discrets par $t_n = n.\Delta t$ avec $n \in \mathbb{N}$. Nous utilisons la méthode des volumes finis qui est basée sur la discrétisation intégrale des équations. Cela nécessite la subdivision de l'espace en cellules ou volumes finis (fig. 6)

$$C_i =]x_{i-1/2}, x_{i+1/2}[,$$

centrés sur

$$x_i = \frac{x_{i-1/2} + x_{i+1/2}}{2},$$

et de longueur

$$\Delta x_i = x_{i+1/2} - x_{i-1/2} > 0,$$

puis nous intègrons le système considéré sur chaque cellule et sur un pas de temps $[t^n, t^{n+1}[\times]x_{i-1/2}, x_{i+1/2}[$. Apparaissent alors les moyennes des solutions sur chaque cellule

$$U_i^n \simeq \frac{1}{\Delta x_i} \int_{C_i} U(t_n, x) dx.$$

Nous obtenons des solutions constantes par morceaux (et donc discontinues) et des

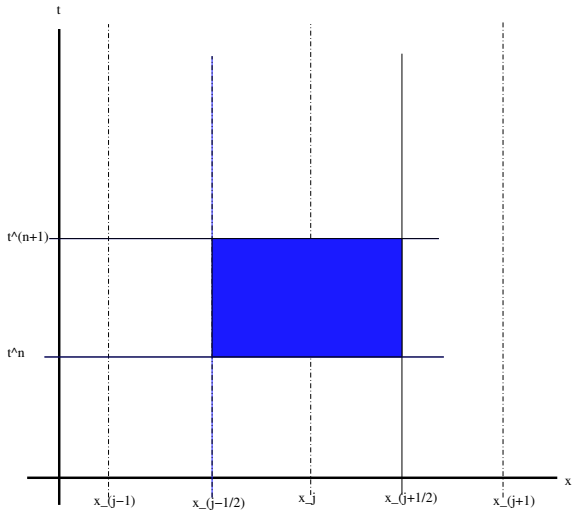


FIG. 6: Discrétisation du domaine en volumes finis

termes de bords (autrement dit, les flux échangés entre les cellules au niveau de leur frontière ou interface). Ainsi, le flux sortant d'une cellule est égal à celui qui rentre dans la cellule voisine (fig. 7), d'où un algorithme conservatif.

A l'ordre 1 (plus l'ordre est élevé, plus le schéma est précis), le schéma aux volumes finis s'écrit

$$U_i^{n+1} - U_i^n + \frac{\Delta t}{\Delta x_i}(F_{i+1/2G}^n - F_{i-1/2D}^n) = 0$$

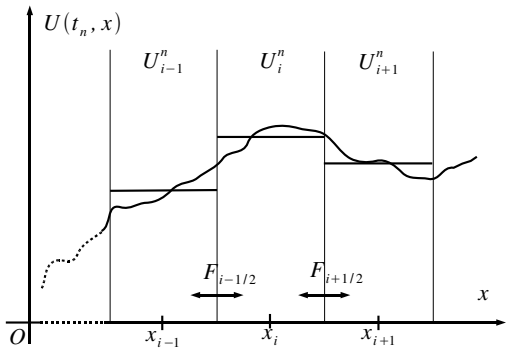


FIG. 7: Illustration des échanges de flux entre les cellules

avec

$$\begin{aligned} F^n_{i-1/2D} &= \mathcal{F}_D(U^n_{i-1}, U^n_i, \Delta z_{i-1/2}) \\ &= \mathcal{F}(U^n_{i-1/2G}, U^n_{i-1/2D}) + S_{i-1/2D} \end{aligned}$$

$$\begin{aligned} F^n_{i+1/2G} &= \mathcal{F}_G(U^n_i, U^n_{i+1}, \Delta z_{i+1/2}) \\ &= \mathcal{F}(U^n_{i+1/2G}, U^n_{i+1/2D}) + S_{i+1/2G} \end{aligned}$$

avec

$$\Delta z_{i+1/2} = z_{i+1} - z_i,$$

$\mathcal{F}$ est un flux numérique à préciser et nous avons la discrétisation du terme source

$$\begin{aligned} S_{i+1/2G} - S_{i-1/2D} &= \begin{pmatrix} 0 \\ g\frac{(h^n_i)^2 - (h^n_{i+1/2G})^2}{2} \end{pmatrix} - \begin{pmatrix} 0 \\ g\frac{(h^n_i)^2 - (h^n_{i-1/2D})^2}{2} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ g\frac{(h^n_{i-1/2D})^2 - (h^n_{i+1/2G})^2}{2} \end{pmatrix} \end{aligned}$$

Les variables $U_{\cdot G}$ et $U_{\cdot D}$ sont obtenues par reconstruction hydrostatique à partir des variables $U_\cdot$.

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

Nous pouvons remarquer qu'avec une telle reconstruction, les variables vérifient l'équation d'équilibre. La partie positive $(\bullet)_+ = \max(0, \bullet)$ assure l'obtention de hauteurs d'eau positives. Afin d'améliorer la précision du schéma, il convient de réaliser une montée à l'ordre 2.

Implications pour la programmation

Cette partie de la méthode numérique est facilement transposable en code informatique. Pour la discrétisation en espace et en temps nous avons recours à des variables entières pour les boucles sur les indices i et n et des variables réelles pour les pas d'espace Δx et de temps Δt. On voit que les calculs nécessitent en entrée :

– la topographie z_i ;
– la hauteur d'eau h_i^0 au temps $n = 0$;
– la vitesse u_i^0 au temps $n = 0$

pour $i = 1$ à J, où J est le nombre total de mailles en espaces.

Via le schéma (reconstruction hydrostatique, flux ...), on obtient après M itérations les données de sorties au temps M :
– la hauteur d'eau h_i^M au temps $n = M$;
– la vitesse u_i^M au temps $n = M$.

Pour cela nous prenons des tableaux de réels permettant de stocker les valeurs nécessaires aux calculs et de les écraser par les suivantes (en prenant le soin de les sauvegarder si nécessaires).
La reconstruction hydrostatique (tab. 1) ne présente pas de difficultés majeures. Les entrées sont des variables sur lesquelles nous effectuons des calculs pour obtenir en sorties des variables reconstruites (hg_rec et hd_rec).

hydrostatique
$-\ hg\_rec$ $-\ hd\_rec$
$+\ calcul(double, double, double, double) : void$ $+\ get\_hg() : double$ $+\ get\_hd() : double$
$hydrostatique()$

TAB. 1: Le diagramme de la classe hydrostatique

.

3.2 Schéma à l'ordre 2

Aspects mathématiques

Pour obtenir un schéma d'ordre 2, nous combinons :
– le schéma précédent à l'ordre 2 en espace ;
– la méthode de Heun pour l'ordre 2 en temps.

Les variables $(U_i, z_i)_{i\in\mathbb{Z}}$ sont remplacées par des variables $(U_{i+1/2\pm, z_i+1/2\pm})_{i\in\mathbb{Z}}$ obtenues par reconstruction linéaire. Nous effectuons la reconstruction hydrostatique sur ces dernières.

Afin d'assurer la consistance du schéma, il est nécessaire d'ajouter un terme source centré Fc_i. Le schéma d'ordre 2 est alors de la forme

$$U_i^{n+1} - U_i^n + \frac{\Delta t}{\Delta x_i}(F_{i+1/2G}^n - F_{i-1/2D}^n - Fc_i^n) = 0$$

avec

$$F_{i-1/2D}^n = \mathcal{F}_D(U_{i-1/2-}^n, U_{i-1/2+}^n, \Delta z_{i-1/2}^n)$$
$$F_{i+1/2G}^n = \mathcal{F}_G(U_{i+1/2-}^n, U_{i+1/2+}^n, \Delta z_{i+1/2}^n)$$
$$Fc_i^n = \mathcal{F}c(U_{i-1/2+}^n, U_{i+1/2-}^n, \Delta z_i^n)$$

où

$$\Delta z^n_{i+1/2} = z^n_{i+1/2+} - z^n_{i+1/2-} \text{ et } \Delta z^n_i = z^n_{i+1/2-} - z^n_{i-1/2+} \tag{8}$$

$\mathcal{F}_D$ et $\mathcal{F}_G$ sont les flux définis à l'ordre 1 et

$$Fc^n_i = \begin{pmatrix} 0 \\ -g\frac{h^n_{i-1/2+}+h^n_{i+1/2-}}{2}\Delta z^n_i \end{pmatrix}$$

Les variables $U_{\cdot G}$ et $U_{\cdot D}$ sont obtenues par reconstruction hydrostatique à partir des variables $U_{\cdot -}$ et $U_{\cdot +}$

$$\begin{cases} U_{i+1/2G} = (h_{i+1/2G}, h_{i+1/2G}u_{i+1/2-}),\ U_{i-1/2D} = (h_{i-1/2D}, h_{i-1/2D}u_{i-1/2+}) \\ h_{i+1/2G} = (h_{i+1/2-} + z_{i+1/2-} - \max(z_{i+1/2-}, z_{i+1/2+}))_+ \\ h_{i-1/2D} = (h_{i-1/2+} + z_{i-1/2+} - \max(z_{i-1/2-}, z_{i-1/2+}))_+ \end{cases},$$

Pour obtenir les variables $(U_{\cdot\pm}, z_{\cdot\pm})$, nous effectuons une méthode de reconstruction linéaire sur les variables u, h, $z+h$ et en déduisons la reconstruction de z. Comme indiqué dans [Audusse04b] et [Audusse04c] la reconstruction est faite sur h et $z+h$ car c'est le seul moyen de préserver les états d'équilibre et de garantir la positivité de h aux interfaces sec/mouillé (voir [Audusse04b] et [Audusse04c] pour plus de détails). Il faut remarquer qu'en procèdant ainsi les variables $z_{\cdot\pm}$ dépendent du temps d'où la présence des exposants en n (8).

Pour reconstruire les variables, deux méthodes nous intéressent : la reconstruction MUSCL (Monotonic Upwind Scheme for Conservation Law) et la reconstruction ENO (Essentially Non Oscillatory).
Pour une fonction scalaire $U \in \mathbb{R}$, on définit la reconstruction MUSCL par

$$U_{i-1/2+} = U_i - \frac{\Delta x_i}{2}.D_{mm}U_i \text{ et } U_{i+1/2-} = U_i + \frac{\Delta x_i}{2}.D_{mm}U_i,$$

avec

$$D_{mm}U_i = \text{minmod}(2.\frac{(U_i - U_{i-1})}{\Delta x_{i-1} + \Delta x_i}, 2.\frac{(U_{i+1} - U_i)}{\Delta x_i + \Delta x_{i+1}})$$

et

$$\text{minmod}(x,y) = \begin{cases} \min(x,y) \text{ si } x,y \geq 0 \\ \max(x,y) \text{ si } x,y \leq 0 \\ 0 \text{ sinon} \end{cases}.$$

Cette reconstruction présente l'avantage de vérifier le principe du maximum (nécessaire à la positivité de h).

La reconstruction ENO est définie par

$$U_{i-1/2+} = U_i - \frac{\Delta x}{2}.D_{eno}U_i \text{ et } U_{i+1/2-} = U_i + \frac{\Delta x}{2}.D_{eno}U_i,$$

avec

$$D_{eno}U_i = \text{minmod}(\frac{U_i - U_{i-1}}{\Delta x} + \frac{\Delta x}{2}.D^2U_{i-1/2}, \frac{U_{i+1} - U_i}{\Delta x} - \frac{\Delta x}{2}.D^2U_{i+1/2}),$$

où

$$D^2U_{i+1/2} = \text{minmod}(\frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}, \frac{U_{i+2} - 2U_{i+1} + U_i}{\Delta x^2}).$$

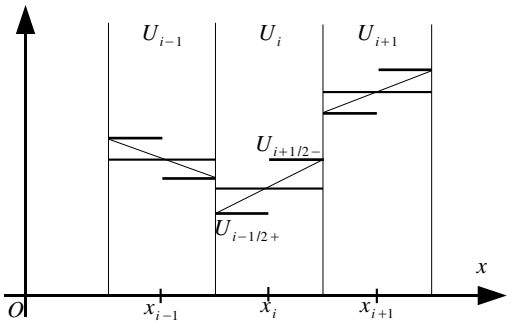


FIG. 8: Reconstruction MUSCL

Cette reconstruction permet de gagner en précision (la dérivée discrète DU_i est d'ordre 2), mais on risque la perte du principe du maximum. Pour remédier à ceci, nous utilisons une méthode préconisée dans [Bouchut04], en effectuant une reconstruction ENO sur la variable u et une reconstruction ENO modifiée sur h et $z+h$. La reconstruction ENO modifiée consiste à prendre

$$D_{enom}U_i = \text{minmod}(D_{eno}U_i, 2.D_{mm}U_i),$$

à la place de

$$D_{eno}U_i.$$

En faisant ainsi, nous obtenons un schéma d'ordre 2 en espace qui peut s'écrire sous la forme

$$U^{n+1} = U^n + \Delta t.\Phi(U^n),$$

où

$$U = (U_i)_{i\in\mathbb{Z}}$$

et

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

Jusqu'à présent notre choix s'était porté sur le limiteur de pente minmod, mais d'autres choix sont possibles, tel que le limiteur de Van Albada [Audusse05]

$$\text{Van Albada}(x,y) = \begin{cases} 0 \text{ si } \text{sign}(x) \neq \text{sign}(y) \\ \dfrac{x(y^2+\epsilon) + y(x^2+\epsilon)}{x^2+y^2+2\epsilon} \text{ sinon} \end{cases},$$

avec $0 \leq \epsilon << 1$.

Pour obtenir un schéma d'ordre 2 en temps, nous utilisons la méthode de Heun qui est une méthode de prédiction-correction. Nous obtenons le schéma d'ordre 2 en temps et en espace suivant :

$$\tilde{U}^{n+1} = U^n + \Delta t.\Phi(U^n),$$
$$\tilde{U}^{n+2} = \tilde{U}^{n+1} + \Delta t.\Phi(\tilde{U}^{n+1}),$$
$$U^{n+1} = \frac{U^n + \tilde{U}^{n+2}}{2}.$$

Maintenant que le schéma est présenté à l'ordre 1 et à l'ordre 2, il convient de choisir un flux numérique.

Implications pour la programmation

Comme les entrées et sorties sont les mêmes qu'à l'ordre 1, nous reprenons les mêmes variables. La méthode de Heun étant une méthode de prédiction-correction, elle nécessite le stockage de valeurs supplémentaires. Pour cela, nous ajoutons des tableaux supplémentaires qui reçoivent les valeurs calculées à chaque itération en temps.

La différence majeure par rapport à l'ordre 1 est l'utilisation de variables reconstruites dans le calcul des flux numériques. Nous avons trois façons différentes d'effectuer cette reconstruction :

- reconstruction MUSCL ;
- reconstruction ENO ;
- reconstruction ENO modifiée.

Ces méthodes sont différentes, cependant elles ont en commun les entrées/sorties et l'utilisation du limiteur de pente. Du fait de la quantité de variables, cette méthode étant complexe, je n'en fournis pas ici de représentation. Au contraire, la structure commune des limiteurs de pente (tab. 2) est assez simple à représenterdégager : nous avons deux variables en entrée et une seule en sortie.

limiteur
$-\ rec$
$+\ calcul(double, double) : void$ $+\ get\_rec() : double$
$limiteur()$

TAB. 2: Le diagramme de la classe limiteur

.

3.3 Flux numériques

Aspects mathématiques

Ce schéma a déjà été utilisé avec plusieurs flux numériques : Suliciu [Bouchut04], cinétique [Audusse04b], VFRoe et sa variante VFRoencv [Bouchut04, Marche05]. Jusqu'à présent notre choix s'est porté sur le flux de Rusanov [Bouchut04] et sur le flux de Harten, Lax et van Leer [Bouchut04], noté HLL. Ces flux sont des généralisations du flux de Lax-Friedrichs.

- Le flux de Rusanov est de la forme

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

avec

$$c = \max(|u_G| + \sqrt{gh_G}, |u_D| + \sqrt{gh_D});$$

- Le flux HLL s'écrit quant à lui

$$\mathcal{F}(U_G, U_D) = \begin{cases} F(U_G) \text{ si } 0 < c_1 \\ \dfrac{c_1 F(U_G) - c_2 F(U_D)}{c_2 - c_1} + \dfrac{c_1 c_2}{c_2 - c_1}.(U_D - U_G) \text{ si } c_1 < 0 < c_2 \\ F(U_D) \text{ si } c_2 < 0 \end{cases},$$

avec deux paramètres

$$c_1 < c_2.$$

Pour c_1 et c_2, nous prenons

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

où $\lambda_1(U) = u - \sqrt{gh}$ et $\lambda_2(U) = u + \sqrt{gh}$ sont les valeurs propres du système de Saint-Venant.
Le flux HLL s'écrit aussi sous la forme :

$$\mathcal{F}(U_G, U_D) = t_1 F(U_D) + t_2 F(U_G) - t_3(U_D - U_G),$$

avec

$$t_1 = \frac{\min(c_2, 0) - \min(c_1, 0)}{c_2 - c_1},\ t_2 = 1 - t_1,\ t_3 = \frac{c_2|c_1| - c_1|c_2|}{2(c_2 - c_1)}.$$

Il est nécessaire d'imposer une condition de CFL (Courant Friedrichs Levy) sur le pas de temps pour prévenir une explosion des valeurs numériques, cette condition est de la forme

$$\Delta t \leq \frac{\min_i(\Delta x_i)}{\max_i(|u_i| + \sqrt{gh_i})} \text{ à l'ordre 1}$$

et

$$\Delta t \leq \frac{\min_i(\Delta x_i)}{2\max_i(|u_i| + \sqrt{gh_i})} \text{ à l'ordre 2,}$$

où i est l'ensemble des valeurs possibles du domaine de calcul.

Implications pour la programmation

Comme pour les reconstruction à l'ordre 2, il faut remarquer que tous les flux numériques (tab. 3) ont les mêmes entrées/sorties. En sortie nous obtenons les deux composantes du flux numérique ($f1$ et $f2$) et la CFL. Plutôt que de mettre une méthode qui récupère $tx = dt/dx$, nous pourrions le faire dans le construction. Mais la méthode $set\_tx()$ permettra de réaliser un programme avec pas de temps adaptable. Ce qui diffère entre les différents flux, c'est leur méthode *calcul*().

3.4 Conditions aux limites

Aspects mathématiques

Ce schéma est explicite : afin de calculer la solution numérique à l'instant t_{n+1}, le schéma numérique a recours aux valeurs trouvées à l'instant t_n, au noeud courant ainsi qu'aux noeuds voisins (fig. 9). Il est donc impossible d'appliquer le schéma numérique au premier et au dernier noeuds du domaine spatial. Il est nécessaire de faire un traitement spécial pour calculer les valeurs de l'écoulement en ces noeuds. Ce traitement s'appelle les conditions aux limites.

Les méthodes les plus utilisées pour traiter les conditions aux limites sont les méthodes d'extrapolation constante et linéaire dans l'espace.

$flux$
$-\ f1$ $-\ f2$ $-\ tx$
$+\ calcul(double, double, double, double) : void$ $+\ set\_tx(double) : void$ $+\ get\_f1() : double$ $+\ get\_f2() : double$ $+\ get\_cfl() : double$
$flux()$

TAB. 3: Le diagramme de la classe flux

.

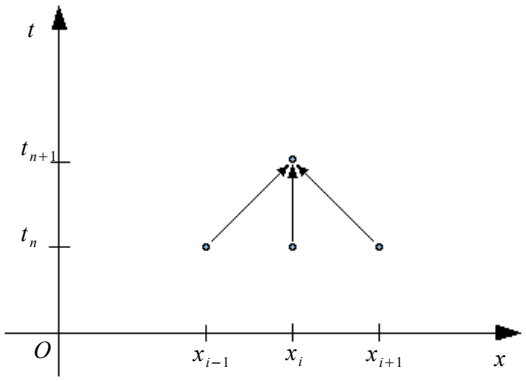


FIG. 9: Schéma explicite centré

Soient $U_1^{n+1}, U_2^{n+1}, U_{J-2}^{n+1}$ et U_{J-1}^{n+1}, les valeurs calculées au temps t_{n+1} à l'aide du schéma numérique, les valeurs U_0^{n+1} et U_J^{n+1} sont calculées à l'aide d'une extrapolation constante par

$$\left\{ \begin{array}{l} U_0^{n+1} = U_1^{n+1} \\ U_J^{n+1} = U_{J-1}^{n+1} \end{array} \right. , \tag{9}$$

ou à l'aide d'une extrapolation linéaire par

$$\left\{ \begin{array}{l} U_0^{n+1} = 2.U_1^{n+1} - U_2^{n+1} \\ U_J^{n+1} = 2.U_{J-1}^{n+1} - U_{J-2}^{n+1} \end{array} \right. . \tag{10}$$

Le traitement des conditions aux limites par ces méthodes présente l'inconvénient de ne pas tenir compte des caractéristiques de l'écoulement, ce qui peut engendrer des problèmes d'instabilité ou d'ondes réfléchies. Dans ce qui va suivre, nous allons voir comment nous pouvons tenir compte de ces caractéristiques (inspiré de [Bristeau01]).

Considèrons que nous résolvons le système de Saint-Venant pour $x \in [0, L]$. Nous définissons $U_{bord} = U(0)$ ou $U(L)$ et $U_{\text{à coté}} = U(\Delta x)$ ou $U(L - \Delta x)$, suivant le bord considéré ($x = 0$ ou $x = L$).

- condition aux limites pour un écoulement fluvial
  Nous considèrons que l'écoulement est fluvial au bord, nous avons donc
  $$|u_{bord}| < \sqrt{g h_{bord}}$$
  , c'est à dire
  $$(u_{bord} - \sqrt{g h_{bord}})(u_{bord} + \sqrt{g h_{bord}}) < 0. \tag{11}$$
  Nous supposons que la hauteur d'eau est donnée
  $$h_{bord} = h_0.$$
  Nous nous plaçons en $x = 0$. L'invariant de Riemann est constant le long de la caractéristique sortante $u - \sqrt{gh}$.
  Nous avons alors
  $$\left\{ \begin{array}{l} h_{bord} = h_0 \\ u_{bord} = u_{\text{à coté}} - 2\sqrt{g}(\sqrt{h_{\text{à coté}}} - \sqrt{h_{bord}}) \end{array} \right. . \tag{12}$$
  En $x = L$, avec un raisonnement analogue, nous avons
  $$\left\{ \begin{array}{l} h_{bord} = h_0 \\ u_{bord} = u_{\text{à coté}} + 2\sqrt{g}(\sqrt{h_{\text{à coté}}} - \sqrt{h_{bord}}) \end{array} \right. . \tag{13}$$
  Attention, si le couple (h_{bord}, u_{bord}) ne vérifie pas (11), alors l'écoulement est en fait torrentiel, il faut alors se reporter aux paragraphes suivants. Dans cette situation, nous devons distinguer deux cas : la sortie et l'entrée.
- condition aux limites pour une sortie torrentielle
  Nous avons $|u| > \sqrt{gh}$ donc les deux invariants de Riemann sont constants le long des caractéristiques sortantes.
  Nous avons donc
  $$\left\{ \begin{array}{l} u_{bord} - 2\sqrt{g h_{bord}} = u_{\text{à coté}} - 2\sqrt{g h_{\text{à coté}}} \\ u_{bord} + 2\sqrt{g h_{bord}} = u_{\text{à coté}} + 2\sqrt{g h_{\text{à coté}}} \end{array} \right. , \tag{14}$$
  nous en déduisons que
  $$\left\{ \begin{array}{l} h_{bord} = h_{\text{à coté}} \\ u_{bord} = u_{\text{à coté}} \end{array} \right. . \tag{15}$$

– condition aux limites pour une entrée torrentielle

Les caractéristiques sont entrantes, nous devons donc imposer h_{bord} et u_{bord}.

Afin de rendre notre méthode numérique plus complète, nous devons tenir compte de la pluie et des frottements.

Implications pour la programmation

Concernant les conditions de bords :

– les sorties sont communes (calcul de $h_0, u_0, h_{J+1}, u_{J+1}$) ;

– les entrées ne sont pas nécessairement les mêmes (pour une condition périodique en $i = 0$ nous avons besoin des valeurs en $i = J$, pour une extrapolation constante des valeurs en $i = 1...$).

D'un point de vue pratique, nous choisissons de mettre plus d'entrées que nécessaires, ainsi les implantations des conditions de bord ont les mêmes entrées-sorties. Il faut aussi remarquer que la méthode des caractéristiques en $i = 0$ est différente de celle en $i = J + 1$, il faudra donc veiller à créer une classe pour les conditions de bord à gauche et pour celles à droite.

3.5 Pluie et frottement

Aspects mathématiques

Le terme de pluie est traité par un splitting qui consiste à résoudre l'équation $\partial_t h = P$. Sur chaque pas de temps nous découpons le calcul en deux étapes

– ajout du volume d'eau ;

– calcul des écoulements.

Concernant les frottements, nous utilisons deux lois qui sont

– Manning $S_f = \dfrac{u|u|}{K^2 h^{4/3}}$,

– Darcy-Weisbach $S_f = \dfrac{fu|u|}{gh}$.

où K et f sont les coefficients. Ces frottements sont traités de façon semi-implicite [Bristeau01].

Pour les frottements de Darcy-Weisbach, nous écrivons

$$q_i^{n+1} + \frac{f|q_i^n|q_i^{n+1}}{8h_i^n h_i^{n+1}}\Delta t = q_i^n + \frac{\Delta t}{\Delta x_i}(\mathcal{F}_{i+1/2G} - \mathcal{F}_{i-1/2D}).$$

En notant q_{j*}^{n+1} la partie de droite, nous avons

$$q_i^{n+1} = \frac{q_{i*}^{n+1}}{1 + \Delta t \dfrac{f|u_i^n|}{8h_i^{n+1}}}.$$

pour les frottements de Manning, nous avons

$$q_i^{n+1} = \frac{q_{i*}^{n+1}}{1 + \Delta t \dfrac{g|u_i^n|}{K^2 (h_i^{n+1})^{4/3}}}.$$

Implications pour la programmation

L'implantation de la pluie ne présente pas de difficultés majeures. Concernant les frottements (tab. 4), les entrées et sorties sont les mêmes.

$frottement$
$- c$ $- qmod$ $- dt$
$+ calcul(double, double, double) : void$ $+ get\_qmod() : double$ $+ set\_c(double) : void$ $+ set\_dt(double) : void$
$frottement()$

TAB. 4: Le diagramme de la classe frottement

.

3.6 Algorithme général

Pour le schéma d'ordre 2, nous pouvons dégager l'algorithme ci-après. On discrétise le temps et l'espace de manière régulière avec dt le pas de temps et dx le pas d'espace.

Algorithme 1 Calcul "well-balanced" à l'ordre 2

Entrées : h^0, u^0 et z
Sorties : h^M et u^M

1: Initialisation de h^0, u^0 et z
2: **pour** $t \leftarrow 1, M$ **faire** ▷ Boucle en temps
3: Ajout de la pluie sur la première moitié du pas de temps
4: Reconstruction ordre 2 en x sur h^{n-1}, u^{n-1} et z (MUSCL, ENO ou ENO modifié)
▷ Nouvelles variables : $h_{i-1/2+}$, $h_{i+1/2-}$, $u_{i-1/2+}$, $u_{i+1/2-}$, $z_{i-1/2+}$, $z_{i+1/2-}$
5: Reconstruction hydrostatique aux interfaces droite et gauche
6: Calcul des flux aux interfaces (Rusanov ou HLL)
7: Application du schéma
8: Traitement des frottements en semi-implicite
▷ Nouvelles variables : hs, us
9: Reconstruction ordre 2 en x sur hs, us et z (MUSCL, ENO ou ENO modifiée)
▷ Nouvelles variables : $hs_{i-1/2+}$, $hs_{i+1/2-}$, $us_{i-1/2+}$, $us_{i+1/2-}$, $zs_{i-1/2+}$, $zs_{i+1/2-}$
10: Reconstruction hydrostatique aux interfaces droite et gauche
11: Calcul des flux aux interfaces (Rusanov ou HLL)
12: Application du schéma
13: Traitement des frottements en semi-implicite
▷ Nouvelles variables : hsa, usa
14: Méthode de Heun (montée en ordre 2 en temps)
15: Ajout de la pluie sur la deuxième moitié du pas de temps
▷ Nouvelles variables : h^n, u^n
16: **fin pour**

4 Travail réalisé

4.1 Réalisations

Dans la section 3, nous avons analysé la méthode numérique en vue de son implantation. Une grande conclusion en est sortie : pour réaliser nos calculs nous avons un nombre de catégories important (ordre du schéma, flux, condition de bord ...). La problèmatique est l'implantation de ces composants. En effet, dans nos codes en fortran, les programmes font des tests sur chaque pas de temps et d'espace pour déterminer le flux numérique, les frottements, etc, à utiliser. Nous pouvons aisément imaginer que ceci est coûteux (et pas très élégant).

Dans l'analyse, nous avons vu que dans chaque catégorie, ces composants avaient les mêmes entrées/sorties. Des structures de classes ont été mises en valeur. J'ai alors naturellement décidé d'utiliser les notions de classe mère virtuelle et d'héritage, pour définir les éléments communs à tout ces composants.

En prenant pour exemple les termes de frottement (fig. 10), nous définissons une classe mère virtuelle $frottement$. Puis, des classes filles $Fr\_Manning$ et $Fr\_Darcy\_Weisbach$. Pour éviter les phénomènes de test sur chaque pas, j'ai créé une classe $choix\_frottement$ qui joue le rôle de distributeur de frottement. Dans l'entête de la classe $choix\_frottement$, nous déclarons un pointeur sur un $frottement$ (qui est une classe abstraite). A l'appel du constructeur $choix\_frottement(intchoix)$ de la classe $choix\_frottement$, ce pointeur subit un transtypage. Il pointe alors sur un objet de classe $Fr\_Manning$ ou $Fr\_Darcy\_Weisbach$ (en fonction de la valeur de la variable $choix$).

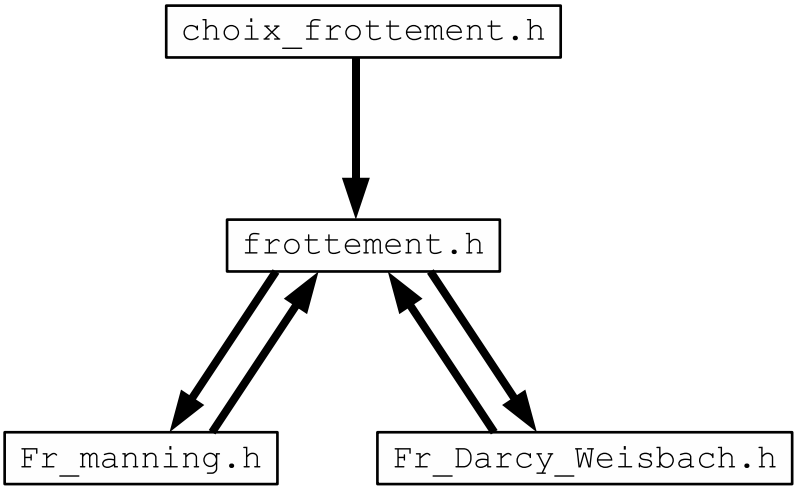


FIG. 10: Diagramme des classes pour les frottements

J'ai reproduit cette méthode sur les autre composants du programme ($choix\_flux$, $choix\_limiteur$...) pour le code monodimensionnel et le code bidimensionnel. Au moment de la rédaction de ce mémoire, le code bidimensionnel en était encore au stade du debogage. Comme il y a plus d'accès que d'écriture dans les cellules des tableaux, j'ai opté pour la classe *vector*. Le conteneur *vector* permet un accès direct à un élément quelconque avec une efficacité en $O(1)$.

J'ai cherché à éliminer les calculs inutiles et essayé de stocker les résultats d'opérations effectuées plusieurs fois ($\frac{g}{2}$, $h \times u$...).

4.2 Les moyens à disposition

D'un point de vue matériel et logiciel, j'ai utilisé mon ordinateur portable avec le compilateur g++ et Valgrind pour chercher les fuites de mémoire. J'ai commencé la réalisation de la documentation de mes codes à l'aide du logiciel Doxygen (fig. 11).

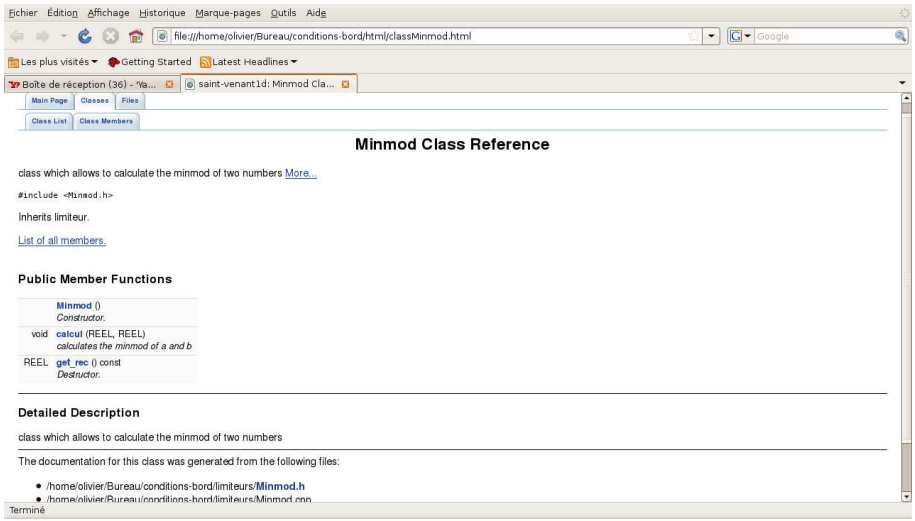


FIG. 11: Documentation de la classe minmod à l'aide de Doxygen

D'un point de vue humain, j'ai eu plusieurs interlocuteurs :
- pour la structuration du code : mon encadrant Frédéric Darboux (INRA), Marie Rousseau (stage BRGM) ;
- pour des questions plus spécifiques en c++ : Thomas Haberkorn (MAPMO) et Ulrich Razafizon (Jussieu).

4.3 Résultats

Dans cette partie, nous allons comparer le code monodimensionnel en c++ avec le code monodimensionnel en fortran. Cette comparaison est faite sur deux cas tests très classiques : des ruptures de barrage idéales (c.à.d. sans frottement). Ces cas test admettent des solutions exactes (pour plus de détails voir [Hervouet03] et l'annexe). Dans les deux cas que nous allons regarder, je me suis placé sur un domaine de longueur $L = 10m$ avec le barrage en $x = 5m$. J'ai utilisé les deux codes (c++ et fortran) à l'ordre 2, avec la reconstruction ENO modifiée et le flux HLL (le flux de Rusanov donne de moins bons résultats). Dans les deux situations, j'ai observé la hauteur d'eau $0,25s$ après la rupture du barrage.

4.3.1 Rupture de barrage sur fond mouillé

Le premier test est une rupture de barrage sur fond mouillé (fig. 12). Habituellement, ce test est utilisé pour comparer deux flux numériques. Mais ici, je l'ai utilisé pour tester leur programmation.

Initialement, nous avons

$$h(x, t = 0) = \begin{cases} 5 \text{ pour } x \leq 5 \\ 2 \text{ sinon} \end{cases},$$

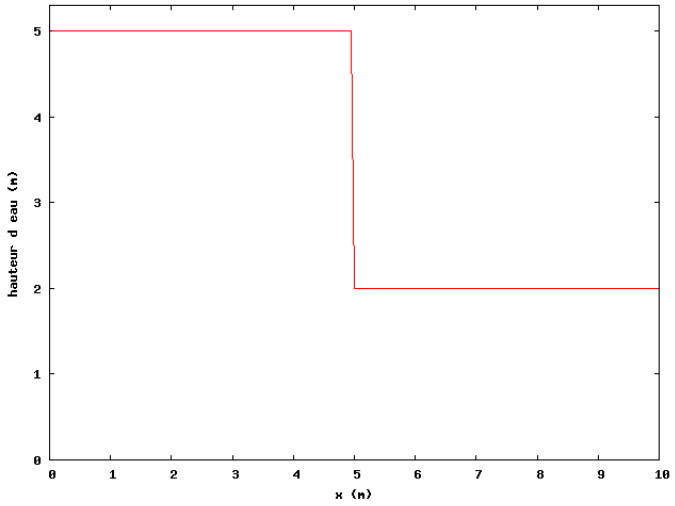


FIG. 12: Etat avant la rupture

Le domaine a été découpé en $J = 200$ mailles en espace, les calculs ont été effectués sur $M = 3200$ pas de temps.

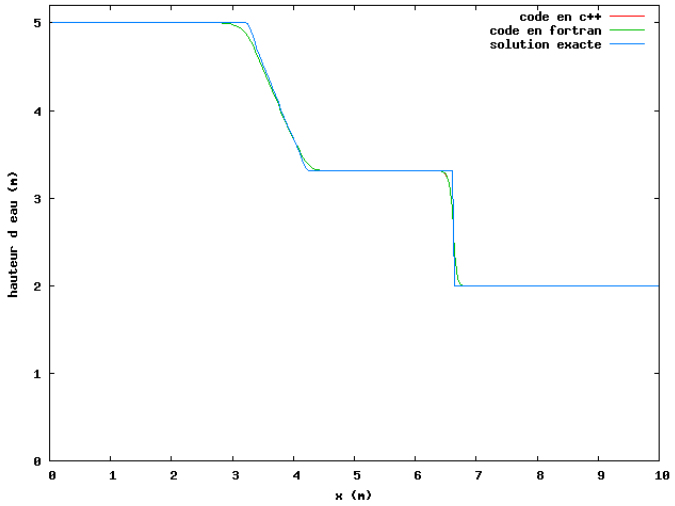


FIG. 13: Hauteur d'eau à $t = 0,25s$

Pour ce cas test, les deux codes donnent les mêmes résultats (fig. 13). De plus, nous pouvons observer que les résultats sont très proches de la solution exacte.

4.3.2 Rupture de barrage sur fond sec

Le deuxième test est une rupture de barrage sur fond sec (fig. 14), nous avons

$$h(x, t = 0) = \begin{cases} 5 \text{ pour } x \leq 5 \\ 0 \text{ sinon} \end{cases} .$$

Le domaine a été découpé en $J = 50$ mailles en espace, les calculs ont été effectués sur $M = 800$ pas de temps. Ce test permet de tester la capacité du programme à

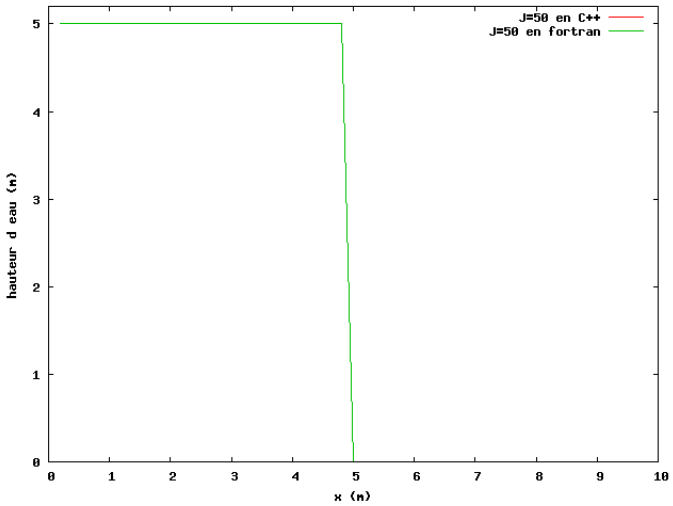
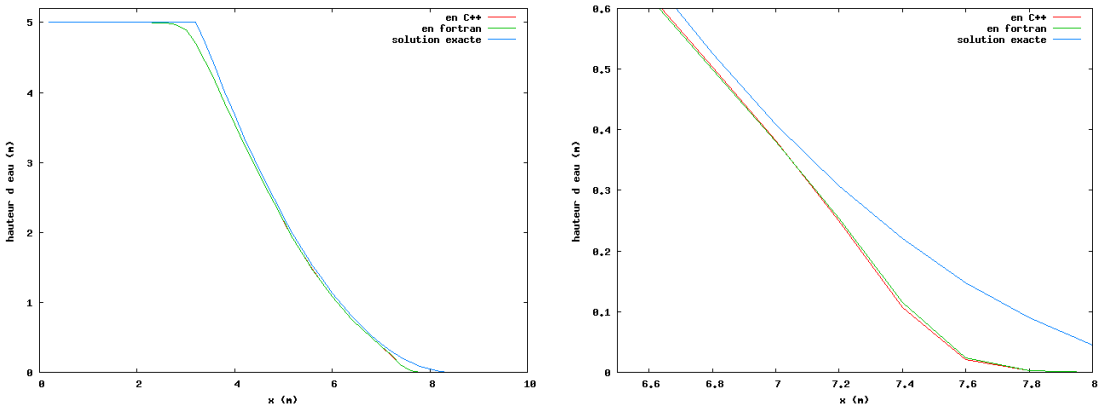


FIG. 14: Etat initial



(a) (b)

FIG. 15: Hauteur d'eau à $t = 0,25s$ (a) vue d'ensemble, (b) zoom

calculer correctement la transition sec/mouillé (certains schémas peuvent conduire à des instabilités voire à des hauteurs d'eau négatives).

En dépit d'un maillage assez grossier, la progression du frond sec/mouillé (fig. 15a) est assez bien calculée. Il faut zoomer sur la courbe (fig. 15b) pour distinguer de très légères différences entre les résultats du code c++ et du code fortran. Cette différence provient peut-être d'une différence entre le zéro du c++ et de celui du fortran, il conviendra de vérifier cette hypothèse.

Conclusion

Ce stage a été un bon moyen de mettre en application et d'approfondir les notions de classe et d'objet vues en cours de Java et en cours de c++. Les premiers tests effectués sur le code monodimensionnel semblent concluants. Cependant, il faudra poursuivre la validation de ce code sur les tests qui ont servi à valider le code 1d en fortran (prise en compte des frottements, de la pluie).

Le code monodimensionnel a permis de dégager les différentes techniques à mettre en oeuvre pour la réalisation du code 2d. Concernant le code 2d, il reste deux phases importantes le débogage et la validation. Des cas tests bien choisis, permettront de mettre rapidement le doigt sur les problèmes existants.

L'étape suivante consistera à développer des outils de pré-traitement et de post-traitement des données afin de faciliter les entrées et sorties du code. Les outils de post-traitement permettront des analyses plus poussées des résultats obtenus.

Références

[Audusse00] E. Audusse, M.-O. Bristeau, Benoit Perthame, Kinetic schemes for Saint-Venant equations with source terms on unstructured grids, *Inria report* **RR-3989**, (2000).

[Audusse04a] E. Audusse, A multilayer Saint-Venant model, *Inria report* **RR-5249**, (2004).

[Audusse04b] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein et B. Perthame, A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows, *SIAM J. Sc. Comp.* **25**, No. 6, (2004), 2050-2065.

[Audusse04c] E. Audusse, *Modélisation hyperbolique et analyse numérique pour les écoulements en eaux peu profondes*, Thèse, Université Paris VI (2004).

[Audusse05] E. Audusse, M.-O. Bristeau, A well-balanced positivity preserving "second-order" scheme for shallow water flows on unstructured meshes, *J. Comp. Phys.* **206**, (2005), 311-333.

[Bailly08] Y. Bailly, *Initiation à la programmation avec Python et C++*, Pearson, (2008).

[Bouchut04] F. Bouchut, *Nonlinear stability of finite volume methods for hyperbolic conservation laws, and well-balanced schemes for sources*, Frontiers in Mathematics, Birkhauser (2004).

[Bristeau01] M.-O. Bristeau, B. Coussin, Boundary conditions for the shallow water equations solved by kinetic schemes, *Inria report* **RR-4282**, (2001).

[Danaila03] I. Danaila, F. Hecht, O. Pironneau *Simulation numérique en C++*, Dunod, (2003).

[Delannoy06] C. Delannoy, *Programmer en C++*, Eyrolles, (2006).

[Delannoy07a] C. Delannoy, *Apprendre le C++*, Eyrolles, (2007).

[Delannoy07b] C. Delannoy, *Exercices en langage C++*, Eyrolles, (2007).

[Dupin05] S. Dupin, *Le langage C++*, Campus Press, (2005).

[ElBouajaji07] M. El Bouajaji, *Modélisation des écoulements à surface libre : étude du ruissellement des eaux de pluie*, Master2, Université de Strasbourg (2007).

[Esteves00] M. Esteves et al., Overland flow and infiltration modelling for small plots during unsteady rain : numerical results versus observed values, *J. Hydrol.* **228**, (2000), 265-282.

[Fiedler00] F.R. Fiedler et J.A. Ramirez, A numerical method for simulating discontinuous shallow flow over an infiltrating surface, *Int. J. Numer. Methods Fluids* **32**, (2000), 219-240.

[Gerbeau00] J.-F. Gerbeau, B. Perthame, Derivation of viscous Saint-Venant system for laminar shallow water ; numerical validation, *Inria report* **RR-4084**, (2000).

[Greenberg96] J. M. Greenberg, A. Y. Leroux, A well-balanced scheme for the numerical processing of source terms in hyperbolic equation, *SIAM Journal on Numerical Analysis* **33**, (1996), 1-16.

[Hervouet03] J.M. Hervouet, *Hydrodynamique des écoulements à surface libre, modélisation numérique avec la méthode des éléments finis*, Presses des Ponts et Chaussées, (2003).

[MacDonald95] I. MacDonald, M.J. Baines, N.K. Nichols, P.G. Samuels, Steady open channel test problems with analytic solutions, *Numerical Analysis Report* **2/95**, Department of Mathematics, University of Reading (1995), 49-63.

[Marche05] F. Marche, *Theorical and numerical study of shallow water models. Applications to nearshore hydrodynamics*, Thèse, Université de Bordeaux (2005).

[Marche07a] F. Marche, Derivation of a new two-dimensional viscous shallow water model with varying topography, bottom friction and capillary effects, *Eur. J. Mech. B Fluids* **26**, (2007), 49-63.

[Marche07b] F. Marche, P. Bonneton, P. Fabrie, N. Seguin, Evaluation of well-balanced bore-capturing schemes for 2D wetting and drying processes, *Int. J. Numer. Meth. Fluids* **53**, (2007), 867-894.

[Rousseau08] M. Rousseau, *Modélisation des écoulements à surface libre : étude du ruissellement des eaux de pluie*, Master2, Université de Nantes (2008).

Annexe 1 – Pré-rapport

Objectif du stage et réalisation attendue

L'objectif du stage consiste à écrire deux codes en C++ pour la simulation du ruissellement. Le premier permettant de faire des calculs en une dimension d'espace et le second en deux dimensions d'espace. Le travail demandé consiste à mettre au point des algorithmes permettant la simulation du ruissellement d'eau de pluie sur des surfaces agricoles. Ces algorithmes permettent différents paramètrages : plusieurs flux numériques, diverses conditions aux limites. Leur programmation nécessite une programmation objet tel le C++.
Enfin, il faut programmer des procédures permettant l'analyse des résultats.

Plan de travail prévisionnel

étape 1 [30/04] code 1D avec topographie avec un solveur (flux de Rusanov), sans frottement et un type de condition aux bords (bords liquides)
étape 2 [10/05] ajouter le traitement de la pluie
étape 3 [20/05] implémentation d'un solveur supplémentaire (HLL)
étape 4 [30/05] passer le schéma à l'ordre 2
étape 5 [10/06] diversification des conditions de bord (tenant éventuellement compte de la topographie et des frottements)
étape 6 [20/07] valider le code 1d et réitérer les étapes précédentes en dimension 2
étape 7 [30/07] coder des procédures de post-traitement
étape 8 [10/08] interface graphique permettant le paramètrage du modèle (GTK ou python ?), étape optionnelle
étape 9 [13/08] ajustement automatique de la CFL, étape optionnelle

Annexe 2 – Ruptures de barrage

Nous allons voir un cas test classique que j'ai mis en oeuvre : la rupture de barrage.
Soit $x_0 \in \mathbb{R}$ la position du barrage, nous avons une condition initiale en hauteur d'eau de type Riemann :

$$h(0,x) = \begin{cases} h_g \text{ pour } x \leq x_0 \\ h_d \text{ pour } x_0 < x \end{cases}, \tag{16}$$

avec

$$0 \leq h_d \leq h_g,$$

donc h_g (respectivement h_d) est la hauteur d'eau en amont (respectivement en aval) du barrage.
La condition initiale en vitesse est

$$u(0,x) = 0 \text{ pour } x \in \mathbb{R}.$$

Nous considèrons une rupture de barrage idéale : la rupture à $t = 0$ est instantanée et il n'y a pas de frottement sur le fond. Ce cas test a une solution analytique (fig. 16) que nous obtenons à l'aide des invariants de Riemann et de la méthode des caractéristiques.

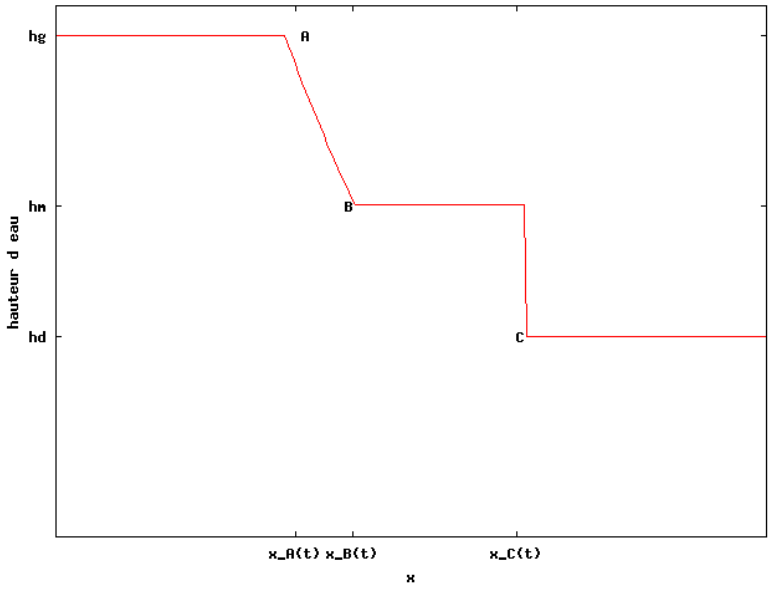


FIG. 16: Solution de Stoker au temps $t > 0$

Pour simplifier la description de cette solution analytique, nous considèrons que le barrage se trouve en $x_0 = 0$ (pour les simulations numériques, nous pourrons faire une translation de la solution).
Nous nous intéressons à la solution au temps $t > 0$ après la rupture. Nous devons observer une détente de forme parabolique (entre les points A et B) liant h_g à une valeur intermédiaire h_m, suivie d'un choc (en C) liant h_m à h_d.
Soit les célérités $c_g = \sqrt{gh_g}$, $c_m = \sqrt{gh_m}$, $c_d = \sqrt{gh_d}$ et u_m la vitesse de l'eau dans

la zone à l'état m (entre B et C).
A l'aide des invariants de Riemann et de la théorie des caractéristiques, nous obtenons la célérité et la vitesse pour $x \in [x_A, x_B]$

$$\begin{cases} c = \sqrt{gh} = \frac{2}{3}(c_g - \frac{x}{2t}) \\ u = \frac{2}{3}(\frac{x}{t} + c_g) \end{cases}, \tag{17}$$

d'où

$$h = \frac{4}{9g}(c_g - \frac{x}{2t})^2.$$

Comme le point B appartient à la fois à la zone m et à la détente, par les invariants de Riemann, nous avons

$$u_m = 2(c_g - c_m).$$

Enfin, à l'aide des invariants de Riemann et d'une relation de saut (relation de Rankine-Hugoniot) liant l'état m à l'état d, nous obtenons la vitesse de déplacement du choc

$$v_c = \frac{h_m u_m}{h_m - h_d},$$

ainsi qu'une équation permettant d'obtenir c_m donc h_m

$$-8{c_d}^2{c_m}^2({c_g}^2 - {c_m}^2)^2 + ({c_m}^2 - {c_d}^2)^2({c_m}^2 + {c_d}^2) = 0. \tag{18}$$

Au temps $t > 0$, les points A, B et C se trouvent respectivement en
- $x_A(t) = -c_g t$;
- $x_B(t) = (u_m - c_m)t = (2c_g - 3c_m)t$;
- $x_C(t) = v_c t$.

En résumé, nous avons

$$(h(t,x), u(t,x)) = \begin{cases} (h_g, 0) \text{ si } x \leq x_A(t) \\ (\frac{4}{9g}(c_g - \frac{x}{2t})^2, \frac{2}{3}(\frac{x}{t} + c_g)) \text{ si } x_A(t) \leq x \leq x_B(t) \\ (h_m, 2(c_g - c_m)) \text{ si } x_B(t) \leq x \leq x_C(t) \\ (h_d, 0) \text{ si } x_C(t) \leq x \end{cases}, \tag{19}$$

Cette solution est la solution de Stoker [Hervouet03]. Ce cas test permettra de tester la capacité du schéma à calculer la propagation d'un choc.
Il faut noter que si la rupture se fait sur sol sec $h_d = 0$, alors (18) devient :

$${c_m}^6 = 0,$$

donc

$$h_m = 0.$$

Dans ce cas, nous avons une perte d'hyperbolicité (hauteur d'eau nulle). Le choc disparaît, nous obtenons alors une détente (entre A et B) liant les états g et d(fig. 17).

Au temps $t > 0$, les points A et B de trouvent respectivement en
- $x_A(t) = -c_g t$;
- $x_B(t) = 2c_g t$.

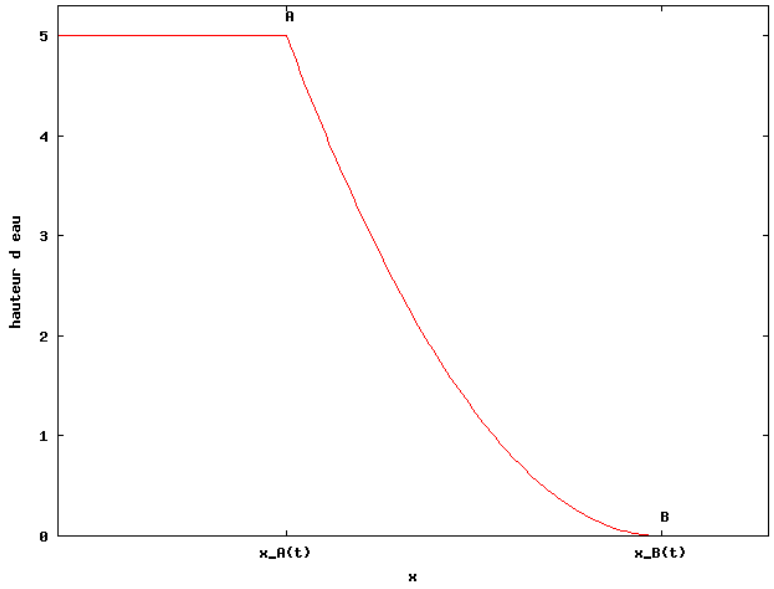


FIG. 17: Solution de Ritter au temps $t > 0$

et nous avons

$$(h(t,x),u(t,x)) = \begin{cases} (h_g, 0) \text{ si } x \leq x_A(t) \\ (\frac{4}{9g}(c_g - \frac{x}{2t})^2, \frac{2}{3}(\frac{x}{t} + c_g)) \text{ si } x_A(t) \leq x \leq x_B(t) \\ (0,0) \text{ si } x_B(t) \leq x \end{cases} , \qquad (20)$$

Cette solution est la solution de Ritter, elle permettra de vérifier la capacité du schéma à calculer une transition sec/mouillé.
Remarque : au niveau du barrage en $x = 0$, à tout temps $t > 0$, nous avons

$$(h(t,0),u(t,0)) = (\frac{4}{9}h_g, \frac{2}{3}c_g),$$

donc le profil de hauteur d'eau pivote autour du point $(0, \frac{4}{9}h_g)$.

Ecriture d'un code C++ pour la simulation en hydrologie

Olivier Delestre
Master CCI 2008 – Orléans

Résumé

Le système de Saint-Venant est un modèle courant pour la simulation du ruissellement. Nous avions un code pour la simulation des écoulements monodimensionnels en langage fortran, basé sur un schéma équilibre. Ce code avait été validé sur divers cas test aussi bien analytiques qu'expérimentaux. Cependant, ce code était mal structuré et pas modulaire, rendant difficile son évolution.

Dans ce stage, nous avons réalisé un code de ruissellement en c++. Tout d'abord développé en une dimension, nous avons ensuite développé un code simulant le ruissellement sur une surface. Une analyse du modèle mathématique nous a permis d'identifier les contraintes de programmation auxquelles nous allions être confrontées. Les simulations issues du programme en c++ 1d ont été comparées aux simulations du programme fortran. Il reste quelques différences à expliquer dans les résultats avant que le code c++ 1d puisse être considéré comme validé. Les corrections à apporter au code c++ 1d s'appliqueront directement au code c++ 2d.

Mots-clefs

Modélisation numérique, ruissellement, C++, Saint-Venant, schéma équilibré, 2D

Abstract

Shallow water system is a typical model for the simulation of overland flow. We had a 1d fortran program to simulate one dimension flow, based on a well-balanced method. This code was validated on several analytical and experimental test cases. Nevertheless, this code was ill-structured and did not allow for the development of modules. As such this code became difficult to develop further.

During this internship, we developed a c++ code for overland flow. First developed in one dimension, with then develop a code simulating the overland flow over a surface. An analysis of the mathematical model allowed us to point out the programming constraints that we will have to face. Simulations based on the c++ 1d program were compared with the simulations of the fortran program. Some differences in the results remaind to be explained before the c++ 1d code could be considered as validated. The changes carried out on the c++ 1d code will be directly implemented into the c++ 2d code.

Keywords

Numerical modelling, overland flow, C++, shallow-flow equations, well-balanced scheme, 2D