UCLouvain

epl

École polytechnique de Louvain

Numerical study of the flow over a weir: verification of widely used design rules

Auteur: **Thibault Prévost**
Promoteur: **Sandra Soares-Frazão**
Lecteurs: **Francesco Contino, Robin Meurice**
Année académique 2020–2021
Master [120] : ingénieur civil des constructions

Remerciements

Je souhaite avant tout remercier Mme la professeure S. Soares-Frazão, ma promotrice, pour son support tout au long de ce mémoire et les nombreuses réunions qui m'ont guidé dans la bonne direction et contribué à mon organisation.

Je remercie également mes lecteurs, MM. le professeur F. Contino et R. Meurice. Mr le professeur F. Contino, m'a notamment guidé dans la maîtrise du programme OpenFOAM à travers plusieurs rencontres. Ses précieux conseils m'ont permis de mener à bien ce projet. Mr. R. Meurice m'a aussi épaulé lors de à la rédaction de ce mémoire.

Finalement je remercie ma famille, pour leur soutien durant mes années d'études et la réalisation de ce mémoire ainsi que pour leur relecture attentive de ce travail.

Abstract

From antiquity to the present day, humans have attached great importance to water. Many cities are installed along the rivers. Rivers provide food and drink, facilitate transportation, and can be a source of energy. However, they are also subject to the time and the weather changes. Humans have sought to tame rivers, making their use easier and more efficient.

Weirs are therefore important structures with many uses. Indeed they can be used in rivers to maintain a certain level of water upstream for navigation or to know the flow of a watercourse thanks to the theory of hydraulics. They can also be used at the head of a spillway to regulate the quantity of water contained in the reservoirs.

The rules currently used for sizing weirs and calculating their flow rate - water depth ratio are based on experimental studies in the field and in the laboratory. The objective of this thesis is to use a modeling software, OpenFOAM, in order to simulate the flows on a weir and to confirm the commonly used design rules.

A digital model, produced using OpenFOAM software, is used to digitize the flows with various conditions. I will first study Creager's standardized model. I will then study the influence of an approach slope on the upstream side of the weir. Finally, I will be interested in the impact of a hydraulic jump downstream of the spillway on the water flow and height. The data collected are then compared with the data usually used, which results from an in situ measurement campaign carried out by the American institute "United States Bureau of Reclamation" or USBR.

This thesis confirms the relationships describing the influence of the slope for the upstream face of the weir on the one hand. On the other hand, this thesis presents a numerical method to study the influence of a hydraulic jump on the flow of a weir, with preliminary results consistent with the theory of weirs.

Résumé

De l'antiquité jusqu'à nos jours, l'être humain a toujours apporté à l'eau une grande importance. De nombreuses villes sont d'ailleurs installées le long des rivières. Les rivières apporte boisson et nourriture, facilitent le transport, et peuvent être source d'énergie. Cependant, elles sont aussi soumises aux aléas du temps et de la météo. Les êtres humains ont cherché à dompter les rivières, rendant leur utilisation plus facile et plus efficiente.

Les déversoirs sont dès lors des structures importantes, aux nombreuses utilisations. En effet ils peuvent servir dans les rivières pour maintenir un certain niveau d'eau en amont pour la navigation ou permettre de connaître le débit d'un cours d'eau grâce à la théorie de l'hydraulique. Ils peuvent aussi servir en tête d'un évacuateur de crues pour réguler la quantité d'eau contenue dans les barrages.

Les règles actuellement utilisées pour dimensionner les déversoirs et calculer leur rapport débit - hauteur d'eau sont basées sur des études expérimentales sur le terrain et en laboratoire. L'objectif de ce mémoire est d'utiliser un logiciel de modélisation, OpenFOAM, afin de simuler les écoulements sur un déversoir et de confirmer les règles communément utilisées.

Un modèle numérique, réalisé à l'aide du logiciel OpenFOAM, est utilisé pour numériser les écoulements avec diverses conditions. Je vais d'abord étudier le modèle standardisé de Creager. Je vais ensuite étudier l'influence d'une pente d'approche sur la pente amont du déversoir. Finalement je m'intéresserai à l'impact d'un ressaut hydraulique à l'aval du déversoir sur le débit et la hauteur d'eau. Les données recueillies sont ensuite confrontées aux données utilisées habituellement, qui résulte d'une campagne de mesures in situ réalisée par l'institut américain "United States Bureau of Reclamation" ou USBR.

Ce mémoire confirme les relations décrivant l'influence du pente pour la face amont du déversoir d'une part. D'autre part ce mémoire présente une méthode numérique pour étudier l'influence d'un ressaut hydraulique sur l'écoulement d'un déversoir, avec des résultats préliminaires cohérents avec la théorie des déversoirs.

Table des matières

1 Introduction

1.1 Contexte

L'eau douce est une ressource précieuse. Elle ne représente que 3 % des eaux sur terre. Et de ces 3 %, seulement 0.3 % est disponible dans les lacs et rivières. Le reste fait partie des eaux souterraines (30.8 %) ou des glaciers et neiges permanentes (68.9 %).[6] Cette eau douce est utilisée dans de nombreux domaines : domestique (hygiène, boisson, cuisine,...), agriculture, élevage, industrie, transport, production hydro-électrique.

L'eau fait aussi partie des forces de la nature qui a modelé notre planète, force qui se révéle parfois destructrice de notre environnement. Entre 2007 et 2016, on compte en moyenne 5624 personnes tuées et 84,8 millions de victimes par an suites aux inondations à travers le monde[3]. Selon le dernier rapport du GIEC, ces inondations vont être de plus en plus fréquentes à cause du changement climatique, tout comme vont l'être les sécheresses [4].

La gestion des cours d'eau est donc un enjeu majeur pour l'avenir. Dès lors, il apparaît très important d'étudier son cycle : la quantité d'eau dont nous disposons dans les rivières, lacs et autres plans d'eau et de dimensionner les infrastructures hydrauliques au mieux pour répondre aux différents défis.

Parmi ces ouvrages hydrauliques, nous retrouvons les barrages et les déversoirs. Les premiers permettent de créer de grandes réserves d'eau, des espaces tampons en cas de fortes pluies ou encore de produire de l'électricité. Les seconds servent à réguler l'écoulement de l'eau. Ils permettent en effet de garantir un certain niveau d'eau dans les rivières ou réservoirs à l'amont de la structure. Ils permettent aussi d'étudier le débit d'un cours d'eau, nous permettant d'allouer au mieux la ressource entre les différents acteurs (prise d'eau, navigation, etc.).

Pour que ces structures fonctionnent correctement, il est important de les dimensionner. Pour cela, il existe un certain nombre de règles reprises dans le document "Design of small dams" de l'United States Departement of the Interior, Bureau of Reclamation [7]. Cet organisme a pris toute une série de mesures relatives aux écoulements sur les déversoirs et les évacuateurs de crues et les a compilées dans un document sur les bonnes pratiques de dimensionnement. En parallèle, on utilise de plus en plus de simulateurs informatiques afin de prévoir le comportement des structures.

1.2 Objectif

L'objectif de ce mémoire est double. Tout d'abord, on va vérifier qu'OpenFOAM, un programme open source de simulation de calcul de dynamique des fluides peut être utilisé efficacement pour simuler les écoulements à surface libre sur un déversoir. Pour cela, on va utiliser un cas de base d'écoulement sur un déversoir de Creager pour lequel la théorie nous donne de bons résultats.

Je vais ensuite vérifier l'influence de différentes situations à l'amont et à l'aval du déversoir sur les courbes hauteur d'eau - débit.

Á l'amont, je vais étudier l'impact d'une pente d'approche du déversoir.

Pour l'aval, je vais étudier l'influence que peut avoir un ressaut hydraulique sur l'écoulement.

Le paramètre étudié est nommé le coefficient de débit. C'est un paramètre qui lie les deux valeurs, hauteur d'eau à l'amont du déversoir et débit, et qui comprend de nombreuses incertitudes dues aux hypothèses posées pour construire le modèle mathématique. Cela va permettre de vérifier la validité des mesures prises par l'USRB au XXème siècle, sur le terrain et en laboratoire, et donc de valider les valeurs empiriques utilisées lors du dimensionnement des déversoirs et autres évacuateurs de crues.

1.3 État de l'art

Les déversoirs sont des obstacles à l'écoulement de la rivière. Ce sont également des points de contrôle pour lesquels on connaît la forme exacte du lit de la rivière. Cela a permis de créer une relation débit-hauteur d'eau qui peut être utilisée pour évaluer la quantité d'eau qui s'écoule dans la rivière.

Un des premiers ingénieurs à documenter ce phénomène est le français Henri Bazin. En 1865 il développe les équations principales pour le déversoir à paroi mince représenté à la figure 1.

Ce déversoir est fait de telle façon que l'écoulement n'est en contact qu'avec l'arête supérieure, et donc la surface est minimale. Le jet est ensuite projeté vers l'aval. Il est très efficace d'un point de vue hydraulique. La faible aire de contact freine en effet un minimum le courant. L'inconvénient majeur de ce déversoir est sa stabilité. Il peut arriver que la bulle d'air sous la lame d'eau disparaisse. Le courant va alors se coller au déversoir et créer une pression négative sur sa face intérieure, ce qui peut mener à sa rupture. Pour cette raison, ce genre de déversoir n'est utilisé que pour de petits cours d'eau.

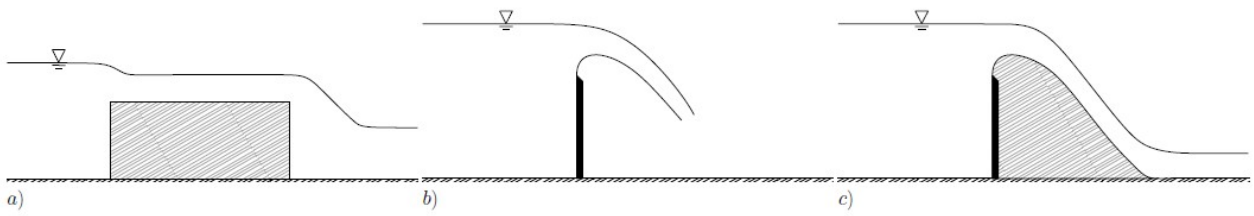

FIGURE 1 – Formes de déversoirs : a) à seuil épais, b) à seuil mince, c) déversoir de Creager.

Pour régler ce problème de stabilité, la première solution trouvée à été le déversoir à seuil épais.

Ce type de déversoir prend le problème à l'envers. Très large par sa nature, il est également très stable. Il représente cependant un obstacle beaucoup plus important sur le parcours de la rivière. Un déversoir à seuil épais va imposer une hauteur d'eau plus importante en amont qu'un déversoir à seuil fin, et ce pour un même débit. Lorsqu'il y a beaucoup d'eau à évacuer (crue, orages, ...) cela peut s'avérer problématique.

Au XXème siècle, W.P. Creager (1917) puis Scimeni (1930) ont développé le profil du déversoir standardisé, aussi appelé le déversoir de Creager. Ce profil remplit l'espace vide sous la lame d'eau d'un déversoir à paroi mince, de béton afin d'améliorer la stabilité de la structure en limitant l'impact sur le débit. En donnant une trajectoire balistique au parement aval du déversoir, la structure épouse la forme de l'écoulement. A son débit nominal, la pression sur le parement aval, et donc le frottement, est négligeable.

L'écoulement au-dessus des déversoirs de Creager a été étudié de manière extensive par la suite, via des expériences en laboratoires et des études sur le terrain, par le "United States

Bureau of Reclamation" (ultérieurement référé sous le nom USBR). Ces études ont permis de créer les lois de dimensionnement des déversoirs, lois que l'on peut retrouver entre autres dans "Design of small dams" de l'USBR.

Ces règles de bonnes pratiques sont utilisées lors du dimensionnement des déversoirs. Jusqu'à récemment, pour les projets plus importants, des modèles réduits étaient également construits en laboratoire pour vérifier les structures. Avec l'augmentation des capacités de calculs et l'avancement des ordinateurs, on commence également à utiliser des modèles numériques. Ceux-ci sont de plus en plus précis et rapides, les rendant pertinents et concurrençant les modèles réduits. Les modèles numériques résolvent les équations de Navier-Stokes, souvent en utilisant la méthode des volumes finis. Le premier modèle de ce type, sur un déversoir existant, a été réalisé en Pologne en 2016. Ils ont créé un modèle 2D et 3D d'un barrage et de son évacuateur de crue afin de définir la courbe hauteur d'eau-débit (Herrera-Granados et Kostecki, 2016).

1.4 Structure du mémoire

La suite de ce mémoire va se diviser en deux parties importantes. Le premier chapitre va d'abord rappeler la théorie des déversoirs. Je vais commencer par définir ce qu'est un déversoir et les différents types principaux. Puis, j'approfondirai la géométrie particulière des déversoirs de Creager.

Ensuite je vais démontrer la formule de calcul de débit sur un déversoir. Cette dernière amène le concept de coefficient de débit, qui est un coefficient reprenant les différentes incertitudes qui séparent les mathématiques de la pratique. Ce sont les valeurs de ce coefficient de débit, μ qui seront discutées par après à l'aide des simulations numériques. Je vais également amener dans cette partie l'influence de l'inclinaison du parement amont du déversoir et son utilité. De même, je vais parler de l'influence du niveau d'eau aval et comment un ressaut hydraulique peut être néfaste pour le déversoir.

Dans le chapitre suivant, je vais présenter le modèle numérique réalisé à l'aide du logiciel OpenFOAM. Je vais également développer comment je dessine la forme du déversoir, et je lance une simulation. Enfin j'explique les étapes de post-processing pour obtenir les résultats afin de pouvoir en tirer des conclusions.

Dans le troisième chapitre, je développe les résultats pour les trois cas étudiés, à savoir :

- le cas général de déversoir de Creager 'influence de la pente du parement amont du déversoir
- l'influence d'un ressaut hydraulique à l'aval sur l'efficacité du déversoir

Je terminerai par une conclusion résumant les résultats principaux des simulations.

2 Théorie

2.1 Théorie sur les déversoirs et évacuateurs de crues

Les déversoirs et les barrages sont des structures hydrauliques conçues pour régulariser l'écoulement des cours d'eau. Tous deux sont construits en travers du lit de la rivière. Ils ont cependant des rôles différents.

Les barrages sont construits pour créer des retenues d'eau et peuvent être couplés avec des centrales hydroélectriques afin de produire de l'énergie. Un barrage n'est pas conçu pour être complètement submergé. L'eau va s'écouler soit à travers l'édifice, par des turbines, soit sur un évacuateur de crues. Ceux-ci sont semblables aux déversoirs, à la différence qu'ils sont couplés à des systèmes de dissipation d'énergie. En effet, sur les grands édifices, l'écoulement doit être ralenti avant de pouvoir rejoindre le lit de la rivière. Les évacuateurs de crues peuvent aussi mener à un canal pour rediriger l'écoulement, par exemple vers une vallée adjacente. Il arrive alors que l'évacuateur soit submergé, ce qui va influencer le débit qui peut s'y écouler. Ce cas est approfondi dans ce mémoire.

FIGURE 2 – Évacuateur de crue latéral, Hoover Dam, USA.

Les déversoirs sont quant à eux prévus pour être entièrement submergés. Ils permettent d'imposer un niveau minimum à une rivière à l'amont de la structure. Ils peuvent aussi permettre de rediriger la rivière.

FIGURE 3 – Déversoir, Shrewsbury, Royaume-Uni.

Comme les déversoirs créent une zone de contrôle pour laquelle on connaît la largeur et la profondeur d'eau, c'est un bon endroit pour y calculer le débit d'une rivière.

Pour cela il existe des liens entre le niveau d'eau à l'amont du déversoir et le débit. C'est cette relation qui va être plus amplement analysée dans ce mémoire.

La forme du déversoir va avoir une influence importante sur la relation hauteur-débit. Il y a donc une série de critères importants à respecter :

- La stabilité de la structure, que le débit soit important ou faible, voire nul.
- Tendre vers un optimum hydraulique : maximiser le rapport débit sur hauteur d'eau, cela équivaut à maximiser le coefficient de débit μ
- Être économique : la structure doit minimiser la quantité de matières premières nécessaires.

Les deux premiers points sont contradictoires.

La structure est très stable pour un déversoir à seuil épais, comme on peut le voir sur la figure 4. Ce seuil épais est un véritable obstacle à l'écoulement. Sa forme n'est pas hydrodynamique et va donc ralentir beaucoup plus le cours d'eau. Qui dit ralentissement dit augmentation de la hauteur d'eau. Cette augmentation de la hauteur d'eau va être bénéfique pour la stabilité de la structure de par son poids. Cependant, pour une même hauteur d'eau, le débit sera beaucoup plus faible. Le coefficient de débit est relativement petit. Cela peut s'avérer problématique par exemple dans le cas d'évacuateur de crue où le but est d'empêcher un barrage d'être submergé et où l'on veut donc maximiser l'évacuation d'eau. Le coefficient de débit d'un déversoir à seuil épais est estimé à $\mu = 0.385$. Comme valeur de comparaison, un déversoir de Creager pour sa charge de nominale à un coefficient de débit $\mu = 0.496$.

Le second point d'attention, atteindre un optimum hydraulique, est quant à lui maximisé dans le cas du déversoir à seuil mince, comme illustré sur la figure 4. Par contre, si la bulle d'air vient à disparaître et que le jet d'eau se retrouve collé contre la paroi, nous observons l'apparition de pressions négatives sur la face aval de la structure. Ces pressions négatives déstabilisent le déversoir et peuvent l'endommager. Elles sont donc à éviter.

Un bon compromis est le déversoir de Creager présenté à la figure 4 .

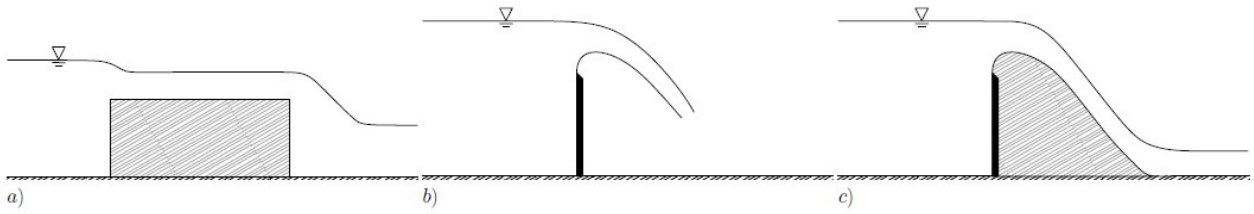

FIGURE 4 – Formes de déversoirs : a) à seuil épais, b) à seuil mince, c) déversoir de Creager.

2.2 Déversoir de Creager

Basé sur le déversoir à paroi mince, le déversoir de Creager remplit l'espace sous la lame d'eau. La partie aval du déversoir a donc une pente basée sur une trajectoire de projectile.

En partant d'un déversoir à paroi mince, on compare la nappe déversante à la trajectoire d'un projectile.

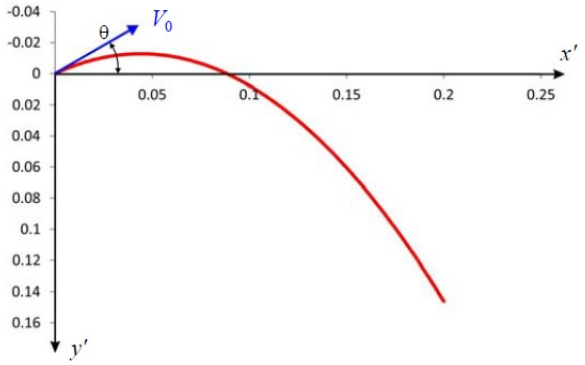

FIGURE 5 – Exemple de trajectoire balistique tel que calculé pour un jet d'eau projeté suite à un déversoir à paroi mince.

Selon l'axe x :

$$x = V_0 cos(\theta) t \tag{1}$$

Selon l'axe y :

$$y = \frac{g}{2} t^2 - V_0 sin(\theta) t \tag{2}$$

On peut combiner les deux équations, ce qui donne :

$$y = \frac{g}{2} \frac{x^2}{V_0^2 cos^2(\theta)} \tag{3}$$

Lorsque la pente aval épouse correctement l'écoulement, nous observons une pression nulle. De plus à la crête l'angle $\theta = 0$.

La vitesse est donnée par l'équation de Bernouilli.

On néglige les pertes de charge et on prend comme hypothèse que la vitesse à l'amont du déversoir est négligeable.

Nous avons donc l'équation qui s'écrit entre un point A en amont du déversoir et un point M au niveau du déversoir à mi-hauteur de la couche d'eau :

$$h + 0 = 0 + \frac{p_M}{\gamma} + \frac{V_M^2}{2g} \tag{4}$$

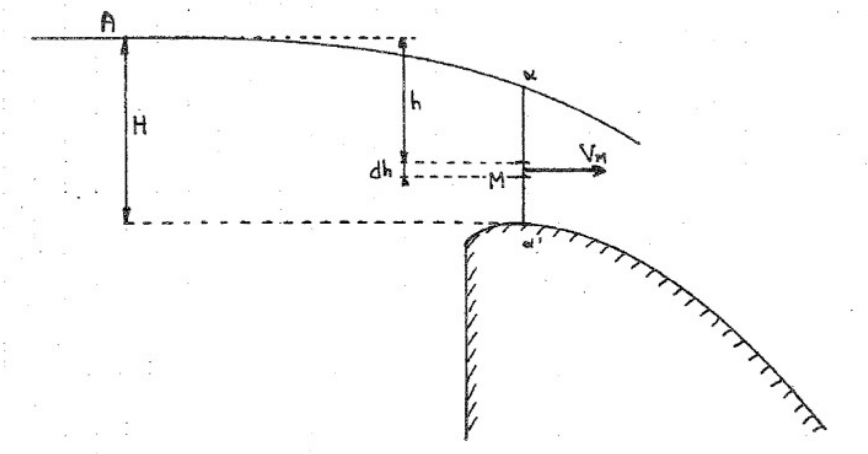

FIGURE 6 – Illustration pour les points de l'équation de Bernouilli sur un déversoir.

On néglige en première approximation la pression, ce qui équivaut à supposer que les trajectoires de chaque filet fluide sont indépendantes et semblables à celle d'une particule libre.

Nous prenons $H_0 = 2h$ par hypothèse du point M. H_0 est la charge en amont du déversoir. Il nous reste donc l'expression pour la vitesse :

$$V_0 = \sqrt{2gh_M} = \sqrt{gH_0} \tag{5}$$

Nous pouvons remettre cette vitesse dans l'équation de la trajectoire d'un projectile (3), ce qui donne :

$$\frac{y}{H_0} = \frac{1}{2}(\frac{x}{H_0})^2 \tag{6}$$

Nous obtenons donc la forme aval du déversoir en fonction de la hauteur d'eau de fonctionnement nominale.

En pratique, l'expression ne peut pas être utilisée telle quelle, ceci est dû aux hypothèses que l'on a prises :vitesse amont nulle, pertes de charges négligeables, filets fluides indépendants.

De plus il faut prendre un coefficient de sécurité contre les dépressions en cas de surcharge du déversoir.

Après de nombreuses mesures, l'ingénieur américain, W.P. Creager a modifié les coefficients, arrivant à la formule que j'ai utilisée pour dimensionner les déversoirs dans mes simulations, à savoir :

$$\frac{y}{H_0} = 0.47(\frac{x}{H_0})^{1.8} \tag{7}$$

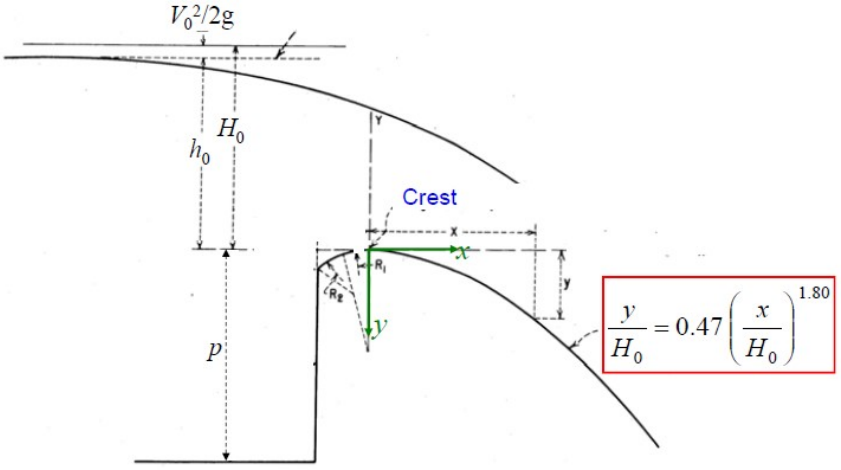

FIGURE 7 – Forme de Creager.

Cette formule est la plus communément utilisée lors du dimensionnement des déversoirs standardisés.

Cette formule est donc basée pour une hauteur d'eau amont H_0 correspondant au débit Q_0 de dimensionnement. Si le débit diminue, on obtient de légères surpressions, mais qui sont négligeables car pour une couche d'eau inférieure à celle de dimensionnement.

Et en cas de fortes crues, le coefficient de débit est alors favorisé. Il faut alors faire attention à ce que le débit ne soit pas trop important car on risquerait de voir apparaître de la cavitation (La cavitation est formation de bulles de vapeurs dans un liquide suite à la diminution rapide de la pression locale en dessous de la pression de vapeur du liquide ce qui donne des pressions négatives. La cavitation a un effet abrasif important qui peut rapidement endommager les structures.)

2.3 Coefficient de débit μ

Il est intéressant de connaître le débit d'eau d'une rivière, mais il est assez ardu de le mesurer.

Le calcul du débit d'un déversoir se ramène à l'équation générale des déversoirs, fondée sur l'intégration de la vitesse calculée par l'équation de Bernouilli (4) Celle-ci s'écrit entre le point A et le point M comme indiqué sur la figure 6 en supposant une vitesse nulle en A et une perte négligeable. Pour rappel :

$$h = \frac{P_M}{\gamma} + \frac{V_M^2}{2g} \tag{8}$$

Comme cité plus haut, lors de l'équation (4), on néglige la pression. On en tire la vitesse V_M et on peut l'intégrer pour trouver le débit :

$$Q = \int_0^H V_M L dh = \int_0^H \sqrt{2gh} L dh \tag{9}$$

Avec L la largeur du déversoir.

On voit que les bornes d'intégrations sont excessives par rapport à la section $\alpha\alpha'$. L'intégrale donne le résultat suivant :

$$Q = \frac{2}{3} m L \sqrt{2g} H^{\frac{3}{2}} \tag{10}$$

Avec le coefficient $m < 1$ qui tient en compte la contraction du jet dans le plan vertical, de la surestimation des vitesses ($\frac{P_M}{\gamma}$ négligé), des bornes d'intégrations et des pertes de charges éventuelles.

Pour plus de facilité, on écrit généralement l'équation comme :

$$Q = \mu L\sqrt{2g}H^{\frac{3}{2}} \tag{11}$$

Avec $\mu = \frac{2}{3}m$ le coefficient de débit. Comme on l'a vu, ce coefficient reprend toutes les incertitudes de m.

Dans ce mémoire, je m'intéresse à ce coefficient en particulier. Le "United States Bureau of Reclamation" aussi appelé l'USBR a fait de nombreuses expériences pour l'étude du Boulder Canyon Dam. Durant ces expériences, ils ont mesuré les valeurs du coefficient de débit et ont obtenus différents graphes en fonction des cas.

Il faut distinguer le cas dans lequel le déversoir fonctionne à sa charge nominale, pour laquelle nous avons des pressions sur le parement aval quasi nulles. Nous avons alors le coefficient μ_0 tel que :

$$Q_0 = \mu_0 L\sqrt{2g}H_0^{\frac{3}{2}} \tag{12}$$

Pour les autres cas nous avons une modification du μ

2.3.1 Vitesse d'approche négligeable, parement amont vertical

Dans le cas de vitesse d'approche négligeable, avec le profil de Creager, on a un coefficient de débit pour la charge nominale H_0 qui vaut :

$$\mu_0 = 0.496 \tag{13}$$

Ce coefficient μ_0 varie légèrement selon les modèles et les mesures prises. Il faudra donc en premier lieu le recalculer.

Les valeurs de μ mesurées sont reprises sur le graphe 8 On y voit que μ est plus grand lorsque les charges sont supérieures à la charge nominale, ce qui s'explique par l'accélération due aux pressions négatives sur le parement aval du déversoir.

Les valeurs de μ ont été théorisées suivant la courbe donnée par :

$$\mu = \mu_0(\frac{H}{H_0})^{0.17} \tag{14}$$

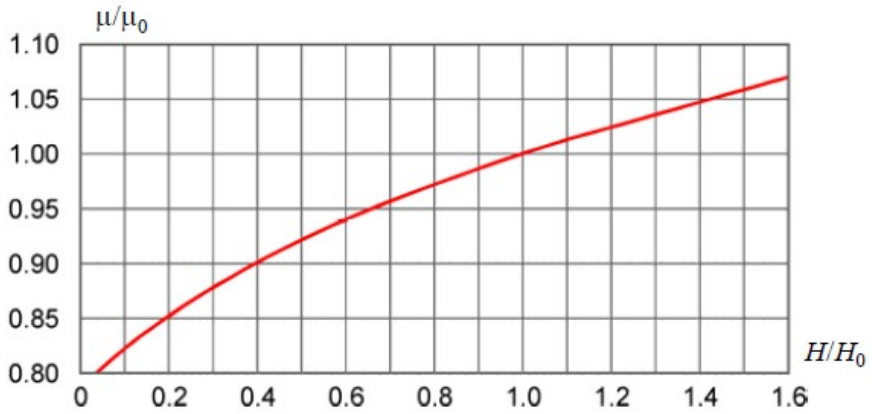

FIGURE 8 – Relation entre le coefficient de débit et la charge à l'amont.

2.3.2 Influence de l'inclinaison du parement amont

On a vu dans l'équation (14) que le coefficient de débit était influencé par la charge. La charge comprend la hauteur d'eau, mais également la vitesse d'approche de l'écoulement.

Dans le cas de la pente de parement amont verticale, on avait émis l'hypothèse que la vitesse d'approche était négligeable, ici ce n'est plus le cas. Le coefficient de débit μ va donc être corrigé par un paramètre κ tel que :

$$\mu_{0,FaceAmontIncline} = \kappa \mu_{0,FaceAmontVerticale} \tag{15}$$

Le coefficient κ dépend de la vitesse d'approche du courant, qui elle-même va dépendre de la profondeur de la pelle p par rapport à la charge nominale H_0 du déversoir comme on le voit sur le graphe 9

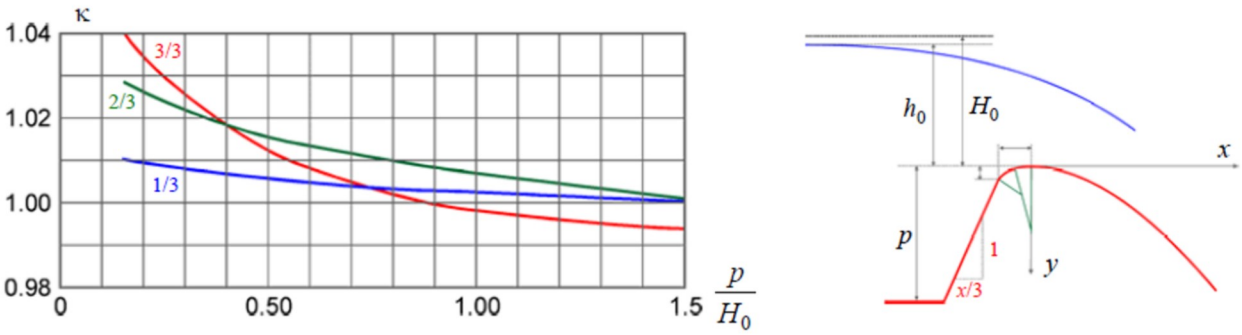

FIGURE 9 – Relation entre le coefficient dû à la vitesse d'approche κ et la charge amont.

2.3.3 Influence du niveau d'eau aval

Le dernier cas étudié dans ce mémoire est le cas d'un déversoir pris dans un ressaut hydraulique. Dans certains cas, il se peut que le niveau d'eau à l'aval du déversoir soit relativement important. Nous avons par exemple le cas dans un évacuateur de crue lié à un canal latéral qui mène à une vallée adjacente ou dans le cas d'un déversoir sur une rivière comme illustré à la figure 10. Le niveau aval va alors avoir une influence sur l'efficacité de l'évacuateur et donc sur son coefficient de débit.

FIGURE 10 – Déversoir de rivière influencé par la hauteur d'eau aval.

Nous retrouvons la structure typique d'un déversoir en rivière sur la figure 11. Nous observons alors qu'à l'aval du déversoir se forme un ressaut hydraulique. En fonction de la hauteur de ce ressaut et de sa distance par rapport au déversoir, il va avoir plus ou moins d'influence sur l'écoulement.

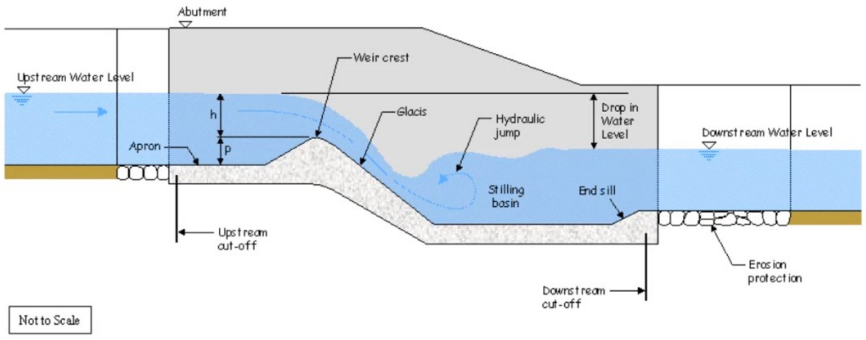

FIGURE 11 – Schéma d'un déversoir avec ressaut hydraulique.

Pour discuter de l'influence du niveau aval du déversoir, je vais me baser sur le schéma présenté à la figure 12. Sur celui-ci, on observe H_0 la charge hydraulique à l'amont et h_0 le niveau d'eau. Comme pour les autres cas, on prend l'hypothèse que ces deux valeurs sont très proches, la vitesse à l'amont du déversoir étant quasi nulle. Ensuite on a h_{ds} qui est le niveau d'eau à l'aval. Enfin h_d qui peut être subdivisé entre l'énergie cinétique et les pertes de charges (qui généralement sont faibles).

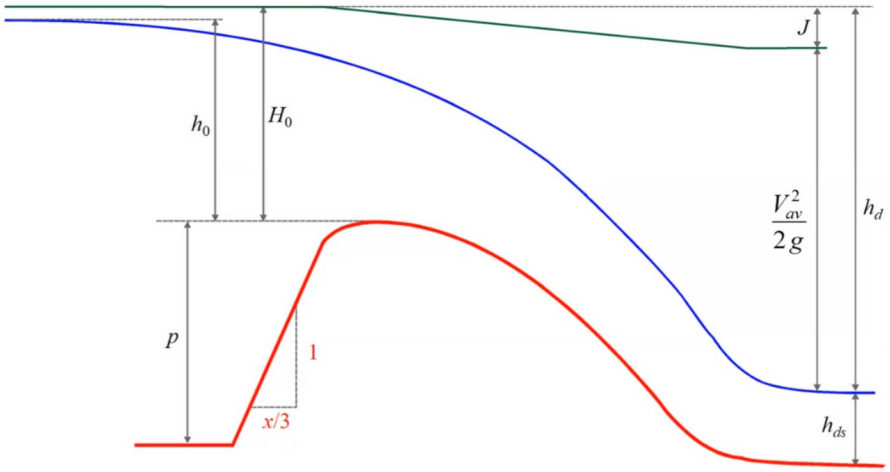

FIGURE 12 – Schéma d'un déversoir avec illustration de la hauteur d'eau H_0, la pelle, p, les hauteurs aval h et h_d.

Le premier graphe, visible à la figure 13 est un graphe complet de l'influence de niveau aval sur l'écoulement.

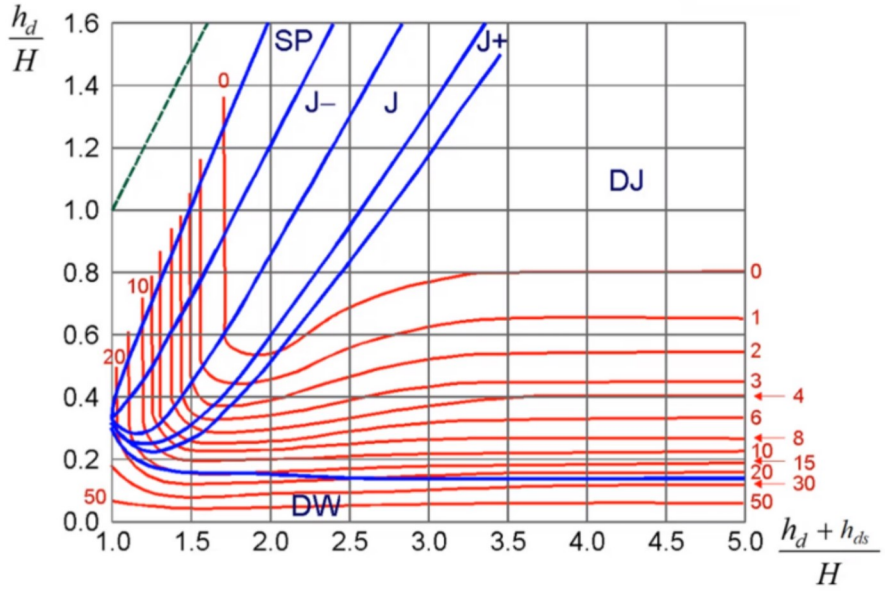

FIGURE 13 – Influence de l'aval, fonction de h_d et de la hauteur du radier.

En abscisse, nous avons le niveau du radier. Cette valeur peut être très importante, par exemple dans le cas d'un évacuateur de crue au sommet d'un barrage.

En ordonnée, nous avons le degré de submersion du déversoir, représenté par la fraction $\frac{h_d}{H}$.

La partie au-dessus de la ligne verte représente un flot nul, c'est donc un cas qui n'est pas intéressant à analyser.

Ensuite, la partie entre la ligne verte et la première ligne bleue représente un flot impossible. En effet, nous avons à l'intersection avec la première ligne bleue et l'axe des ordonnées :

$$\frac{h_d + h_{ds}}{H} = 1\frac{h_d}{H} = \frac{1}{3} \tag{16}$$

La première ligne nous montre qu'on a un radier au niveau du sommet du déversoir Si on rassemble les deux équations, on obtient

$$h_{ds} = \frac{2}{3}H = h_c \tag{17}$$

On retrouve par sa définition, h_c qui représente la hauteur critique. On a donc $H = E_c$ la charge d'eau qui équivaut à l'énergie critique. On ne peut donc pas avoir un ratio h_d/H plus grand car on serait sur une expression qu'on ne peut pas représenter sur le diagramme de l'énergie spécifique.

Ensuite nous avons les différentes zones du diagramme.

- SP = écoulement super critique
- $J^{(-,\,,+)}$ = ressaut hydraulique (faible, normal, fort)
- DJ = ressaut hydraulique noyé
- DW = le déversoir est complètement noyé

Les lignes rouges correspondent au pourcentage de réduction du coefficient de débit dû à l'influence de l'aval.

Pour clarifier un peu les données, nous pouvons prendre une ligne verticale sur le graphe 13, on a alors une hauteur de radier fixe.

Dans le nouveau graphe, visible sur la figure 14, le coefficient de réduction du coefficient de débit est donné en fonction de h_d. On y trouve la valeur pour ξ qu'on utilise :

$$\mu = \xi \; \mu_{\text{free fall spilling}} \tag{18}$$

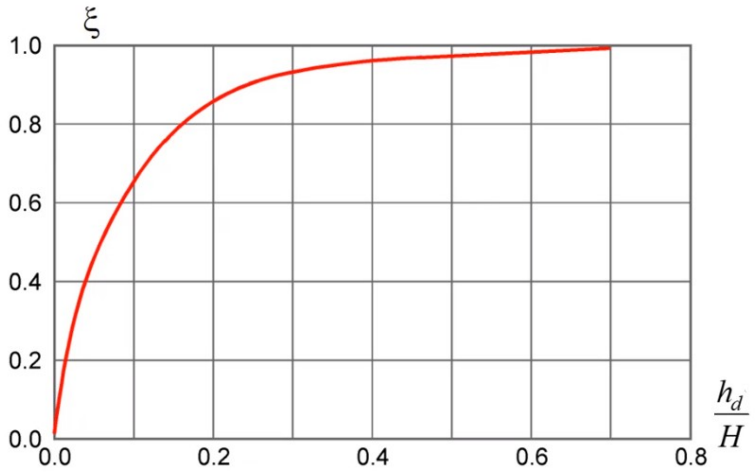

FIGURE 14 – Influence de l'aval, fonction de h_d.

3 Modèle numérique - OpenFOAM

Dans cette section, je vais développer la création du modèle numérique utilisé pour les simulations, ainsi que le programme de simulation OpenFOAM.

Afin de réaliser les simulations, il faut d'abord créer un modèle de déversoir. J'ai modélisé un déversoir de Creager, en utilisant un script Matlab pour la forme exacte. Pour cela j'ai

suivi les indications que l'on peut retrouver sur la figure 7 pour laquelle j'ai utilisé une valeur de $H_0 = 0.5$ et une hauteur de pelle $p = 3 * H_0 = 1.5$.

Ce modèle, discrétisé en une centaine de points via un document Excel, est ensuite intégré dans le programme Autodesk Inventor. Ce programme me permet, en partant de ce profil 2D, de créer une configuration 3D du déversoir. Ce modèle est ensuite sauvé au format de stéréolithographie .slt pour qu'il puisse être intégré dans le maillage OpenFOAM.

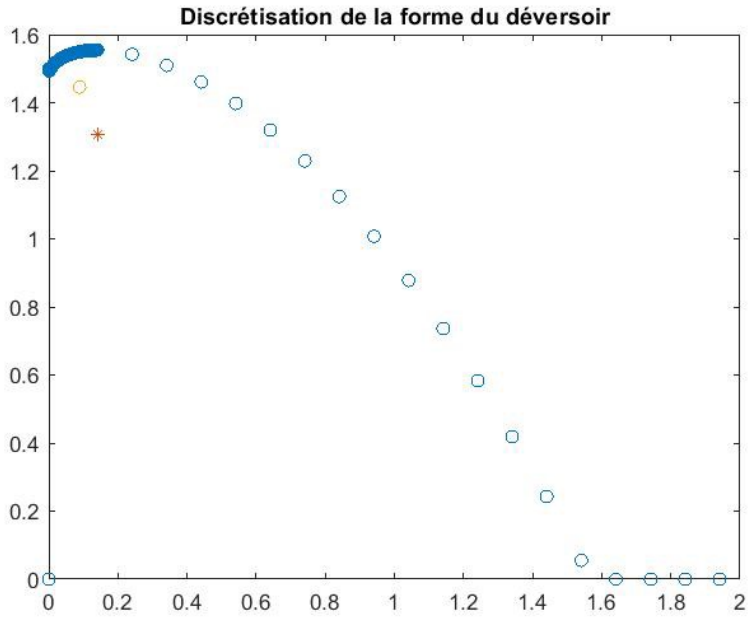

FIGURE 15 – Design du déversoir Matlab.

Sur Autodesk Inventor, j'ajoute une courbe à l'aval du déversoir pour le relier à la suite du canal. En effet, si on le garde trop vertical, on va observer une cassure qui va causer des turbulences. La forme du déversoir est donc adaptée. En général, on considère pour ce genre de courbe que le rayon de courbure doit être supérieur à 10 fois la hauteur d'eau de dimensionnement afin d'éviter d'éventuelles surpressions.

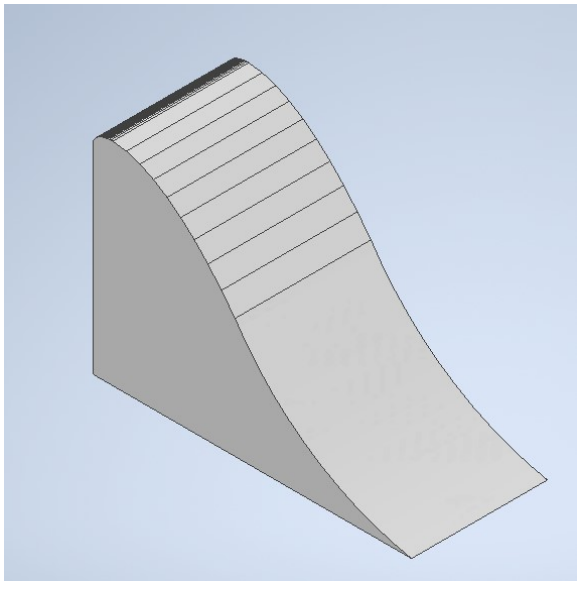

FIGURE 16 – Modèle 3D Autodesk Inventor.

3.1 OpenFOAM

OpenFOAM (pour "Open-source Field Operation And Manipulation") est une boite d'outils C++ pour le développement de solveurs numériques personnalisé et d'utilitaires de pré/post-traitement pour la résolution de problèmes de mécanique des milieux continus, notamment la dynamique des fluides numériques. Il n'y a cependant pas de pré-requis nécessaire en C++ afin de le prendre en main. C'est un instrument utilisé dans de nombreux domaines d'ingénierie, mais particulièrement bien conçu pour résoudre des problèmes de dynamique des fluides de tous types (turbulences, transferts de chaleur, réactions chimiques, etc.). OpenFOAM contient une série de solveurs, chacun spécialisé pour les différents problèmes qui peuvent se poser. Il existe aussi une série d'utilitaires qui permettent de manipuler les données et permettent d'analyser en pré- et post-processing.

Pour la prise en main du programme, OpenFOAM propose de nombreux tutoriels. Ceux-ci permettent de découvrir les nombreuses possibilités à travers des cas simples et variés. Pour reproduire les résultats explicités dans ce mémoire, il est intéressant de commencer par le tutoriel weirOverFlow qui utilise le solveur "interFOAM".

3.2 Création d'une simulation "InterFOAM"

Dans cette partie, je vais approfondir l'utilisation du solveur "interFOAM". Je vais d'abord expliquer les équations qui vont être résolues, ensuite parler du domaine et des conditions limites et enfin expliquer les outils supplémentaires utilisés, par exemple pour vérifier la hauteur d'eau dans le canal.

3.2.1 Equations

Tous les solveurs d'OpenFOAM utilisent la méthode aux volumes finis. Le principe de la méthode de volumes finis est que pour résoudre les équations différentielles, le domaine de calcul est divisé en un nombre finis de volumes de contrôles, ou cellules. Chaque variable est évaluée au centre de ce volume de contrôle. La valeur en ce centre vaut la valeur moyenne sur la cellule concernée.

Pour simuler un écoulement à surface libre, on utilise "interFOAM". C'est un outil de calcul qui va considérer la couche d'air et la couche d'eau comme deux fluides incompressibles, isothermiques et immiscibles. Cela signifie que les propriétés d'un fluide dans une région donnée sont constantes. L'interface entre les deux fluides va lui être traité séparément.

Cet outil de calcul résout les équations de Navier-Stokes. On a donc

• L'équation de continuité avec la densité constante

$$\frac{\delta u_j}{\delta x_j} = 0 \tag{19}$$

Avec

- u_j la vélocité dans la case j
- x_j l'abscisse de la case j

• L'équation de conservation des moments

$$\frac{\delta(\rho u_i)}{\delta t} + \frac{\delta}{\delta x_j}(\rho u_j u_i) = -\frac{\delta p}{\delta x_i} + \frac{\delta}{\delta x_j}(\tau_{ij} + \tau_{t_{ij}}) + \rho g_i + f_{\sigma i} \tag{20}$$

avec

- u la vélocité
- g_i la constante de gravité
- p la pression
- τ_{ij} les contraintes visqueuses
- $\tau_{t_{ij}}$ les contraintes de turbulences
- $f_{\sigma i}$ la tension de surface

Comme le problème est résolu pour deux fluides qui parfois se retrouvent ensemble dans une cellule, on doit définir la variable α. C'est la proportion d'eau dans une cellule.

α vaut 1 lorsque la cellule est entièrement remplie d'eau et vaut 0 quand la cellule est remplie d'air. À l'interface on observe donc un α entre 0 et 1dont il faut tenir compte dans les équations. La densité est alors calculée comme

$$\rho = \alpha\rho_1 + (1 - \alpha)\rho_2 \tag{21}$$

Avec

- ρ_1 la densité de l'eau
- ρ_2 la densité de l'air

On a également la tension de surface $f_{\sigma i}$ qui est modélisée comme une force de surface continue et est calculée comme :

$$f_{\sigma i} = \sigma\kappa\frac{\delta\alpha}{\delta x_i} \tag{22}$$

Avec

- σ la constante de tension de surface
- κ la courbure

κ est calculé avec la fonction suivante :

$$\kappa = \frac{\delta n_i}{\delta x_i} = -\frac{\delta}{\delta x_i}\left(\frac{\delta\alpha/\delta x_i}{|\delta\alpha/\delta x_i|}\right) \tag{23}$$

Enfin on a l'équation qui régit l'interface, afin de savoir où elle se trouve. On résout donc pour α :

$$\frac{\delta\alpha}{\delta t} + \frac{\delta(\alpha u_j)}{\delta x_j} = 0 \tag{24}$$

3.2.2 Génération du maillage

La génération du maillage, première étape d'une simulation, est très importante. Cette génération va en effet être une étape décisive et doit donc être bien pensée. Un maillage mal élaboré est une source d'erreur fréquente lors des simulations.

La première partie utilise l'outil "blockMesh". C'est un outil facile à prendre en main qui permet de créer un domaine de simulation à base de points et de droites. Il est donc utile pour des formes assez simples. La dernière version permet également d'y entrer des fonctions pour définir les frontières du domaine, mais je n'ai pas utilisé cette fonction. Donc tout d'abord j'ai défini un canal rectangulaire.

Openfoam permet de faire des simulations en 2D en utilisant une seule cellule de profondeur (axe z) et en n'imposant pas de conditions sur les faces latérales.

Une fois le domaine défini, j'ai utilisé l'outil "surfaceFeatureExtract" qui permet de lire l'objet 3-D dans le fichier .stl et de définir les différentes faces. A savoir dans ce cas-ci de lier le déversoir à une surface lisse de type wall. Ensuite avec la fonction "snappyHexMesh", j'ai intégré le modèle de déversoir dans le canal préalablement compilé. On retrouve les fichiers OpenFOAM permettant de mettre en oeuvre ces étapes dans l'annexes E.1. On peut également retrouver plus d'information dans le guide d'utilisation d'OpenFOAM [5].

3.3 Conditions de bord

Les conditions de bords sont divisés en 6 catégories. Il y a les conditions d'"atmosphere" sur le dessus du maillage, les conditions d"inlet" à l'amont, "outlet" à l'aval. Il y a "emptyFaces" de part et d'autre du canal pour avoir une simulation 2D et enfin les conditions de "wall" et de "weir", qui sont essentiellement les mêmes, une pour le fond du canal et l'autre pour le déversoir. Des illustrations montrant les emplacements de ces différentes conditions frontières du domaines sont visibles dans l'annexe F.

La face, "inlet", amont sert ici de prise d'eau. Le débit est imposé et les pressions sont considérées hydrostatiques. Le champ de vitesse à l'amont est calculé pour que le débit d'entrée soit réparti sur toute la hauteur d'eau. Comme la hauteur d'eau est variable, ce champs de vitesse sera recalculé tout au long de la simulation afin de maintenir un débit constant. De plus ; il est important de mettre un niveau d'eau suffisamment élevé lors de l'initialisation de la simulation pour avoir un écoulement et non un jet pressurisé.

La face aval, "outlet", est considérée comme la sortie. Dans le cas général et dans les cas de test de la pente du pavement amont du déversoir, les conditions de flux à travers cette face sont laissées libres, pour ne pas influencer l'écoulement. Le champs de vitesse est donc mis sur 0 dans les conditions initiales et est par la suite calculé par le débit constant. La pression est définie comme une condition de bord nul de von Neumann, ce qui crée un profil hydrostatique. Pour les simulations avec une hauteur d'eau imposée, j'ai imposé une condition de vitesse sur l'écoulement. En effet, le débit est connu car égal au débit entrant (conservation de la masse). Il suffit donc d'imposer une vitesse pour avoir une hauteur d'eau donnée. Cette condition est valable dans le cas des filets fluides parallèles. J'y reviendrai dans la partie concernée.

Enfin, la face supérieure, "atmosphere" est considérée comme une atmosphère ouverte. C'est la seul face pour laquelle le champs de pression n'est pas dans une condition d'écoulement libre. Pour cette face, on impose une pression d'une atmosphère.

3.3.1 Post processing

Une fois les simulations réalisées il faut encore pouvoir les visualiser et analyser les données qui en ressortent. Le programme phare pour visualiser la situation OpenFOAM est un autre

programme OpenSource, nommé Paraview. Il permet de voir la simulation évoluer pour les différents pas de temps.

Pour ce qui est de l'analyse de données, j'ai utilisé un outil OpenFOAM est le "linesample", que l'on retrouve dans l'annexe E.1, dans la partie controlDict. Cet outil permet de prendre les valeurs d'un paramètre, dans ce cas α, des cellules le long d'une verticale. Ce sont ces valeurs qui vont me permettre de connaître le niveau d'eau en une certaine abscisse. En effet, lors de l'étape de post-prossessing, un script Matlab, disponible à l'annexe A, me permet d'intégrer les valeurs de α sur la verticale et en déduire la hauteur d'eau. J'ai utilisé le programme Matlab car j'ai eu l'occasion de travailler avec celui-ci durant mes études. Il est cependant tout à fait possible de faire la même chose avec des scripts Python.

4 Simulations des écoulements

Dans cette section je vais développer les différentes simulations que j'ai réalisées grâce à OpenFOAM.

J'explique comment je lance la simulation et je précise les modifications effectuées pour chaque cas, les conditions de bords spécifiques, etc.

Ensuite je développe les résultats analysés grâce à de petits programmes Matlab. Je mettrai en valeur les éléments importants et leurs implications.

Enfin je conclus avec les données et informations qu'on peut recouper avec la théorie. J'explique alors les leçons à en tirer.

4.1 Cas général, écoulement sur un déversoir à parement amont vertical

Le premier cas permet de vérifier la validité du modèle numérique OpenFOAM. Pour ce cas-ci, j'ai utilisé la géométrie de base d'un déversoir de Creager, avec un parement amont vertical. La hauteur d'eau utilisée pour dimensionner le déversoir est de 0.5 [m]. La pelle est assez importante, valant 3 fois la valeur H_0 de charge de dimensionnement. Ça permet d'avoir un écoulement pratiquement nul en amont du déversoir comme lors des hypothèses de calcul théorique.

Pour ce qui est des conditions aux frontières, on a un débit fixe à l'amont, et un écoulement libre à l'aval.

La première chose à faire est de créer le déversoir. D'abord sa forme exacte puis l'objet en 3D comme annoncé dans la section 3.2. Une fois cet élément placé dans le fichier de simulation sur "interFOAM", il faut créer le maillage. J'utilise d'abord la fonctionnalité "blockMesh" pour créer le canal dans lequel viendra s'intégrer le déversoir. Ensuite la fonctionnalité "surfaceFeaturesExtract" permet de décomposer l'élément 3D en surfaces que OpenFOAM peut lire. La fonctionnalité "snappyHexMesh" permet ensuite d'assembler les deux éléments. Nous avons alors le maillage dans lequel on va faire tourner la simulation. Il est visible sur la figure 17.

Les dimensions du canal sont les suivantes :

- longueur (x) : 15 [m], 100 cellules
- hauteur (y) : 4[m], 40 cellules
- largeur (z) : 1[m], 1 cellule

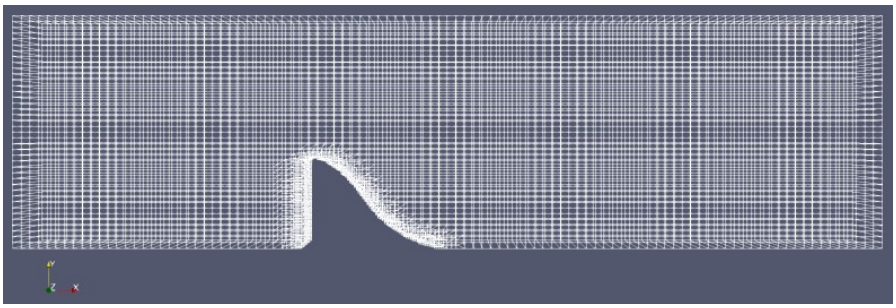

FIGURE 17 – Maillage pour le premier cas.

Le canal est assez long de part et d'autre du déversoir pour que l'écoulement puisse se stabiliser sans être influencé par les conditions de bords.

Une fois le maillage effectué, setFields permet de remplir le canal au niveau d'eau voulu au temps 0. Il est intéressant de remplir suffisamment le canal pour que la simulation ne prenne pas trop de temps à remplir le réservoir en amont du déversoir et puisse se stabiliser au plus vite. Dans mes simulations, je remplis le réservoir à l'amont du déversoir sur une hauteur de 1.5 [m]. De ce fait, l'écoulement peut rapidement s'effectuer sur le déversoir et se stabiliser.

Enfin, on peut lancer la simulation avec la commande "interFoam".

Les résultats sont ensuite visualisés sur ParaView tels qu'affichés sur la figure 18.

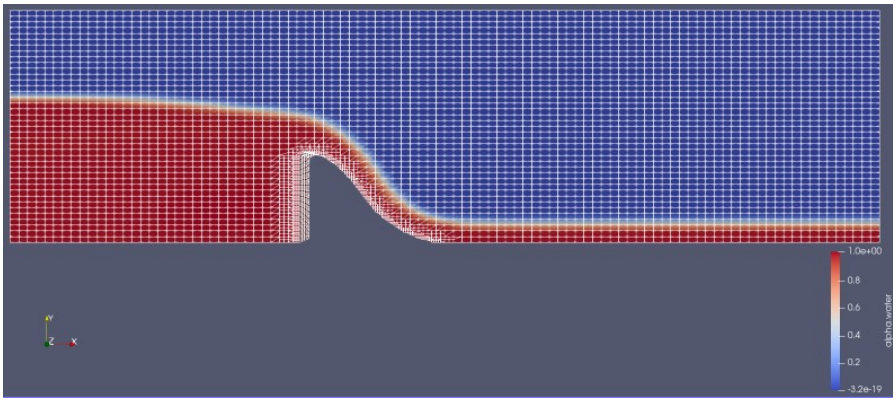

FIGURE 18 – Visualisation du résultat de la simulation.

Pour ce qui est des mesures de niveau d'eau, les données sont extraites au fur et à mesure de la simulation avec l'outil linesample. Ils sont ensuite lus avec un petit script Matlab comme visible dans l'annexe A.

Les données des différentes simulations sont rassemblées dans un second script Matlab (voir Annexe B) afin de tracer les différents graphes liant hauteur d'eau, débit et coefficient de débit.

Comme c'est le débit qui est contrôlé et non la charge (ou niveau d'eau) en amont, la simulation a été répétée pour différents débits. Pour chaque débit, la hauteur d'eau à l'amont a alors été calculée, à l'aide de Matlab, donnant le tableau 1 avec en première ligne les débits en $[m^3 s^{-1}]$ et en seconde ligne la charge à l'amont en $[m]$

Q	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5	2.75	3
H	0.333	0.446	0.553	0.648	0.738	0.730	0.918	0.983	1.044	1.108	1.174	1.221
mu	0.293	0.379	0.412	0.432	0.445	0.450	0.456	0.464	0.476	0.484	0.488	0.502

TABLE 1 – Relation débit - Hauteur d'eau dans le cas à parement amont vertical.

Par interpolation linéaire, il est alors facile de trouver le débit Q_0 correspondant au H_0 de dimensionnement. Et delà, on trouve la valeur pour μ_0 :

$$\mu_0 = 0.4001 \tag{25}$$

On observe sur la figure 19 que pour un même débit, les simulations OpenFOAM calcule une hauteur d'eau plus importante à l'amont du déversoir. Cela confirme que le coefficient de débit expérimentale doit être plus faible que celui apporté par la littérature.

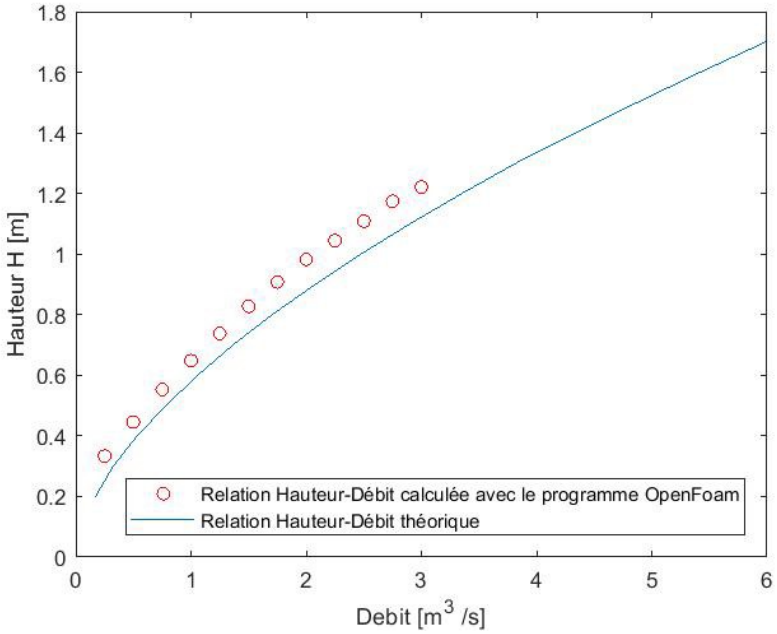

FIGURE 19 – Relation entre débit et charge amont.

Le μ_0 est un peu plus faible que dans les mesures de Creager, mais voyons également comment il évolue comparé à la théorie, sur la figure 20.

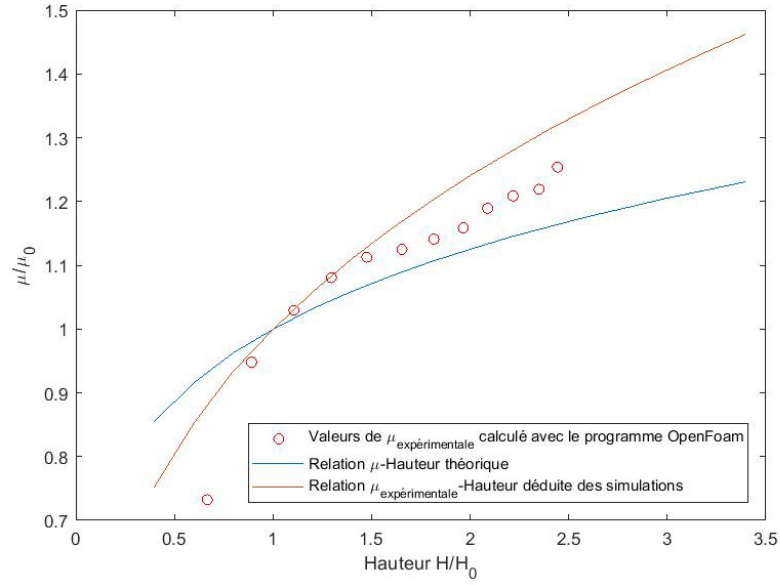

FIGURE 20 – Relation μ/μ_0 théorique et expérimentale.

Nous observons que la courbe n'est pas de même pente. J'ai donc cherché le nouveau facteur qui lie μ et H_0.

$$X = \frac{log(\frac{\mu}{\mu_0})}{log(\frac{H}{H_0})} \tag{26}$$

Cette formule me permet de trouver une valeur de X pour chaque simulation, reprise dans le tableau 1. J'ai alors pris $\bar{X}$ l'exposant moyen. Il nous permet de formuler une nouvelle relation pour l'équation (26). Pour rappel, nous avions vu à l'équation (14) que X valait 0.17. Avec les valeurs expérimentales, nous trouvons un nouveau

$$\bar{X} = 0.3144$$

D'autre part, nous observons que le coefficient de débit μ donné dans le programme OpenFOAM, est inférieur à celui calculé lors des expériences de l'USBR. Cela n'est pas pénalisant car on aura donc un niveau d'eau supérieur pour un même débit, ce qui place le dimensionnement du côté de la sécurité lors de l'analyse du déversoir.

Lorsque l'on trace cette nouvelle courbe, on observe qu'elle est assez proche des valeurs de $\mu_{exprimentale}$ lorsque que l'on est proche des valeurs de dimensionnement du déversoir. Cependant pour les valeurs extrêmes, cette relation ne semble pas convenir. Cette nouvelle courbe évolue plus vite que le coefficient de débit. Analysons pour voir si cela va poser problème.

Avec l'exposant X et le graphe 20 nous observons que le coefficient de débit numérique grandit plus rapidement que le théorique. Nous aurons donc un point pour lequel les deux courbes du graphe Hauteur - débit 19 vont s'inverser. Il convient alors de voir si ce point de croisement est suffisamment élevé pour qu'il ne rentre pas dans les cas possibles en situation réelle.

On démontre ensuite par l'écriture d'un programme Matlab que ce point de croisement intervient pour un $\frac{H}{H_0} = 4.8$. Les déversoirs sont généralement conçu pour aller jusqu'à des charges de $\frac{H}{H_0}$ entre 1.6 et 2. Nous avons donc une marge confortable. Les simulations OpenFOAM restent donc du côté de la sécurité pour la gamme de hauteurs d'eau d'utilisation.

4.2 Influence de l'inclinaison du tablier amont

Dans cette partie je modifie la pente du tablier en amont du déversoir. En effet, dans le premier cas nous avions un tablier amont vertical. Le déversoir se présente donc comme un mur, un obstacle perpendiculaire au flux de la rivière. On a remarqué qu'incliner la face amont du déversoir permet de guider le flux vers son sommet et d'améliorer le coefficient de débit. Nous pouvons donc observer la nouvelle géométrie sur la figure 21.

Nous étudions trois inclinaisons différentes. Ce sont ces inclinaisons qui ont été étudiées et répertoriées dans les documents de l'USBR.

- $\frac{1}{3} \longrightarrow \alpha = 18.43°$
- $\frac{2}{3} \longrightarrow \alpha = 33.69°$
- $\frac{3}{3} \longrightarrow \alpha = 45°$

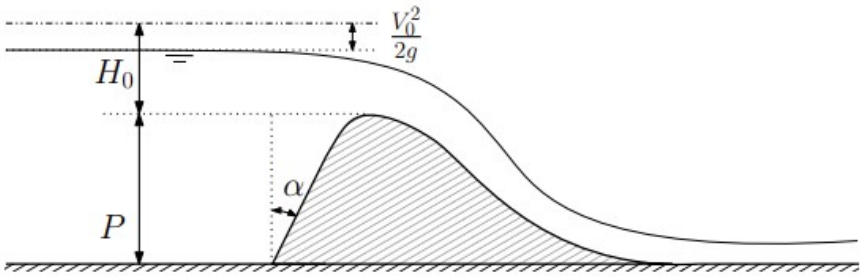

FIGURE 21 – Inclinaison de la face amont du déversoir.

Dans la figure 22 suivante on observe les courbes théoriques de correction du coefficient de débit. Nous retrouvons les valeurs de κ en fonction de la hauteur d'eau à l'amont du déversoir. Ici, κ est défini comme :

$$\mu_{0,\text{ face amont inclinée}} = \kappa\ \mu_{0,\text{ face amont verticale}} \tag{27}$$

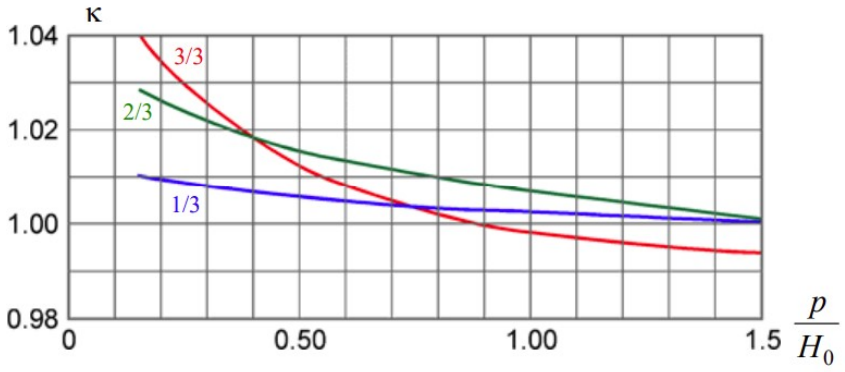

FIGURE 22 – Coefficient de correction κ tenant compte de la pente amont.

Le but de cette section est donc de simuler l'écoulement du cours d'eau pour les différentes géométries et vérifier la cohérence du graphique.

Les conditions de bords restent les mêmes que dans le cas de base. La seule différence est donc la géométrie du déversoir. On peut observer les différents maillage pour les trois cas analysés sur la figure 23.

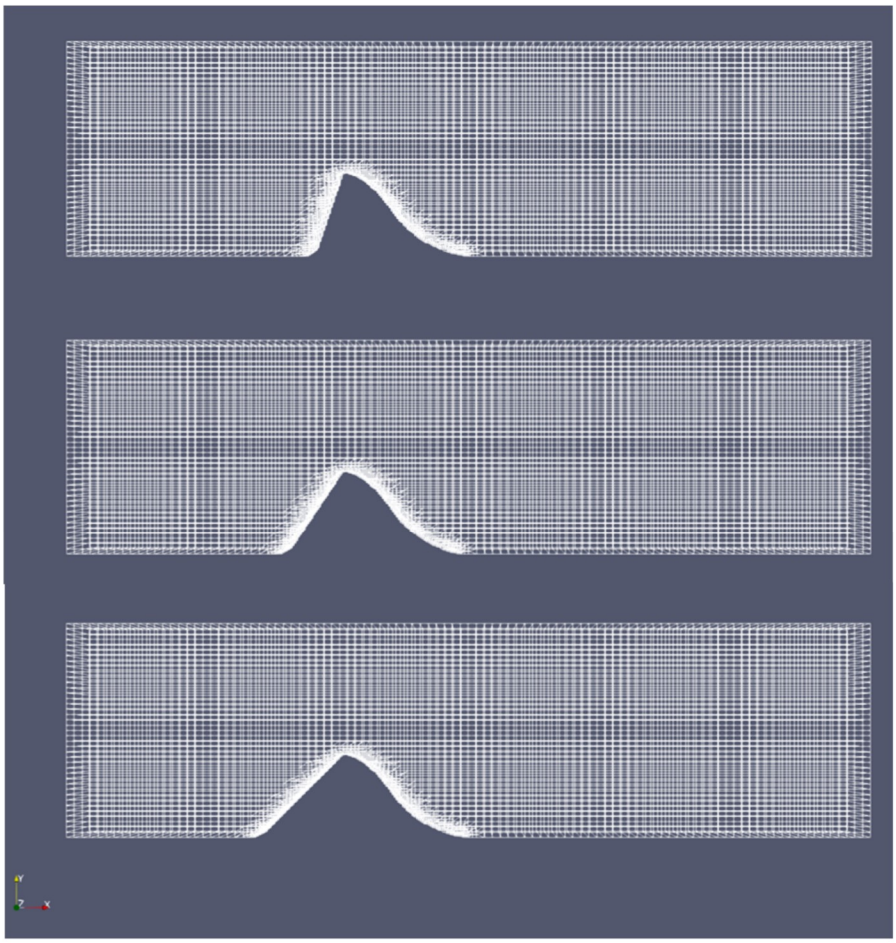

FIGURE 23 – Maillages pour les cas avec pente à l'amont de 1/3 , 2/3 et 3/3.

4.2.1 Analyse des résultats

Pour cette partie, j'ai refait les simulations en faisant varier le débit, comme dans le cas initial. La différence est que j'ai fait varier la forme du déversoir pour observer l'influence de la pente amont. Pour visualiser les résultats, j'ai ramené ceux-ci dans un tableau2

Q	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5	2.75	3
$\alpha : 0$	0.333	0.446	0.553	0.648	0.738	0.730	0.918	0.983	1.044	1.108	1.174	1.221
$\alpha : 1/3$	0.332	0.458	0.565	0.656	0.737	0.8152	0.891	0.959	1.021	1.086	1.139	1.196
$\alpha : 2/3$	0.334	0.454	0.560	0.656	0.737	0.813	0.885	0.946	1.015	1.080	1.131	1.192
$\alpha : 3/3$	0.329	0.446	0.551	0.651	0.735	0.811	0.884	0.953	1.012	1.079	1.128	1.188

TABLE 2 – Hauteur d'eau pour les différentes pentes amont.

Ces résultats sont illustrés sur le graphe 24. On peut y voir que plus la pente est importante, plus efficace sera l'écoulement. En effet, pour un même débit, nous avons une hauteur d'eau plus faible.

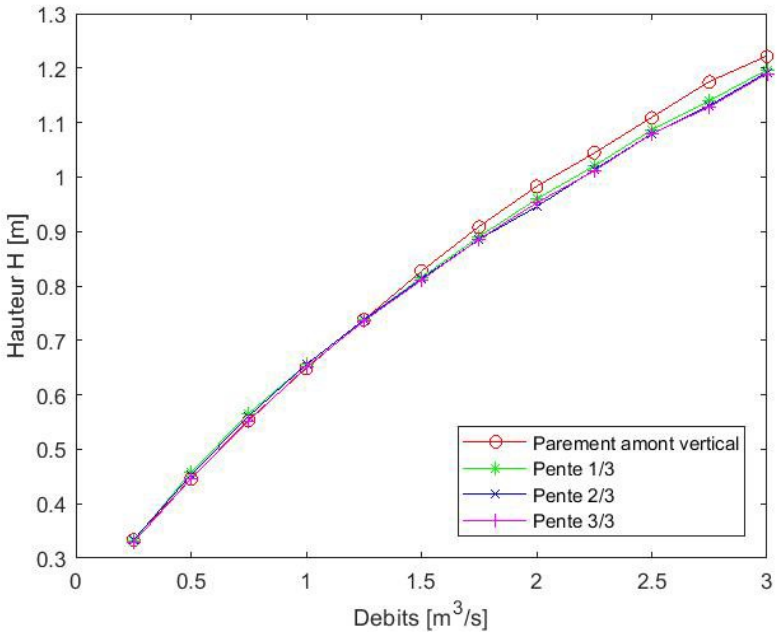

FIGURE 24 – Relation entre débit et charge amont pour les différentes pentes.

Nous pouvons alors calculer les valeurs du coefficient de débit et leurs évolutions par rapport au cas de base. A nouveau nous pouvons les rassembler dans un tableau 3

Q	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5	2.75	3
$\mu_{0/3}$	0.293	0.379	0.412	0.432	0.445	0.450	0.456	0.463	0.476	0.483	0.487	0.502
$\mu_{1/3}$	0.294	0.364	0.398	0.424	0.446	0.460	0.470	0.481	0.492	0.498	0.511	0.518
$\mu_{1/3}$	0.293	0.369	0.404	0.425	0.446	0.462	0.475	0.490	0.497	0.503	0.516	0.520
$\mu_{3/3}$	0.298	0.378	0.413	0.429	0.448	0.464	0.475	0.485	0.499	0.503	0.518	0.523

TABLE 3 – Valeur de μ pour les différentes pentes amont.

Ces résultats peuvent eux aussi être visualisés sur un graphique 25. Dans le cadre de mes simulations, j'ai fait varier le débit et donc la hauteur d'eau H plutôt que la hauteur de pelle p.

On peut cependant rapprocher le graphe théorique et celui que j'ai obtenu lors des simulations. Dans les deux cas nous observons que lorsque le ratio $\frac{p}{H}$ est faible, donc pour de grands débits, la pente amont est bénéfique à l'écoulement. Un déversoir avec une pente amont fortement inclinée sera donc plus efficace lors d'une crue du cours d'eau.

On retrouve aussi les valeurs qui tendent vers 1 lorsque la pelle devient très importante. En effet, dans ce cas là, la vitesse de l'écoulement est fortement réduite et on se retrouve dans un cas proche du cas initial avec la face amont verticale. On se rappelle qu'on avait fait l'hypothèse, lors du calcul du débit de la rivière en fonction de la hauteur d'eau (11), que l'eau a une vitesse nulle à l'amont du déversoir.

L'importance d'une face amont inclinée va donc dépendre de la vitesse d'approche de l'eau dans le cours d'eau. Plus celle-ci est importante, plus il va être intéressant d'avoir cette rampe d'accès pour le déversoir. Il est donc intéressant d'en prévoir pour les petits cours d'eau où il n'est pas possible de créer un bassin profond à l'amont d'un déversoir, et qui sont à risque de crues importantes.

Il est par contre beaucoup moins intéressant d'avoir une rampe inclinée dans le cas d'un

évacuateur de crue positionné au sommet d'un barrage. Dans ce cas ci, l'importance du bassin à l'amont va rejoindre l'hypothèse de vitesse nulle.

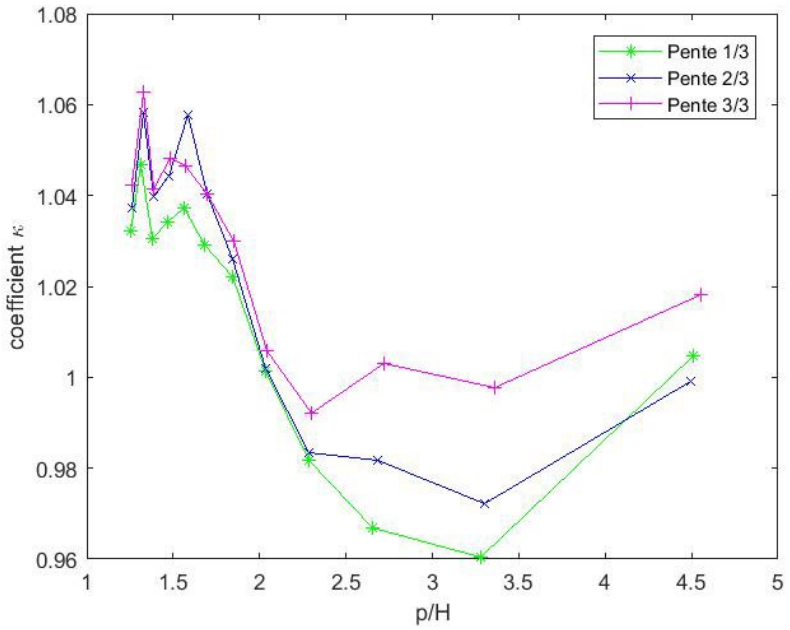

FIGURE 25 – Influence de la pente à l'amont sur le coefficient de débit, fonction du rapport hauteur de pelle sur hauteur d'eau.

4.3 Ressauts noyés

4.3.1 Niveau d'eau aval imposé

Je me suis ensuite intéressé dans cette section à la simulation de ressaut hydraulique dans la seconde partie du canal. La meilleure solution que j'ai trouvée est d'imposer une vitesse de sortie du flux. En effet, on ne peut pas simplement imposer un niveau d'eau dans OpenFOAM. Cependant le débit est connu car imposé à l'entrée. En imposant la vitesse de sortie, indirectement on choisit donc la hauteur d'eau. Pour que cette solution fonctionne, je prend l'hypothèse des filets fluides parallèles (les couches de fluides sont supposées s'écouler parallèlement les unes aux autres). Il faut donc que le canal soit suffisamment long afin de permettre une certaine stabilisation du flux entre le déversoir et la sortie. De ce fait l'écoulement étudié sur le déversoir ne sera pas influencé par des effets de bords.

Comme on observe un ressaut hydraulique, il a également fallut imposer dans la résolution que les valeurs d'alpha (pour rappel, $\alpha = 1$ pour l'eau et $\alpha = 0 pour l'air$) ne sortent pas de l'intervalle [0 1]. voir E.1

Dans cette partie, je vais étudier l'évolution du coefficient ξ en fonction de la fonction $\frac{h_d}{H}$ comme visible sur le graphe 14.

J'impose un débit entrant de 1 $[m^3/s]$. Nous avons vu dans le cas du déversoir de Creager à paroi verticale que ça équivaut à une hauteur d'eau de H=0.648 [m] (voir tableau 1). Sur le graphe général 13 de l'influence d'un ressaut hydraulique sur l'écoulement d'un déversoir, je vais vérifier les valeurs que l'on retrouve sur un droite verticale au niveau de $\frac{h_d+h_{ds}}{H} = 3.31$.

4.3.2 Analyse des résultats

On observe dans ce cas ci l'apparition d'un ressaut noyé à l'aval du déversoir. Nous pouvons voir ce ressaut sur la figure suivante 26.

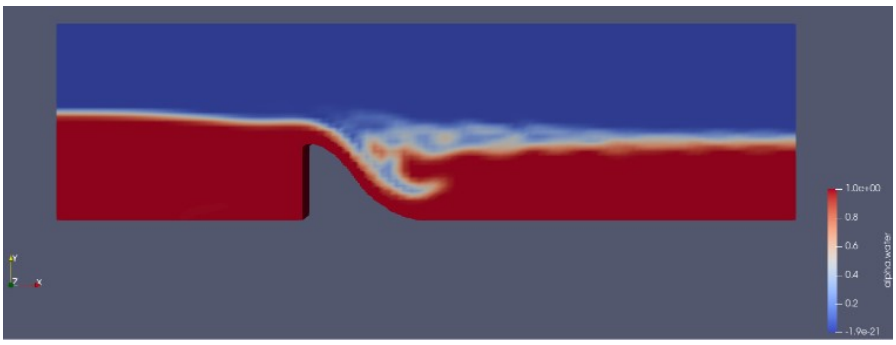

FIGURE 26 – Ressaut hydraulique visualisé grâce à Paraview.

Ce ressaut est une source de turbulence importante. L'écoulement met donc plus de temps que dans les cas précédents pour se stabiliser dans une situation pseudo constante. En effet, d'abord le niveau aval doit se stabiliser, puis ensuite son influence va se ressentir peu à peu à l'amont. J'ai donc allongé les temps de simulation de 60 à 140 secondes de simulation. Nous pouvons voir les courbes atteindre un plateau à ce moment là sur les figures 27 et 28

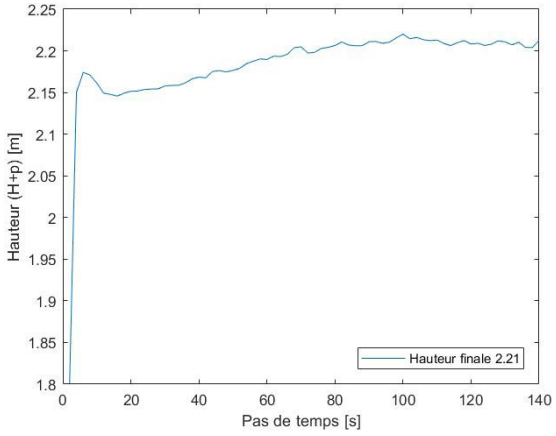

FIGURE 27 – Courbe de l'évolution de la hauteur d'eau à l'amont du déversoir.

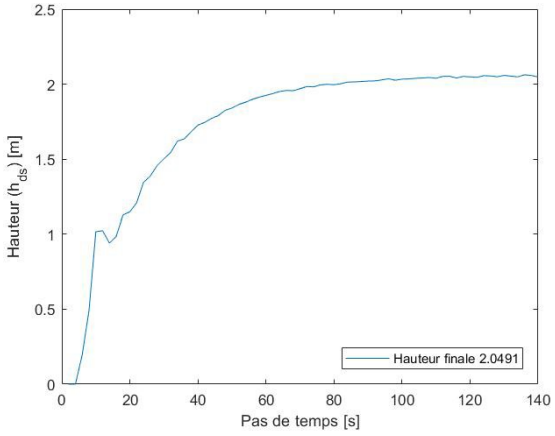

FIGURE 28 – Courbe de l'évolution de la hauteur d'eau à l'aval du déversoir.

Afin de tester la théorie, il m'a fallut décider d'un certain nombre de hauteur d'eau aval à numériser. Ensuite, j'ai déduit les vitesses nécessaires à imposer à la sortie du canal pour visualiser les résultats.

h_d/H	1	0.6	0.4	0.2	0.1	0
V_{out}	0.607	0.5411	0.5133	0.4883	0.4766	0.4

TABLE 4 – Valeurs de vitesses imposées à l'aval en fonction de la hauteur d'eau voulue.

La constante que j'impose dans la simulation est le débit. Je vais donc développer le tableau 5 avec les valeurs expérimentales.

h_d/H	0.8281	0.5290	0.4038	0.2915	0.266	0
$\mu_{\text{Influence Aval}}$	0.418	0.404	0.392	0.380	0.377	0.260
ξ	0.967	0.935	0.908	0.879	0.873	0.603

TABLE 5 – Evolution du coefficient de débit pour un déversoir avec différentes hauteurs de ressaut.

Pour rappel, $\xi = \frac{\mu_{Ressault}}{\mu_{\text{sortie libre}}}$

Ces résultats peuvent être mis en graphe afin de le comparer au théorique visible au graphe 14.

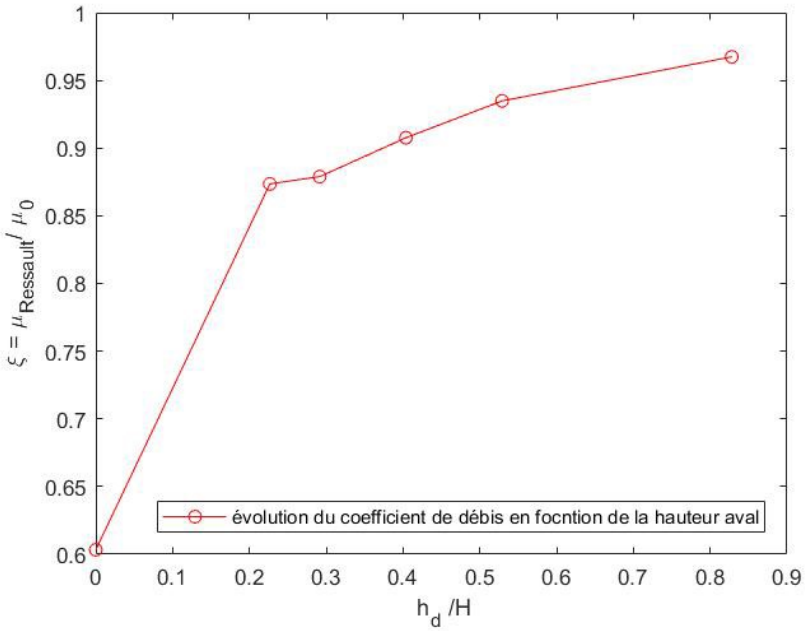

FIGURE 29 – Évolution de ξ en fonction de la hauteur à l'aval du déversoir.

Pour les valeurs de h_d/H relativement importante, nous obtenons d'assez bon résultats. Cependant, pour la dernière valeur, lorsque $h_d/H = 0$ la valeur de ξ expérimentale ne tend pas vers 0. En effet, comme j'impose un débit à l'entrée du système, le canal va continuer d'opérer même lorsque le déversoir est complètement noyé. Il y a donc une limite à ce modèle. On ne peut pas simuler la sortie d'un plan d'eau pour lequel le débit serait nul dans le cas d'une inondation de la voie d'évacuation.

5 Conclusion

Le but de ce mémoire est de vérifier les relations hauteur-débits utilisées lors du dimensionnement des déversoirs, et ce à l'aide de simulations informatiques. Plus particulièrement, on étudie l'influence que peut avoir une inclinaison de la face amont du déversoir sur l'écoulement d'une part et d'autre part on étudie l'influence d'un ressaut hydraulique à l'aval de la structure.

Tout d'abord, la théorie des déversoir est récapitulée. Les règles de design d'un déversoir de Creager et les règles d'hydraulique déterminant l'écoulement sur la structure sont redéveloppées. On explicite aussi les influences amont et aval sur le coefficient de débit, coefficient qui permet de lier la hauteur d'eau à l'amont du déversoir au débit de l'écoulement.

Ensuite, le programme OpenFOAM est étudié. La recherche a été concentrée sur les équations résolues dans le cas d'un écoulement libre ainsi que sur la méthode utilisée pour créer un modèle numérique d'un canal comportant un déversoir.

Les premières simulations ont permis d'analyser plus en détails le cas d'un déversoir de Creager classique, afin de vérifier la viabilité du projet. Ce premier pas a permis de comprendre le fonctionnement du programme, et des opérations de post-traitements à réaliser. Les simulations numériques montrent des résultats ayant un coefficient hydraulique moins élevé que la théorie. Le modèle surestime donc la hauteur d'eau pour un débit donné.

Ensuite l'influence de la pente de la face amont du déversoir a été analysée. L'importance de cette dernière est démontrée lors d'écoulements plus rapides, pour lesquels la hauteur d'eau au dessus du déversoir est importante vis à vis de la profondeur de pelle en amont de celui-ci. On y montre aussi que les résultats du modèle numérique de comparaisons adimensionnelles entre la configuration générale et les configurations modifiées, convergent avec ceux du modèle analytique, confirmant leur pertinence.

Enfin, le cas du ressaut hydraulique à l'aval du déversoir est développé. Tout d'abord la méthode permettant de forcer un ressaut hydraulique dans le canal de simulation est expliquée. Ensuite les résultats sont analysés et à nouveau, une fois adimensionnalisé par rapport au cas général, la simulation donne des résultats convergeant avec la théorie des déversoirs.

L'objectif de confirmer l'utilisation des relations théorique pour estimer l'influence amont est complété. Pour la seconde partie, concernant l'influence d'un ressaut en aval du déversoir, ce mémoire montre que l'utilisation du logiciel est pertinente et les résultants préliminaires semblent concluant. Les modèles numériques développés dans ce mémoire permettent de soutenir les relations retrouvées dans la documentation sur les déversoirs, basés sur des études in situ et en laboratoire.

Pour la suite, il serait intéressant d'approfondir le cas général afin de se rapprocher des valeurs que l'on observe in situ afin d'obtenir facilement des résultats exploitables lors du dimensionnent de nouveaux déversoirs. De plus il serait intéressant de reprendre la méthode pour le ressaut aval afin d'en faire une étude plus complète.

6 Bibliographie

Références

[1] AirShaper. *Meshing Advanced Course*. URL : https://airshaper.com/assets/courses/3_meshing.pdf.

[2] Arnau Bayon et P. López-Jiménez. "Numerical analysis of hydraulic jumps using OpenFOAM". In : *Journal of Hydroinformatics* 17 (mar. 2015), p. 662-678. DOI : 10.2166/hydro.2015041.

[3] D. Cred et Guha-Sapir. *EM-DAT : The Emergency Events Database* -. Université catholique de Louvain(UCLouvain) - Brussels, Belgium.

[4] GIEC. *Climate Change 2021 : The Physical Science Basis*. 2021-2022. URL : https://www.ipcc.ch/report/ar6/wg1/#FullReport.

[5] Christopher J. Greenshields. *OpenFOAM The Open Source CFD Toolbox Programmer's Guide*. OpenFOAM Foundation Ltd, 2015.

[6] F. Jonard. *LBRES2204 – Gestion Intégrée des Ressources en Eau*. (Syllabus) Université Catholique de Louvain-la-Neuve, 2008.

[7] Bureau of Reclamation. *Design of small dams*. United States Département of the interior, 2015.

[8] Olivier ROELANDT et Thibaut VERSCHOORE. *Numerical modeling of a flow on a deteriorated weir : application to the Dory weir, Cavaillon River, Haiti*. Université Catholique de Louvain-la-Neuve, 2015-2016.

[9] S. Soares-Frazão. *LGCIV2051 – Applied Hydraulics : open-channel flows*. (Syllabus) Université Catholique de Louvain-la-Neuve, 2019-2020.

[10] *Unofficial OpenFoam wiki*. URL : https://openfoamwiki.net/index.php/Main_Page.

[11] Y. Zech. *AUCE2151 – Hydraulique Appliquée*. (Syllabus) Université Catholique de Louvain-la-Neuve, 2020-2021.

7 Annexes

A Appendix 1 : Script de récupération des données OpenFOAM et post-traitement pour mesurer la hauteur d'eau

```
% hauteur d'eau amont
%
% Auteurs : Thibault Pr vost.
% NOMA(s)   :0525 1500.
% Date     : 06/05/2021

clc;
clear all;
close all;

k=0;
Moyenne=0;

for j=1:20

%r cup ration des donn es de post traitements
CheminDonnees=strcat(fullfile('C:\Users\memoire\WEIR0\weir0i\postProcessing\...
linesample',num2str(2*j),'data_alpha.water.xy'));
Donnees=load(CheminDonnees);

y=Donnees(:,2);
alpha=Donnees(:,4);

y2=Donnees(:,2);
alphaWater2=Donnees(:,4);

%calcul de l'int gral pour connaitre la hauteur de l'eau
Integral=sum(alphaWater2);
Hauteur2(j)=y2(round(Integral))+ ...
    (Integral-round(Integral))*(y2(round(Integral))-y2(round(Integral-1)));

%je prend la hauteur moyenne une fois l'  coulement   stabilis  car il ...
    y a
%toujours de petites turbulences
if j>14 && j≤20
    %     if Hauteur2(j)-Hauteur2(j-1)<0.01
    Moyenne=Moyenne+Hauteur2(j)
    %     end
    k=k+1;
end
end

Hprint2=Moyenne/k;

figure
plot(Hauteur2)
legend(['Hauteur finale ', ...
    num2str(Hprint2),'[m]'],'Location','southeast')
```

B Appendix 2 :Script de calcul de la relation hauteur débit et des nouveaux coefficient de débits

Ce script est utilisé dans le cas du déversoir classique et des analyse pour l'influence de la pente à l'amont du déversoir.

```
% Relation HQ
%
% Auteurs : Thibault Pr vost.
% NOMA(s)   :0525 1500.
% Date    : 06/05/2021

clc;
clear all;
%close all;

DebitOpenfoam=[0.25 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3];
HauteurOpenFoam=[1.8335 1.9457 2.0529 2.1484 2.2379 2.3272 2.4081 ...
    2.4826 2.5445 2.6085 2.6745 2.7215];

%Dimensionnement
H0=0.5;
p=3*H0;

%valeurs th oriques
mu0=0.496;
%creager y/H0 = 0.47(x/H0)^(1.8)
L=1;
g=9.81;

Q0=mu0*L*sqrt(2*g)*H0^(3/2);

Hauteur=0.2:0.1:1.7;
Debit=length(Hauteur);

mu=mu0.*(Hauteur./H0).^0.17;

for i=1:length(Hauteur)
Debit(i)= mu(i) * L * sqrt(2*g) * Hauteur(i)^(3/2);
end

Hauteur=Hauteur+p;


%% Calcul valeurs exp rimentales de mu

for i=1:length(DebitOpenfoam)

    muExpe(i)= DebitOpenfoam(i)/(L * sqrt(2*g) * ...
       (HauteurOpenFoam(i)-p)^(3/2));

end

%calcul du mu0 exp riementale
mu0Expe=0;

for i=2:length(DebitOpenfoam)
```

```
50      if HauteurOpenFoam(i)>2 && mu0Expe==0
51          Δ...
                =(2-HauteurOpenFoam(i-1))/(HauteurOpenFoam(i)-HauteurOpenFoam(i-1));
52          DebitDimensionnement=DebitOpenfoam(i-1)+Δ*(DebitOpenfoam(i)-DebitOpenfoam(i
53          mu0Expe = DebitDimensionnement/(L * sqrt(2*g) * (2-p)^(3/2))
54      else
55      end
56  end
57
58  Q0exp=mu0Expe*L*sqrt(2*g)*H0^(3/2);
59
60  %Calcul de la courbe avec la formule th orique (mu/mu0) = (H/H0)^0.17
61  muexpExtrapol=mu0Expe.*((HauteurOpenFoam-p)./0.5).^0.17;
62
63  %Calcul du nouvel exposant pour la formule th orique
64  expo= (log(muExpe./mu0Expe)./log((HauteurOpenFoam-p)./H0));
65  expoMoyen=mean(expo)
66
67  Exposigma=std(expo);
68
69  minExp=expoMoyen-2*Exposigma;
70  maxExp=expoMoyen+2*Exposigma;
71
72  %calcul de la nouvelle courbe de \mu
73  NouveauMu=mu0Expe.*((Hauteur-p)./H0).^expoMoyen;
74
75  figure
76  plot(DebitOpenfoam,HauteurOpenFoam-p,'ro')
77  hold on
78  plot(Debit,Hauteur-p)
79  xlabel('Debit [m^3 /s]')
80  ylabel('Hauteur H [m]')
81  legend('Relation Hauteur-D bit calcul e avec le programme ...
        OpenFoam','Relation Hauteur-D bit th orique','Location','southeast')
82  hold off
83
84
85  figure
86  plot((HauteurOpenFoam-p)./H0,muExpe./mu0Expe,'ro')
87  hold on
88  plot((Hauteur-p)./H0,mu./mu0)
89  hold on
90  plot((Hauteur-p)./H0,NouveauMu./mu0Expe)
91  ylabel('\mu/\mu_0')
92  xlabel('Hauteur H/H_0')
93  legend('Valeurs de \mu_{exp rimentale} calcul  avec le programme ...
        OpenFoam','Relation \mu-Hauteur th orique','Relation ...
        \mu_{exp rimentale}-Hauteur d duite des ...
        simulations','Location','southeast')
94  hold off
```

C Appendix 3 :Script pour comparer l'influence des différentes pentes amont

```
% Relation HQ
%
% Auteurs : Thibault Pr vost.
% NOMA(s)   :0525 1500.
% Date    : 06/05/2021

clc;
clear all;
%close all;

DebitOpenfoam=[1 1 1 1 1 1];
HauteurOpenFoam=[2.1633 2.1786 2.1921 2.2071 2.21 2.4087];
HauteurAval= [1.614 1.8196 1.9126 2.001 2.0491 2.4087];

%Dimensionnement
H0=0.5;
p=3*H0;

%valeurs th oriques
mu0=0.496;
%creager y/H0 = 0.47(x/H0)^(1.8)
L=1;
g=9.81;


%% Calcul valeurs exp rimentales de mu

for i=1:length(DebitOpenfoam)

muExpe(i)= DebitOpenfoam(i)/(L * sqrt(2*g) * ...
    (HauteurOpenFoam(i)-p)^(3/2));

end

%calcul du mu0 exp riementale
mu0Expe=0;

for i=2:length(DebitOpenfoam)
    if HauteurOpenFoam(i)>2 && mu0Expe==0
        △...
            =(2-HauteurOpenFoam(i-1))/(HauteurOpenFoam(i)-HauteurOpenFoam(i-1));
        DebitDimensionnement=DebitOpenfoam(i-1)+△*(DebitOpenfoam(i)-DebitOpenfoam(i
        mu0Expe = DebitDimensionnement/(L * sqrt(2*g) * (2-p)^(3/2))
    else
    end
end

%mu calcul  pour le cas de base (sans influence aval)

muCreager = 0.432;

Abscisses = (HauteurOpenFoam-HauteurAval)./(HauteurOpenFoam-p)
Ordonnees = muExpe ./muCreager
```

```
52
53
54 figure
55
56 plot(Abscisses,Ordonnees,'ro-')
57 xlabel(' h_{d} /H')
58 ylabel('\xi = \mu_{Ressault}/ \mu_0')
59 legend('  volution  du coefficient de d bis en focntion de la hauteur ...
       aval','Location','southeast')
60 hold off
```

D Appendix 4 :Script pour analyser l'influence du ressaut hydraulique sur le coefficient de débis

```
% Relation HQ
%
% Auteurs : Thibault Pr vost.
% NOMA(s)   :0525 1500.
% Date    : 06/05/2021

clc;
clear all;
%close all;

DebitOpenfoam=[1 1 1 1 1 1];
HauteurOpenFoam=[2.1633 2.1786 2.1921 2.2071 2.21 2.4087];
HauteurAval= [1.614 1.8196 1.9126 2.001 2.0491 2.4087];

%Dimensionnement
H0=0.5;
p=3*H0;

%valeurs th oriques
mu0=0.496;
%creager y/H0 = 0.47(x/H0)^(1.8)
L=1;
g=9.81;


%% Calcul valeurs exp rimentales de mu

for i=1:length(DebitOpenfoam)

muExpe(i)= DebitOpenfoam(i)/(L * sqrt(2*g) * ...
    (HauteurOpenFoam(i)-p)^(3/2));

end

%calcul du mu0 exp riementale
mu0Expe=0;

for i=2:length(DebitOpenfoam)
    if HauteurOpenFoam(i)>2 && mu0Expe==0
        ...
            =(2-HauteurOpenFoam(i-1))/(HauteurOpenFoam(i)-HauteurOpenFoam(i-1));
        DebitDimensionnement=DebitOpenfoam(i-1)+*(DebitOpenfoam(i)-DebitOpenfoam(i
        mu0Expe = DebitDimensionnement/(L * sqrt(2*g) * (2-p)^(3/2))
    else
    end
end

%mu calcul  pour le cas de base (sans influence aval)

muCreager = 0.432;

Abscisses = (HauteurOpenFoam-HauteurAval)./(HauteurOpenFoam-p)
Ordonnees = muExpe ./muCreager
```

```
52
53
54 figure
55
56 plot(Abscisses,Ordonnees,'ro-')
57 xlabel(' h_{d} /H')
58 ylabel('\xi = \mu_{Ressault}/ \mu_0')
59 legend('  volution  du coefficient de d bis en focntion de la hauteur ...
       aval','Location','southeast')
60 hold off
```

E Appendix 5 : OpenFOAM

Les différents documents dans cette annexe commence tous avec une entête comme présentée ci dessous. Par soucis de clarité, elle sera omise.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      epsilon;
}
```

E.1 fichier "system"

blockMeshDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1;

vertices        //list of vertices defining the geometry
(
    (-5 0 0)        //This is point 0. Not 1!
    (10 0 0)        //This is point 1
    (10 4 0)        //point 2
    (-5 4 0)        //point 3
    (-5 0 1)        //point 4
    (10 0 1)        //point 5
    (10 4 1)        //point 6
    (-5 4 1)        //point 7
);

blocks          //defining the block which the geometry is made out of
(
    hex (0 1 2 3 4 5 6 7) (100 40 1) simpleGrading (1 1 1)
//this means that the only block in the geometry is defined by ...
   vertices (0 1 2 ...), there are 100, 40
//and 1 cells in the x, y, and z direction and that in each direction ...
   all the cells have the same length
);

edges
```

```
(
);

patches
    // patches are used to allocate the different Boundary Conditions
(
    patch inlet
    (
        (3 7 4 0)
    )
    patch outlet
    (
        (2 1 5 6)
    )
    wall lowerWall
    (
        (0 4 5 1)
    )

    patch atmosphere
    (
        (2 6 7 3)
    )
);

mergePatchPairs
(
);

// ************************************************************************* //
```

controlDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         40;

ΔT          0.005;

writeControl    adjustableRunTime;

writeInterval   2;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression off;

```

```
timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  on;

maxCo           0.2;

maxAlphaCo      0.2;

maxDeltaT       1;

writeIntervalProbes 100;


functions
{
    linesample //mesure the fields between the two points defined in ...
       the sets over Time (used for convergence analysis)
    //For hydraulics jump analysis, I use a second linesample downstream
    {

        type            sets;
        libs            ("libsampling.so");
        writeControl        adjustableRunTime;
        writeInterval       2;

        interpolationScheme cellPoint;

        setFormat       raw;

        sets
        (
            data
            {
                type    uniform;
                axis    xyz;
                start   (-4.5 0 0.5);
                end (-4.5 4 0.5);
                nPoints 100;
            }
        );

        fields          (alpha.water);

        includeOutOfBounds  true;
    }
}
// ************************************************************************* //
```

fvSchemes

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         localEuler; //localEuler allows faster ...
```

```
        computation but need stable flow. For transient flow (like ...
        donwstream hydraulic jump) I need to use Euler wich define the ...
        time step on the whole simulation instead of locally.
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)    Gauss linear;
    div(phi,alpha)   Gauss vanLeer; //When studying transient flow, I ...
        had to use limitedVanLeer 0 1. It forces the value of alpha to ...
        stay in the interval. Otherwise with lots of turbulences ...
        discontinuity would appear
    div(phirb,alpha) Gauss linear;
    div(phi,k)       Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)       Gauss upwind;
    div(R)           Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

// ************************************************************************* //
```

fvSolution

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    alpha.water
    {
        nAlphaCorr      1;
        nAlphaSubCycles 2;
        cAlpha          1;
    }

    "pcorr.*"
    {
        solver          PCG;
```

```
        preconditioner  DIC;
        tolerance       1e-10;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|epsilon)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
    }

    "(U|k|epsilon)Final"
    {
        $U;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor no;
    nCorrectors     5;
    nNonOrthogonalCcorrectors 2;
}


// ************************************************************************* //
```

meshQualityDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//- Maximum non-orthogonality allowed. Set to 180 to disable.
maxNonOrtho 65;

//- Max skewness allowed. Set to <0 to disable.
maxBoundarySkewness 20;
maxInternalSkewness 4;

//- Max concaveness allowed. Is angle (in degrees) below which concavity
//  is allowed. 0 is straight face, <0 would be convex face.
//  Set to 180 to disable.
maxConcave 80;
```

```
14
15 //- Minimum pyramid volume. Is absolute volume of cell pyramid.
16 //  Set to a sensible fraction of the smallest cell volume expected.
17 //  Set to very negative number (e.g. -1E30) to disable.
18 minVol 1e-13;
19
20 //- Minimum quality of the tet formed by the face-centre
21 //  and variable base point minimum decomposition triangles and
22 //  the cell centre. Set to very negative number (e.g. -1E30) to
23 //  disable.
24 //     <0 = inside out tet,
25 //      0 = flat tet
26 //      1 = regular tet
27 minTetQuality 1e-15;
28
29 //- Minimum face area. Set to <0 to disable.
30 minArea -1;
31
32 //- Minimum face twist. Set to <-1 to disable. dot product of face normal
33 // and face centre triangles normal
34 minTwist 0.02;
35
36 //- Minimum normalised cell determinant. This is the determinant of all
37 //  the areas of internal faces. It is a measure of how much of the
38 //  outside area of the cell is to other cells. The idea is that if all
39 //  outside faces of the cell are 'floating' (zeroGradient) the
40 //  'fixedness' of the cell is determined by the area of the internal ...
      faces.
41 //  1 = hex, ≤ 0 = folded or flattened illegal cell
42 minDeterminant 0.001;
43
44 //- Relative position of face in relation to cell centres (0.5 for ...
      orthogonal
45 //  mesh) (0 -> 0.5)
46 minFaceWeight 0.05;
47
48 //- Volume ratio of neighbouring cells (0 -> 1)
49 minVolRatio 0.01;
50
51 //- Per triangle normal compared to average normal. Like face twist
52 //  but now per (face-centre decomposition) triangle. Must be >0 for ...
      Fluent
53 //  compatibility
54 minTriangleTwist -1;
55
56
57 //- If >0 : preserve cells with all points on the surface if the
58 //  resulting volume after snapping (by approximation) is larger than
59 //  minVolCollapseRatio times old volume (i.e. not collapsed to flat ...
      cell).
60 //  If <0 : delete always.
61 //minVolCollapseRatio 0.1;
62
63
64 // ************************************************************************* //
```

setFieldsDict

```
1 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
2
3  defaultFieldValues
4  (
5      volScalarFieldValue alpha.water 0
6  );
7
8  regions
9  (
10     boxToCell
11     {
12         box (-5 0 0) (0.5 1.5 1);
13
14         fieldValues
15         (
16             volScalarFieldValue alpha.water 1
17         );
18     }
19 );
20
21 // ************************************************************************* //
```

snappyHexMeshDict

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3  // Which of the steps to run
4  castellatedMesh true;
5  snap            true;
6  addLayers       false;
7
8
9  // Geometry. Definition of all surfaces. All surfaces are of class
10 // searchableSurface.
11 // Surfaces are used
12 // - to specify refinement for any mesh cell intersecting it
13 // - to specify refinement for any mesh cell inside/outside/near
14 // - to 'snap' the mesh boundary to the surface
15 geometry
16 {
17     weir.stl
18     {
19         type triSurfaceMesh;
20         name weir;
21     }
22
23 };
24
25
26
27 // Settings for the castellatedMesh generation.
28 castellatedMeshControls
29 {
30
31     // Refinement parameters
32     // ~~~~~~~~~~~~~~~~~~~~~
33
34     // If local number of cells is >= maxLocalCells on any processor
35     // switches from from refinement followed by balancing
36     // (current method) to (weighted) balancing before refinement.
```

```
    maxLocalCells 100000;

    // Overall cell limit (approximately). Refinement will stop ...
       immediately
    // upon reaching this number so a refinement level might not ...
       complete.
    // Note that this is the number of cells before removing the part ...
       which
    // is not 'visible' from the keepPoint. The final number of cells ...
       might
    // actually be a lot less.
    maxGlobalCells 2000000;

    // The surface refinement loop might spend lots of iterations ...
       refining just a
    // few cells. This setting will cause refinement to stop if ≤ ...
       minimumRefine
    // are selected for refinement. Note: it will at least do one ...
       iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 2;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 2;



    // Explicit feature edge refinement
    // ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
        {
            file "weir.eMesh";
            level 1;
        }
    );



    // Surface based refinement
    // ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

    // Specifies two levels for every surface. The first is the ...
       minimum level,
    // every cell intersecting a surface gets refined up to the ...
       minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
        weir
        {
```

```
                // Surface-wise min and max refinement level
                level (1 2);
            }
        }

        resolveFeatureAngle 30;


        // Region-wise refinement
        // ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

        // Specifies refinement level for cells in relation to a surface. ...
           One of
        // three modes
        // - distance. 'levels' specifies per distance to the surface the
        //   wanted refinement level. The distances need to be specified in
        //   descending order.
        // - inside. 'levels' is only one entry and only the level is ...
           used. All
        //   cells inside the surface get refined up to the level. The ...
           surface
        //   needs to be closed for this to be possible.
        // - outside. Same but cells outside.

        refinementRegions
        {
            weir
            {
                mode inside;
                levels ((1 2));
            }
        }


        // Mesh selection
        // ¬¬¬¬¬¬¬¬¬¬¬¬¬¬

        // After refinement patches get added for all refinementSurfaces and
        // all cells intersecting the surfaces get put into these ...
           patches. The
        // section reachable from the locationInMesh is kept.
        // NOTE: This point should never be on a face, always inside a ...
           cell, even
        // after refinement.
        // This is an outside point locationInMesh (-0.033 -0.033 0.0033);
        locationInMesh (-1 1 0); // Inside point

        // Whether any faceZones (as specified in the refinementSurfaces)
        // are only on the boundary of corresponding cellZones or also allow
        // free-standing zone faces. Not used if there are no faceZones.
        allowFreeStandingZoneFaces true;
    }



    // Settings for the snapping.
    snapControls
    {
        //- Number of patch smoothing iterations before finding ...
```

```
        correspondence
141     //  to surface
142     nSmoothPatch 3;
143
144     //- Relative distance for points to be attracted by surface ...
        feature point
145     //  or edge. True distance is this factor times local
146     //  maximum edge length.
147     tolerance 4;
148
149     //- Number of mesh displacement relaxation iterations.
150     nSolveIter 30;
151
152     //- Maximum number of snapping relaxation iterations. Should stop
153     //  before upon reaching a correct mesh.
154     nRelaxIter 5;
155
156     // Feature snapping
157
158         //- Number of feature edge snapping iterations.
159         //  Leave out altogether to disable.
160         nFeatureSnapIter 5;
161
162         //- Detect (geometric) features by sampling the surface
163         implicitFeatureSnap true;
164
165         //- Use castellatedMeshControls::features
166         explicitFeatureSnap false;
167
168         //- Detect features between multiple surfaces
169         //  (only for explicitFeatureSnap, default = false)
170         multiRegionFeatureSnap false;
171 }
172
173
174
175 // Settings for the layer addition.
176 addLayersControls
177 {
178     // Are the thickness parameters below relative to the undistorted
179     // size of the refined cell outside layer (true) or absolute ...
        sizes (false).
180     relativeSizes true;
181
182     // Per final patch (so not geometry!) the layer information
183     layers
184     {
185         "weir_*"
186         {
187             nSurfaceLayers 2;
188         }
189     }
190
191     // Expansion factor for layer mesh
192     expansionRatio 1.0;
193
194
195     // Wanted thickness of final added cell layer. If multiple layers
196     // is the thickness of the layer furthest away from the wall.
```

```
197     // See relativeSizes parameter.
198     finalLayerThickness 0.9;
199 
200     // Minimum thickness of cell layer. If for any reason layer
201     // cannot be above minThickness do not add layer.
202     // See relativeSizes parameter.
203     minThickness 0.05;
204 
205     // If points get not extruded do nGrow layers of connected faces ...
            that are
206     // also not grown. This helps convergence of the layer addition ...
            process
207     // close to features.
208     nGrow 0;
209 
210 
211     // Advanced settings
212 
213     // When not to extrude surface. 0 is flat surface, 90 is when two ...
            faces
214     // are perpendicular
215     featureAngle 30;
216 
217     // Maximum number of snapping relaxation iterations. Should stop
218     // before upon reaching a correct mesh.
219     nRelaxIter 5;
220 
221     // Number of smoothing iterations of surface normals
222     nSmoothSurfaceNormals 1;
223 
224     // Number of smoothing iterations of interior mesh movement direction
225     nSmoothNormals 3;
226 
227     // Smooth layer thickness over surface patches
228     nSmoothThickness 2;
229 
230     // Stop layer growth on highly warped cells
231     maxFaceThicknessRatio 0.5;
232 
233     // Reduce layer growth where ratio thickness to medial
234     // distance is large
235     maxThicknessToMedialRatio 0.3;
236 
237     // Angle used to pick up medial axis points
238     minMedianAxisAngle 90;
239 
240     // Create buffer region for new layer terminations
241     nBufferCellsNoExtrude 0;
242 
243 
244     // Overall max number of layer addition iterations. The mesher ...
            will exit
245     // if it reaches this number of iterations; possibly with an illegal
246     // mesh.
247     nLayerIter 30;
248 
249     // Max number of iterations after which relaxed meshQuality controls
250     // get used. Up to nRelaxIter it uses the settings in ...
            meshQualityControls,
```

```
251     // after nRelaxIter it uses the values in ...
            meshQualityControls::relaxed.
252     nRelaxedIter 10;
253 }
254
255
256
257 // Generic mesh quality settings. At any undoable phase these determine
258 // where to undo.
259 meshQualityControls
260 {
261     //- Maximum non-orthogonality allowed. Set to 180 to disable.
262     maxNonOrtho 65;
263
264     //- Max skewness allowed. Set to <0 to disable.
265     maxBoundarySkewness 20;
266     maxInternalSkewness 4;
267
268     //- Max concaveness allowed. Is angle (in degrees) below which ...
            concavity
269     //  is allowed. 0 is straight face, <0 would be convex face.
270     //  Set to 180 to disable.
271     maxConcave 80;
272
273     //- Minimum pyramid volume. Is absolute volume of cell pyramid.
274     //  Set to a sensible fraction of the smallest cell volume expected.
275     //  Set to very negative number (e.g. -1E30) to disable.
276     minVol 1e-13;
277
278     //- Minimum quality of the tet formed by the face-centre
279     //  and variable base point minimum decomposition triangles and
280     //  the cell centre.  Set to very negative number (e.g. -1E30) to
281     //  disable.
282     //     <0 = inside out tet,
283     //      0 = flat tet
284     //      1 = regular tet
285     minTetQuality 1e-9;
286
287     //- Minimum face area. Set to <0 to disable.
288     minArea -1;
289
290     //- Minimum face twist. Set to <-1 to disable. dot product of ...
            face normal
291     //- and face centre triangles normal
292     minTwist 0.05;
293
294     //- minimum normalised cell determinant
295     //- 1 = hex, <= 0 = folded or flattened illegal cell
296     minDeterminant 0.001;
297
298     //- minFaceWeight (0 -> 0.5)
299     minFaceWeight 0.05;
300
301     //- minVolRatio (0 -> 1)
302     minVolRatio 0.01;
303
304     //must be >0 for Fluent compatibility
305     minTriangleTwist -1;
306
```

```
307     //- if >0 : preserve single cells with all points on the surface ...
            if the
308     //  resulting volume after snapping (by approximation) is larger than
309     //  minVolCollapseRatio times old volume (i.e. not collapsed to ...
            flat cell).
310     //  If <0 : delete always.
311     //minVolCollapseRatio 0.5;
312
313
314     // Advanced
315
316     //- Number of error distribution iterations
317     nSmoothScale 4;
318     //- amount to scale back displacement at error points
319     errorReduction 0.75;
320
321
322
323     // Optional : some meshing phases allow usage of relaxed rules.
324     // See e.g. addLayersControls::nRelaxedIter.
325     relaxed
326     {
327         //- Maximum non-orthogonality allowed. Set to 180 to disable.
328         maxNonOrtho 75;
329     }
330 }
331
332
333 // Advanced
334
335 // Flags for optional output
336 // 0 : only write final meshes
337 // 1 : write intermediate meshes
338 // 2 : write volScalarField with cellLevel for postprocessing
339 // 4 : write current intersections as .obj files
340 debug 0;
341
342
343 // Merge tolerance. Is fraction of overall bounding box of initial mesh.
344 // Note: the write tolerance needs to be higher than this.
345 mergeTolerance 1E-6;
346
347
348 // ************************************************************************* //
```

streamlines

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3  Description
4      Writes out files of streamlines with interpolated field data in ...
           VTK format.
5
6  \*---------------------------------------------------------------------------*/
7
8  nLines  10;
9  start   (-0.0205 0.001  0.00001);
10 end     (-0.0205 0.0251 0.00001);
11 fields  (p k U);
```

```
12
13 // Must be last entry
14 #includeEtc "caseDicts/postProcessing/visualization/streamlines.cfg"
15
16 // ************************************************************************* //
```

surfaceFeatureExtractDict

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3  weir.stl
4  {
5      // How to obtain raw features (extractFromFile || extractFromSurface)
6      extractionMethod    extractFromSurface;
7
8      extractFromSurfaceCoeffs
9      {
10         // Mark edges whose adjacent surface normals are at an angle less
11         // than includedAngle as features
12         // - 0  : selects no edges
13         // - 180: selects all edges
14         includedAngle   150;
15     }
16
17     // Write options
18
19         // Write features to obj format for postprocessing
20         writeObj                yes;
21 }
22
23
24 // ************************************************************************* //
```

E.2 fichier "constant"

g

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3  dimensions      [0 1 -2 0 0 0 0];
4  value           (0 -9.81 0);
5
6  // ************************************************************************* //
```

transportProperties

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ...
       * //
2
3  phases (water air);
4
5  water
6  {
7      transportModel  Newtonian;
8      nu              [0 2 -1 0 0 0 0] 1e-06;
```

```
    rho             [1 -3 0 0 0 0 0] 1000;
}

air
{
    transportModel  Newtonian;
    nu              [0 2 -1 0 0 0 0] 1.48e-05;
    rho             [1 -3 0 0 0 0 0] 1;
}

sigma           [1 0 -2 0 0 0 0] 0.07;


// ************************************************************************* //
```

turbulenceProperties

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RAS;

RAS
{
    RASModel        kEpsilon;

    turbulence      on;

    printCoeffs     on;
}


// ************************************************************************* //
```

E.3 fichier "0"

alpha.Water

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ...
    * * //

#include        "include/initialConditions"

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            variableHeightFlowRate;
        lowerBound      0;
        upperBound      1;
        value           uniform 0;
    }
```

```
    // adapt the waterHeight at the inlet so the inflow can be split ...
       on the whole water area

    outlet
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }
    weir
    {
        type            zeroGradient;
    }


    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
}

// ...
    ************************************************************************* ...
    //
```

epsilon

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform $turbulentEpsilon;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }
```

```
23
24      lowerWall
25      {
26          type            epsilonWallFunction;
27          value           $internalField;
28      }
29
30      atmosphere
31      {
32          type            inletOutlet;
33          inletValue      $internalField;
34          value           $internalField;
35      }
36
37      defaultFaces
38      {
39          type            empty;
40      }
41      weir
42      {
43          type            epsilonWallFunction;
44          value           $internalField;
45      }
46  }
47
48
49  // ************************************************************************* //
```

k

```
1   // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3   #include        "include/initialConditions"
4
5   dimensions      [0 2 -2 0 0 0 0];
6
7   internalField   uniform $turbulentKE;
8
9   boundaryField
10  {
11      inlet
12      {
13          type            fixedValue;
14          value           $internalField;
15      }
16
17      outlet
18      {
19          type            inletOutlet;
20          inletValue      $internalField;
21          value           $internalField;
22      }
23
24      lowerWall
25      {
26          type            kqRWallFunction;
27          value           $internalField;
28      }
29
```

```
    atmosphere
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    defaultFaces
    {
        type            empty;
    }
    weir
    {
        type            kqRWallFunction;
        value           $internalField;
    }


}


// ************************************************************************* //
```

nut

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    lowerWall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    atmosphere
    {
        type            calculated;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
    weir
```

```
35      {
36          type            nutkWallFunction;
37          value           uniform 0;
38      }
39
40  }
41
42
43  // ************************************************************************* //
```

p_rgh

```
1   // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3   #include        "include/initialConditions"
4
5   dimensions      [1 -1 -2 0 0 0 0];
6
7   internalField   uniform $pressure;
8
9   boundaryField
10  {
11      inlet
12      {
13          type            zeroGradient;
14      }
15
16      outlet
17      {
18          type            zeroGradient;
19      }
20
21      lowerWall
22      {
23          type            zeroGradient;
24      }
25
26      atmosphere
27      {
28          type            totalPressure;
29          p0              uniform 0;
30          U       U;
31          phi     phi;
32          gamma       1;
33          value       uniform $pressure;
34      }
35
36      defaultFaces
37      {
38          type            empty;
39      }
40      weir
41      {
42          type            zeroGradient;
43      }
44  }
45
46  // ************************************************************************* //
```

U

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            variableHeightFlowRateInletVelocity;
        flowRate        $inletFlowRate;
        alpha           alpha.water;
        value           uniform (0 0 0);
    }
    // adapt the velocity at the inlet so the inflow can be split on ...
       the whole water area


    outlet
    {
        type            zeroGradient;
    }
    //For hydraulic jup, I set the velocity on the outlet
    //the velocity is computed to impose a certain waterlevel h ...
       (width of the canal is 1 so A=h)
    // V=Q/h with Q the outflow
    //
    //outlet
    //{
    //    type            fixedValue;
    //    value      uniform (3 0 0);
    //}
    //

    lowerWall
    {
        type            fixedValue;
        value        uniform (0 0 0);
    }

    atmosphere
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }

    defaultFaces
    {
        type            empty;
    }
    weir
    {
        type            fixedValue;
```

```
55          value           uniform (0 0 0);
56      }
57  }
58
59  // ************************************************************************* //
```

initialConditions

```
1  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
2
3  inletFlowRate           1;
4  pressure                0;
5  turbulentKE             4.14e-03;
6  turbulentEpsilon        4.39e-05;
7  #inputMode       merge
8
9  // ************************************************************************* //
```

F Appendix 6 : Illustrations des zones pour les différentes conditions de bord

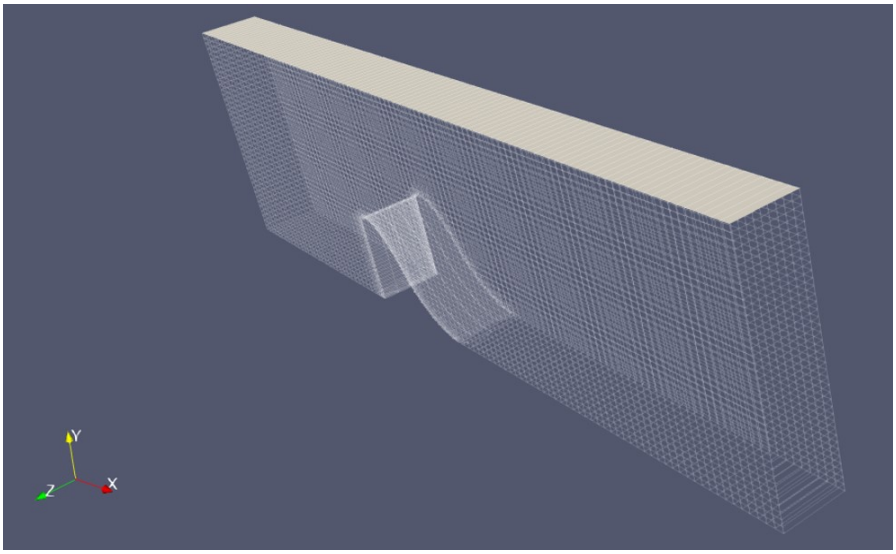

FIGURE 30 – Conditions de bord : atmosphère

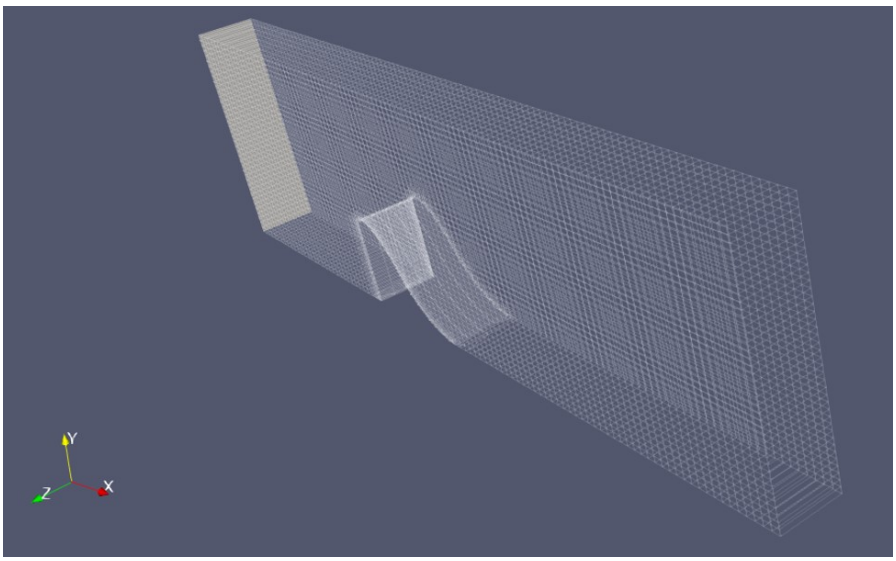

FIGURE 31 – Conditions de bord : atmosphère

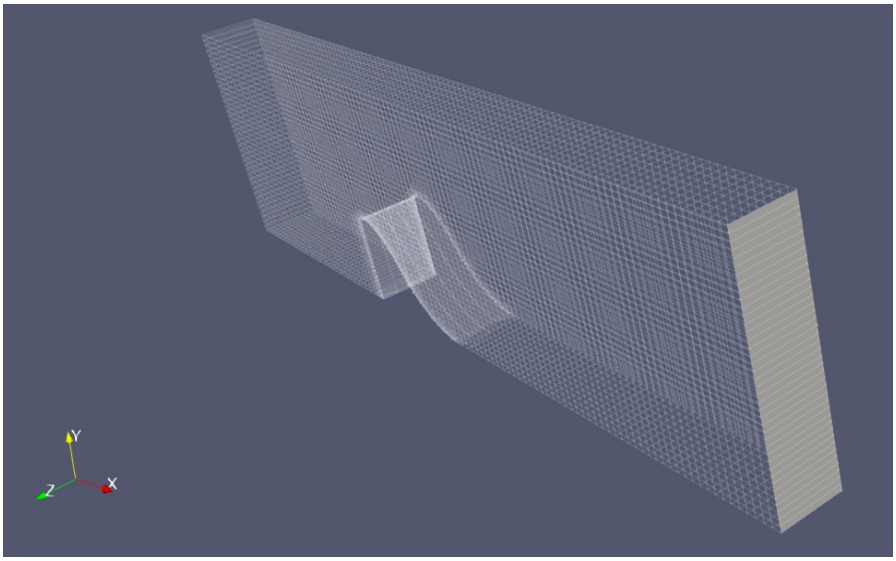

FIGURE 32 – Conditions de bord : atmosphère

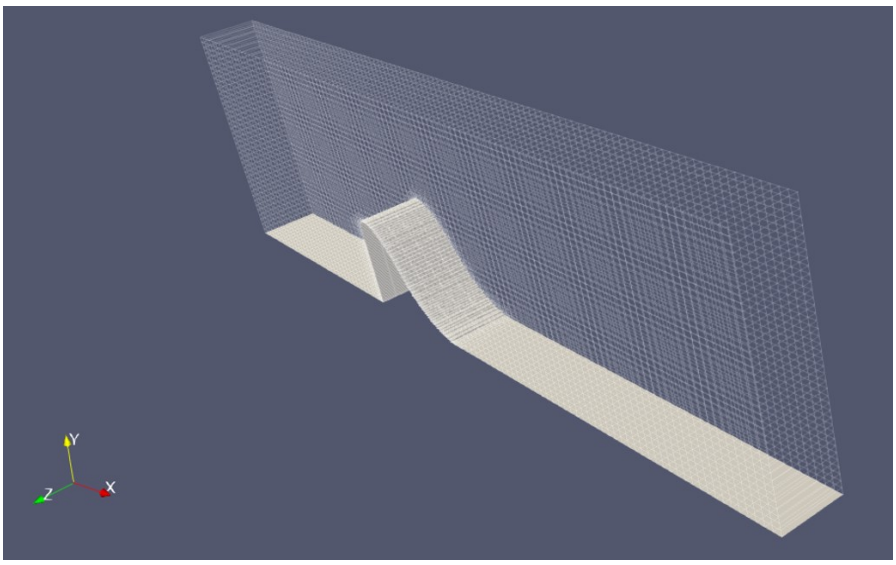

FIGURE 33 – Conditions de bord : atmosphère

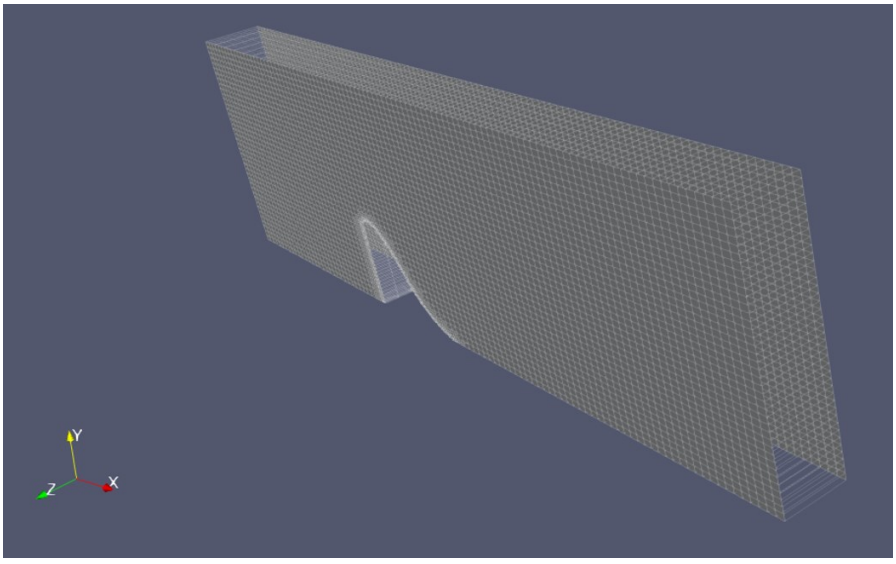

FIGURE 34 – Conditions de bord : atmosphère

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**