A finite point method to solve shallow water equations

C. Buachart
E. Ortega
E. Oñate

Publication CIMNE Nº-337, February 2009

A finite point method to solve shallow water equations

C. Buachart
E. Ortega
E. Oñate

Publication CIMNE Nº-337, February 2009

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

Table of Contents

1. INTRODUCTION

The Finite Point Method (FPM) proposed by Oñate et al.[1] is a truly meshfree method, which has a great capability to handle fluid flow[2] and solid mechanics[3] problems. The numerical analysis process for this meshfree scheme consists of generating first a set of points, named global cloud of points or nodes, within problem domain. Then, for each of these points, a local cloud of neighboring points is selected. Local approximations of unknowns are performed using weighted least square procedure. Finally, the algebraic equations are obtained by substituting above local approximations directly into the governing partial differential equations of the problem (strong form).

Due to the great capability to solve the fluid flow problem, in this present work we will try to solve the shallow water equations[4], which are degenerated from tri-dimensional Navier-Strokes equations by integrating them in gravity direction. Similar with other fluid flow problems, the solution procedure need some stabilization. In this work, we use the approximate Riemann solver, which is most suitable to work with the strong form of governing equations[5]. Also, to preserve the high accuracy of FPM approximation, the iterative scheme to generate the local cloud and local approximation proposed by Ortega et al.[6], are used to construct FPM approximation.

The report is organized as follows: Section 2 introduce the basic weighted least square (WLS) approximation for finite point method. Section 3 explains the iterative procedure to compute the shape functions. Section 4 describes the construction of local clouds, and Section 5 the shallow water equations and the flow solver. Numerical examples are shown in Section 6, and some conclusions are drawn in Section 7.

[1] Oñate, E., Idelsohn, S. Zienkiewicz, O. C. Taylor, R. L. and Sacco, C. (1996). A finite point method for analysis of fluid mechanics problems. Applications to convective transport and fluid flow, *International Journal for Numerical Methods in Engineering*, 39: 3839-3866.

[2] Löhner, R., Sacco, C., Onate, E. and Idelsohn, S. (2002). A finite point method for compressible flow, *International Journal for Numerical Methods in Engineering,* 53: 1765-1779.

[3] Oñate, E., Perazzo, F. and Miquel, J. (2001). A finite point method for elasticity problems, *Computer and Structures*, 79: 2151-2163.

[4] Zienkiewicz, O. C. and Taylor, R. L. (2000). *The Finite Element Method*, vol III: *Fluid Dynamics*, 5th edition, Butterworth-Heinemann.

[5] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors and difference schemes, *Journal of Computational Physics*, 43: 357-372.

[6] Ortega, E., Oñate, E. and Idelsohn, S. (2007). An improved finite point method for tri-dimensional potential flows, *Computational Mechanics*, 40: 949-963.

2. WEIGHTED LEAST SQUARES APPROXIMATIONS

In this section, we recall the basic weighted least squares procedure. An approximation of unknown function $u(\mathbf{x})$ defined in a closed domain $\Omega \in \mathbb{R}^d$, where d = 1, 2 or 3, which is discretized to be a set of points $\mathbf{x}_i$, $i = 1, 2, ..., n$. In order to obtain a local approximation of $u(\mathbf{x})$, the domain is divided into local subdomains Ω_i so that $\Omega \subseteq \bigcup_{i=1}^{n} \Omega_i$. Assume that Ω_i contains np points (clouds of points). Each cloud of points consists of a point $\mathbf{x}_i$ called *star point* and a set of points $\mathbf{x}_j$, $j = 1, 2, ..., np-1$, surrounding star point $\mathbf{x}_i$. The unknown function $u(\mathbf{x})$ may be approximated within Ω_i by

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x}) = \sum_{\zeta=1}^{m} p_\zeta(\mathbf{x}) \alpha_\zeta = \mathbf{p}(\mathbf{x})^{\mathrm{T}} \cdot \boldsymbol{\alpha} \tag{2.1}$$

where $\boldsymbol{\alpha}^{\mathrm{T}} = \begin{bmatrix} \alpha_1 & \alpha_2 & ... & \alpha_m \end{bmatrix}$ is a vector must be determined, and the vector $\mathbf{p}(\mathbf{x})$ contains the so-called basis functions, which are typically monomials. For 2D problems we have used:

- Quadratic basis (*m* = 6)

$$\mathbf{p}(\mathbf{x})^{\mathrm{T}} = \begin{bmatrix} 1 & \xi & \eta & \xi^2 & \xi\eta & \eta^2 \end{bmatrix} \tag{2.2}$$

- Cubic basis (*m* = 10)

$$\mathbf{p}(\mathbf{x})^{\mathrm{T}} = \begin{bmatrix} 1 & \xi & \eta & \xi^2 & \xi\eta & \eta^2 & \xi^3 & \xi^2\eta & \xi\eta^2 & \eta^3 \end{bmatrix} \tag{2.3}$$

where

$$\xi = \frac{x - x_i}{d_{\max}}, \quad \eta = \frac{y - y_i}{d_{\max}} \tag{2.4}$$

and $d_{\max} = \max\left(\left\| \mathbf{x}_j - \mathbf{x}_i \right\|\right)$ is the distance between the star point and the furthest point in the cloud.

Throughout this report, indices *i* and *j* are used to represent the star points, and cloud of points, respectively. Hence, the range are $i = 1, 2, ..., n$ and $j = 1, 2, ..., np$. On the other hand, indices *k* and *l* are ranged from 1 to *d*. The Einstein summation convention will be employed, i.e., a sum is always performed over repeated indices, except for an index *i*. Define the notation $u_j^h = u(\mathbf{x}_j), \hat{u}_j = \hat{u}(\mathbf{x}_j), \mathbf{p}_j = \mathbf{p}(\mathbf{x}_j)$ and $\varphi_{ij} = \varphi_i(\mathbf{x}_j)$, the weighted least squares approximation (WLSQ) within Ω_i is obtained by minimizing

$$J_i = \varphi_{ij} \left(\hat{u}_j - u_j^h \right)^2 = \varphi_{ij} \left(u_j^h - \mathbf{p}_j^{\mathrm{T}} \cdot \boldsymbol{\alpha} \right)^2 \tag{2.4}$$

The quality of WLSQ approximation depends on the shape of the weighting function. In FPM, a fixed weighting function $\varphi_i(\mathbf{x})$, centered on the star point $\mathbf{x}_i$ of the cloud is chosen so that it satisfies the following conditions

$$\begin{aligned} &\varphi_i(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \Omega_i \\ &\varphi_i(\mathbf{x}) = 0,\ \forall \mathbf{x} \notin \Omega_i \\ &\varphi_i(\mathbf{x}_i) = 1 \end{aligned} \tag{2.5}$$

This kind of approximation is called Fixed Least Squares Method (FLS) and can be considered as a particular case of the Moving Least Squares Method (MLS) proposed by Lancaster and Salkauskas[7]. It should be noticed that the approximation functions obtained are discontinuous and this fact imposes certain restrictions on the local approximation.

We remark that $np \geq m$ is always required. Moreover, for $np = m$ the effect of weighting function vanishes, and the procedure reverts to interpolation. The minimization of J_i with respect to $\boldsymbol{\alpha}$ yields

$$\mathbf{A} \cdot \boldsymbol{\alpha} = \mathbf{B} \cdot \mathbf{u}^h \tag{2.6}$$

i.e.

$$\boldsymbol{\alpha} = \mathbf{C} \cdot \mathbf{u}^h,\ \mathbf{C} = \mathbf{A}^{-1} \cdot \mathbf{B} \tag{2.7}$$

where

$$\mathbf{A} = \sum_{j=1}^{np} \varphi_{ij} \left(\mathbf{p}_j \otimes \mathbf{p}_j^{\mathrm{T}} \right) \tag{2.8}$$

$$\mathbf{B} = \left[\varphi_{i1}\mathbf{p}_1 \quad \varphi_{i2}\mathbf{p}_2 \quad \cdots \quad \varphi_{inp}\mathbf{p}_{np} \right] \tag{2.9}$$

Replacing (2.7) into (2.1), the approximation to the unknown function is obtained for $\mathbf{x} = \tilde{\mathbf{x}}$, thus

$$\hat{u}(\tilde{\mathbf{x}}) = \mathbf{p}^{\mathrm{T}}(\tilde{\mathbf{x}}) \cdot \mathbf{C} \cdot \mathbf{u}^h = \mathbf{N}(\tilde{\mathbf{x}}) \cdot \mathbf{u}^h \tag{2.10}$$

where $\mathbf{N}(\tilde{\mathbf{x}}) := \mathbf{p}^{\mathrm{T}}(\tilde{\mathbf{x}}) \cdot \mathbf{C}$, is the shape function (row) vector of the point $\tilde{\mathbf{x}}$ in Ω_i.

The adoption of an FLS scheme, where matrices **A** and **B** are constant in Ω_i, of course matrix **C**, noticeably simplifies the calculation of shape functions derivatives. Consequently, for any *s*-order of derivatives,

$$\frac{\partial^s \mathbf{N}(\tilde{\mathbf{x}})}{\partial x_k^s} = \frac{\partial^s \mathbf{p}^{\mathrm{T}}(\tilde{\mathbf{x}})}{\partial x_k^s} \cdot \mathbf{C} \tag{2.11}$$

and the unknown function derivatives are calculated as

[7] Lancaster, P. and Salkauskas, K. (1981). Surfaces generated by moving least squares methods, *Mathematics and. Computation*, 37: 141-158.

$$\frac{\partial^s \hat{u}(\tilde{\mathbf{x}})}{\partial x_k^s} = \frac{\partial^s \mathbf{N}(\tilde{\mathbf{x}})}{\partial x_k^s} \cdot \mathbf{u}^h = \frac{\partial^s \mathbf{p}^{\mathrm{T}}(\tilde{\mathbf{x}})}{\partial x_k^s} \cdot \mathbf{C} \cdot \mathbf{u}^h \tag{2.12}$$

If the point $\tilde{\mathbf{x}} \to \mathbf{x}_i$, any value or derivative can be simplified in the form

$$\hat{u}(\mathbf{x}_i) = C_i^{1j} u_j^h = N^{ij} u_j^h \tag{2.13}$$

$$\frac{\partial \hat{u}(\mathbf{x}_i)}{\partial x_k} = D_k^{ij} u_j^h,\ D_k^{ij} := \frac{1}{d_{\max}} C_i^{qj} \text{ and } q = k+1 \tag{2.14}$$

$$\nabla^2 \hat{u}(\mathbf{x}_i) = L^{ij} u_j^h,\ L^{ij} := \frac{2}{d_{\max}^2}\left(C_i^{4j} + C_i^{6j}\right) \tag{2.15}$$

2.1. Consistency of the approximation

The term of consistency is a useful tool to describe the ability of numerical approximation to reproduce a given polynomial of order *p* and its derivatives in an exact way. It says that any shape functions $\mathbf{N}(\mathbf{x})$ has *p*-order consistency if they are satisfied

$$\frac{\partial^s N^j(\mathbf{x})}{\partial x_k^q \partial x_l^r} \mathbf{p}_j = \frac{\partial^s \mathbf{p}(\mathbf{x})}{\partial x_k^q \partial x_l^r},\ q + r = s \text{ and } s = 0,1,...,p;\ \forall \mathbf{x} \in \Omega \tag{2.16}$$

where $\mathbf{p}(\mathbf{x})$ is a complete polynomial basis of order *p*. In usual MLS approximation, the consistency should be satisfied for all point in the domain. However, due to the fact that the shape function and their derivatives are discontinuous in FLS scheme, it is only possible to satisfy the consistency requirement (2.16) in the star point $\mathbf{x}_i$, where the weighting function is located.

2.2. The weighting function

There exist many possibilities for choosing the functional form of the weighting function that satisfies the conditions given in (2.5). In present scheme, a normalized Gaussian function is chosen and defined by

$$\varphi_i(\mathbf{x}) = \frac{\exp\left[-\left(wd/\beta\right)^2\right] - \exp\left(-w^2\right)}{1 - \exp\left(-w^2\right)} \tag{2.17}$$

where $d = \left\|\mathbf{x} - \mathbf{x}_i\right\|$ and $\beta = \gamma d_{\max}$ $(\gamma > 1.0)$. The support of this function is isotropic, circular in 2D and spherical in 3D. The parameters w and γ govern the shape of the weighting function and the quality of approximation. Hence, these free parameters should be properly set. Here, we briefly present the schemes to set the free parameters from the work proposed by Ortega et al.[8,9].

Parameter γ provides more or less weight to the boundary points of the cloud by increasing or decreasing the size of the weighting function's support. Increasing of γ provide a bigger of the overlapping zone between neighboring clouds of points. These effects allow user to improving the approximation quality where highly distorted clouds of points happen in some regions of problem domain. In this case, good results are obtained by setting $1 < \gamma < 1.25$.

Finally, the very importance parameter is the parameter *w*. This parameter is introduced to locally adjust the shape of weighting function as shown in Figure 1. For large values of *w*, the shape of weighting function tends to the Dirac's delta function. The shape function $N^j(\mathbf{x})$, also tends to the Dirac's delta function. When the value of *w* is increased, the approximation procedure tends to interpolate nodal data. This causes the error in the approximation to decrease and the condition number of matrix **A**, $\kappa(\mathbf{A})$, to increase, and the problem becomes more and more ill-conditioned. If the condition number go beyond the threshold value, i.e. $\kappa(\mathbf{A}) > \kappa_{\max}(\mathbf{A})$, it will impossible to invert matrix **A** with accuracy. The approximation quality deteriorates quickly and numerical instabilities appear.

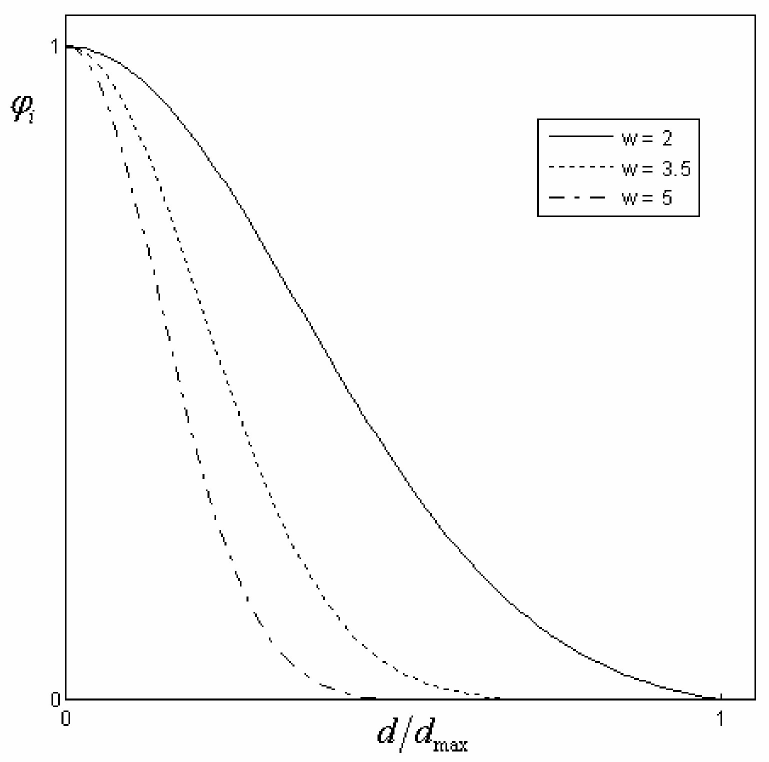

Figure 1. Effects of the parameter *w* on the weighting function shape, $\gamma = 1.01$.

[8] Ibid. 6.

[9] Ortega, E., Oñate, E. and Idelsohn, S. A finite point method for adaptive three-dimensional compressible flow calculations, In press, *International Journal for Numerical Methods in Fluid.*

The numerical experiment results from 3D problem by Ortega et al.[10] are used to be the guidance to set a range of maximum value for parameter *w*. An admissible range for maximum value are given by $3.0 \le w_{\max} \le 4.5$. In this work, a value is set to be $w_{\max} = 3.5$ in the whole domain, and then it is reduced for each cloud of points whenever necessary. This point will be discussed later in the next section.

2.3. Discretization of equations

In the FLS method the weighting function is fixed at each star point of the cloud and leads to multi-value shape functions at the same point $\tilde{\mathbf{x}}$ in the whole domain, depending on which cloud is used to approximate the unknown function at that given point. Hence, the approximation is considered to be valid only at the star point of each cloud. Consequently, a collocation method becomes a suitable choice in FPM.

Although the concept of collation procedure is simple, they usually suffer from numerical instability of the global equation system. However, this numerical instability can be suppressed by introducing the stabilization term to the system of equation. This can be done either direct[11, 12] or indirect[13] ways.

[10] Ibid. 6.

[11] Oñate, E. (1998). Derivation of stabilized equations for numerical solution of advective-diffusive transport and fluid flow problems, *Computer Methods in Applied Mechanics and Engineering*, 151: 233-265.

[12] Boroomand, B., Tabatabaei, A. A. and Oñate, E. (2005). Simple modifications for stabilization of the finite point method, *International Journal for Numerical Methods in Engineering*, 63: 351-379.

[13] Xiaozhong, J., Gang, L. and Aluru, N. R. (2004). Positivity conditions in meshless collocation methods, *Computer Methods in Applied Mechanics and Engineering*, 193: 1171-1202.

3. COMPUTATION OF THE SHAPE FUNCTIONS PARAMETERS

According to WLSQ approximation in previous section, in order to compute the shape function **N** and their derivatives for a given cloud of points, the unknown coefficients $\boldsymbol{\alpha}$ have to be solved formerly from the following linear system

$$\mathbf{A} \cdot \boldsymbol{\alpha} = \mathbf{B} \cdot \mathbf{u}^h \tag{3.1}$$

With the matrix **A** and **B** are defined by equations (2.8) and (2.9), respectively. Due to the fact that **u** is not known in advance, this system has to be solved via inversion. The solution of the equations (3.1) by direct inversion of matrix **A** must be restricted to cases when the condition number $\kappa(\mathbf{A})$ is moderate. In general, when the condition number $\kappa(\mathbf{A})$ is large, its inverse is not appropriate to compute the shape functions and their derivatives.

In this work, the procedure to calculate the shape function and their derivatives is taken from Ortega et al.[14] and can be described following. Given a certain cloud of points, first, the inversion of matrix **A** is computed. If the condition number $\kappa(\mathbf{A})$ is smaller than a given maximum admissible value the shape functions are calculated. Then, if the calculated shape functions satisfy some quality tests, they are accepted. If some of the proceeding requirements are not met, the normal equations (3.1) are solved by an alternative procedure based on QR factorization.

3.1. Solution of the normal equations via QR factorization

In general, QR factorization is a more stable and accurate method for solving least squares problems when the matrix **A** is ill-conditioned. An acceptable solution is expected from this procedure in case where the other procedures fail without having to modify the geometrical structure of the cloud. However, note that the cost of least square solution via QR factorization is twice as much as the solution via matrix **A** inversion if $np \gg m$ [15]. The definition of QR factorization can be briefly described below.

If any matrix $\mathbf{P} \in \mathbb{R}^{\mathrm{np}\times\mathrm{m}}$ has rank *m* and $np > m$, then it can be uniquely factored as

$$\mathbf{P} = \mathbf{Q} \cdot \mathbf{R} \tag{3.2}$$

where $\mathbf{Q} \in \mathbb{R}^{\mathrm{np}\times\mathrm{m}}$ is an orthogonal matrix, i.e., $\mathbf{Q}^{\mathrm{T}} \cdot \mathbf{Q} = \mathbf{I}$ and $\mathbf{R} \in \mathbb{R}^{\mathrm{m}\times\mathrm{m}}$ is upper triangular with positive diagonal entries $R_{ii} > 0$. If a matrix **P** has rank deficient, column pivoting can be applied.

In order to solve the system (3.1) by QR factorization, it is necessary to obtain another equivalent form, which the weight disappears from the equation. This can be done by defined,

[14] Ibid. 6.

[15] Demmel, J. W. (1997). *Applied numerical linear algebra*, Society for Industrial and Applied Mathematics.

$$\tilde{\mathbf{\Phi}} := diag\left(\sqrt{\varphi_{i1}} \quad \sqrt{\varphi_{i2}} \quad \cdots \quad \sqrt{\varphi_{inp}}\right) \tag{3.3}$$

and the modification of matrix **B**

$$\tilde{\mathbf{B}} = \left[\mathbf{p}_1\sqrt{\varphi_{i1}} \quad \mathbf{p}_2\sqrt{\varphi_{i2}} \quad \cdots \quad \mathbf{p}_{np}\sqrt{\varphi_{inp}}\right] \tag{3.4}$$

Recall to equations (2.8) and (2.9), matrices **A** and **B** can be rewritten into the forms,

$$\mathbf{A} = \tilde{\mathbf{B}} \cdot \tilde{\mathbf{B}}^{\mathrm{T}} \tag{3.5}$$

$$\mathbf{B} = \tilde{\mathbf{B}} \cdot \tilde{\mathbf{\Phi}} \tag{3.6}$$

Hence, equation (3.1) is recast into the new form, i.e.

$$\left(\tilde{\mathbf{B}} \cdot \tilde{\mathbf{B}}^{\mathrm{T}}\right) \cdot \boldsymbol{\alpha} = \left(\tilde{\mathbf{B}} \cdot \tilde{\mathbf{\Phi}}\right) \cdot \mathbf{u} \tag{3.7}$$

Then, the transpose of modified matrix (3.4) is factorized, i.e., $\tilde{\mathbf{B}}^{\mathrm{T}} = \mathbf{Q} \cdot \mathbf{R}$, substitute into (3.1), and use the orthogonal property of matrix **Q**, yields

$$\mathbf{R} \cdot \boldsymbol{\alpha} = \left(\mathbf{Q}^{\mathrm{T}} \cdot \tilde{\mathbf{\Phi}}\right) \cdot \mathbf{u}^h \tag{3.8}$$

The unknown vector $\boldsymbol{\alpha}$ can be finally obtained by

$$\boldsymbol{\alpha} = \mathbf{R}^{-1} \cdot \left(\mathbf{Q}^{\mathrm{T}} \cdot \tilde{\mathbf{\Phi}}\right) \cdot \mathbf{u}^h \tag{3.9}$$

Here, matrix **R** is generally well-conditioned and due to it is upper triangular, inversion is easy to obtain with accuracy. The well-conditioning of matrix **R** can be expected, even for the cases when the condition number of matrix **A** is large. This fact reduces the approximation's dependence on the spatial distribution of points and on the shape of weighting function significantly.

3.2. An iterative procedure for calculating the shape functions

To obtain high-order approximation with a given cloud of points, and attempt to avoid cloud regeneration, the following iterative procedure will be used to calculate the shape functions. This scheme is proposed by Ortega et al.[16]

First, the initial value of weighting function parameter w is set to be equal to maximum admissible value, i.e., $w_{\mathrm{int}} = w_{\max}$, and the WLSQ problem (3.1) is solved via matrix **A** inversion. Then, the shape function and their derivatives are computed by (2.10) and (2.11), respectively. The resulting approximation is accepted if it satisfied the following requirements:

[16] Ibid. 6.

r_1. $\kappa(\mathbf{A}) \le \kappa_{\max}$

r_2. $\left|\sum_j N^j(\mathbf{x}_i) - 1.0\right| \le tol$ and $\left|\sum_j \frac{\partial N^j(\mathbf{x}_i)}{\partial x_k}\right| \le tol$

r_3. Consistency

The first requirement (r_1) imposes a limit to the condition number of matrix **A**, in order to guarantee that its inversion has an acceptable accuracy. The second requirement (r_2) is the well-known *partition of unity* (PU) and *partition of nullities* (PNs) properties, respectively. The last requirement (r_3) measures the ability to reproduce any polynomials, which their order are not greater than the basis polynomial, by checking the consistency requirements (2.16) at the cloud's star point. Note that in the last requirement (r_3), the tolerance parameter *tol* is also used. Generally, the values adopted for setting $\kappa_{\max}$ and *tol* depend on the problem under consideration. In this work a value $\kappa_{\max} = 10^6$ based on the infinite norm and the parameter $tol = 10^{-10}$ are adopted. The consistency check (r_3) is performed according to the guidelines given by Lohner et al.[17].

If any of the preceding requirements is not satisfied, the approximation is rejected, and the solver changes to the QR factorization based methodology (3.9), by keeping all approximation parameters constant. The above requirements are checked again, but the first requirement (r_1) is changed to be,

r_1'. $\kappa(\mathbf{R}) \le \kappa_{\max}$.

If the requirements r_1', r_2 and r_3 are not satisfied, the parameter *w* is reduced by 25 per cent, then equation (3.9) is recalculated. This procedure is repeated until all requirements are satisfied or parameter *w* reaches a minimum admissible value $w_{\min}$. Numerical experiments[18] have shown that two or three iterations are enough.

Finally, if a local cloud of points does not allow, i.e. the parameter *w* reaches a minimum value, the cloud points are regenerated by increase the size of the clouds. Even though the regeneration of clouds is very time consuming, it is needed only for a few problematic clouds, which in general, represent a small percent of the overall clouds. Consequently, the computational efficiently is not affected in a large extent.

[17] Ibid. 2.

[18] Ibid. 6.

4. GENERATION OF LOCAL CLOUDS

Any WLSQ approximation in FPM requires the construction of a local cloud of points for all star points in the domain. The quality of local approximation depends on the number of points in the cloud and their spatial distribution with respect to the star point.

In this section, the methodology to generate the cloud points is taken from Ortega et al.[19]. Only two geometrical restrictions are concerned to make sure that all cloud points can 'see' the star point when they have a concave boundary pass through the group of close points (relate to the star point). Hence, these two restrictions are designed for every star point in the domain which is located either over a surface or sufficiently close to a surface. Suppose that normal vector of all those points over a surface are known, and the search radius (r_{search}) is set. The restrictions for cloud point's inclusion and local clouds construction are described below.

4.1. Star point located over a surface boundary

In this case (Figure 2), any points which are the candidate of cloud point is accepted if they satisfy the following conditions

$$\cos(\theta) \geq \cos\left(\frac{\pi}{2}+\delta\right) \text{ where } \cos(\theta) := \frac{\mathbf{n}_i^{\mathrm{T}} \cdot \mathbf{r}_j}{\|\mathbf{n}_i\| \cdot \|\mathbf{r}_j\|} \tag{4.1}$$

$$r_j^t := \left\|\mathbf{r}_j - \left(\mathbf{n}_i^{\mathrm{T}} \cdot \mathbf{r}_j\right)\mathbf{n}_i\right\| < \alpha r_{search} \tag{4.2}$$

Condition (4.1) determines an acceptable domain around the star point, which is defined in the normal direction to the surface $\mathbf{n}_i$ at the star point. The parameter δ is a small angle, represent a surface curvature. The second condition (4.2) is a specification of an aspect ratio to the cloud, obtain from parameter $\alpha > 0$.

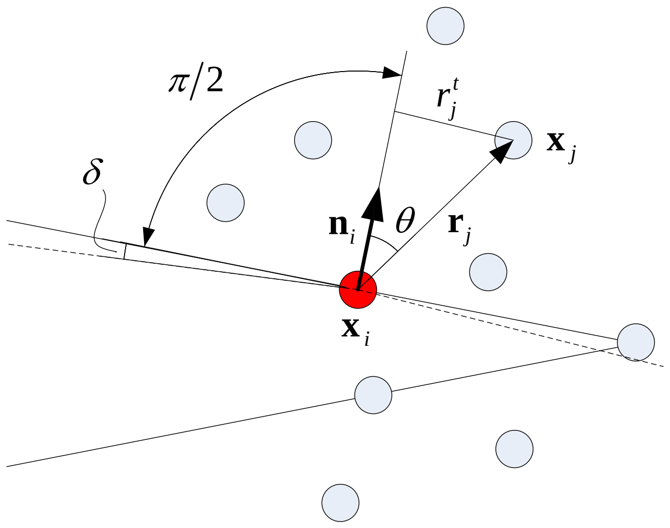

Figure 2. Construction of local clouds when the star point is located over a boundary.

[19] Ibid. 6.

4.2. Cloud of points intercepting a boundary

In this case (Figure 3), the point located over a surface nearest to the star point $\mathbf{x}_i$, namely $\mathbf{x}_{jnea}$, must be sought. Then, any candidate point is accepted to be the cloud if

$$\cos(\theta) \geq \cos\left(\frac{\pi}{2}+\delta\right) \text{ where } \cos(\theta) := \frac{\mathbf{n}_{jnea}^{\mathrm{T}} \cdot \mathbf{r}_j}{\left\|\mathbf{n}_{jnea}\right\| \cdot \left\|\mathbf{r}_j\right\|} \qquad (4.3)$$

and no restriction is imposed to the aspect ratio of the cloud of points.

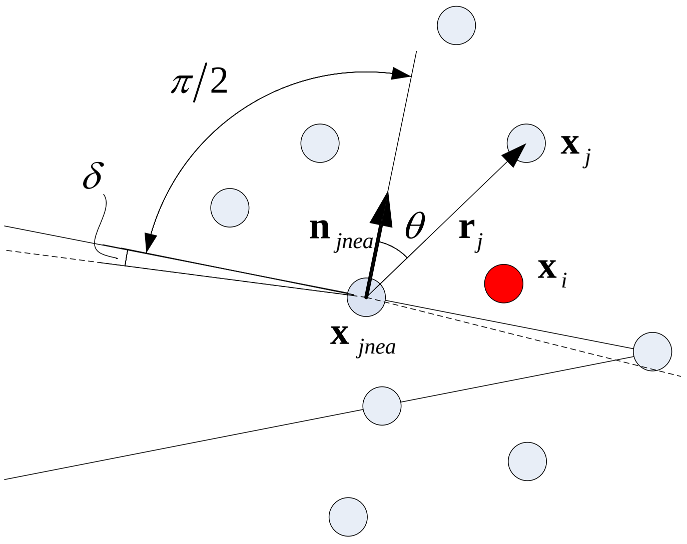

Figure 3. Construction of local clouds when a cloud of points intercepting a boundary.

4.3. Local clouds construction

Given a point discretization of the domain and a set of normal vectors belonging to the edges (i.e., line element in 2D and triangular element in 3D) that bounds this domain, a maximum ($np_{\max}$) and minimum ($np_{\min}$) allowable number of cloud points and an initial search radius (r_{search}) are set. The constructions of cloud for all star points are as follows:

```
do: For each star point ipoin;
    Initialize the search region around ipoin;
    while: not enough close points ($np_{\min} \leq n_c \leq np_{\max}$):
        Enlarge the search region;
        Obtain the points in the search region (i.e., using octree technique);
        Remove, from the close points, those whose not satisfy equations
        (4.1) and (4.2), or (4.3);
    endwhile
    Produce a Delaunay grid with the local points;
    while: order of an approximation is more than one:
        Initialize the local cloud list with the first layer of nearest
        neighbours;
        If the requirements $\mathrm{r}_1, \mathrm{r}_2$ and $\mathrm{r}_3$ are satisfied: exit;
```

```
                do: For all close points, according to increasing distance :
                        Add further points, in groups of 4 for 2D case and 6 for the
                        3D case, to the local cloud;
                        If the requirements $r_1^{'}, r_2$ and $r_3$ are satisfied: exit;
                enddo
                As no proper cloud was found: change to QR solver;
                If requirements $r_1^{'}, r_2$ and $r_3$ are satisfied: exit;
                while: $\left(w > w_{\min}\right)$:
                        Reduce the value of parameter w by 25 per cent;
                        If requirements $r_1^{'}, r_2$ and $r_3$ are satisfied: exit;
                endwhile
                As no proper cloud was found: increase the search region;
        endwhile
enddo
```

Above pseudo code roughly describe the idea of local cloud construction. Note that the requirements $r_1\left(r_1^{'}\right), r_2$ and r_3 are listed in previous section (section 3).

5. SOLVING THE SHALLOW WATER EQUATIONS

The two-dimensional shallow water equations (SWEs) describe flow in shallow water bodies. They can be derived from three-dimension incompressible Navier-Stokes equations by integrate through the vertical direction and assume that the vertical acceleration within the fluid is negligible and the pressure is hydrostatic[20]. The 2D SWEs are written as[21]

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^l}{\partial x_l} - \frac{\partial \mathbf{G}^l}{\partial x_l} = \mathbf{S} \tag{5.1}$$

where $\mathbf{U}$ is the vector of the conservative variables, $\mathbf{F}^1$ and $\mathbf{F}^2$ are the convective fluxes, $\mathbf{G}^1$ and $\mathbf{G}^2$ are the diffusive fluxes, and $\mathbf{S}$ is the vector of source terms. They are given as

$$\begin{gathered}\mathbf{U} = \begin{bmatrix} h \\ hu_1 \\ hu_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix},\ \mathbf{F}^k = \begin{bmatrix} hu_k \\ hu_1 u_k + p\delta_{1k} \\ hu_2 u_k + p\delta_{2k} \end{bmatrix},\ p := \frac{1}{2} g h^2, \\ \mathbf{G}^k = \begin{bmatrix} 0 \\ vhu_{1,k} \\ vhu_{2,k} \end{bmatrix},\ \mathbf{S} = \begin{bmatrix} 0 \\ -gh\left(Sf_1 + S_{o1}\right) + hC_f u_2 \\ -gh\left(Sf_2 + S_{o2}\right) - hC_f u_1 \end{bmatrix}\end{gathered} \tag{5.2}$$

where h is the water depth, u_k is depth-averaged velocities in x_k-direction, $u_{1,k}$ and $u_{2,k}$ are derivatives of the depth-averaged velocity components, g is the acceleration due to gravity, Sf_k represent the bottom friction terms, S_{ok} are the bed slope terms, C_f is the Corolis parameter and v is the kinematic viscosity coefficient. The quantities Sf_k are the slopes of the energy grade lines in x_k-direction, and are determined from the steady-state friction formulae (in SI units)[22]:

$$Sf_k := \frac{n_f^2 u_k \|\mathbf{u}\|}{h^{4/3}} \tag{5.3}$$

where n_f represents Manning's roughness coefficient, and

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2} \tag{5.4}$$

Also, note that the bottom friction S_{ok} may be defined by the undisturbed depth of water, namely $H\left(\mathbf{x}\right)$, viz.

[20] Ibid. 4.

[21] Wang, Ji-Wen and Liu, Ru-Xun (2005). Combined finite volume-finite element method for shallow water equations, *Computers and Fluids*, 34: 1199-1222.

[22] Glaister, P. (1993). Flux difference splitting for open-channel flows. *International Journal for Numerical Methods in Fluids*, 16: 629-654.

$$S_{ok} := -\frac{\partial H}{\partial x_k} \tag{5.5}$$

Here, we complete the SWEs. The discretization scheme of these equations will be discussed below. The conservative variables and fluxes in (5.1) will be approximated by WSLQ in the forms (see equation (2.13)),

$$\begin{aligned}
\mathbf{U}(\mathbf{x}_i) &\approx \hat{\mathbf{U}}(\mathbf{x}_i) = C_i^{1j}\mathbf{U}_j^h = N^{ij}\mathbf{U}_j^h \\
\mathbf{S}(\mathbf{x}_i) &\approx \hat{\mathbf{S}}(\mathbf{x}_i) = N^{ij}\mathbf{S}_j^h \\
\mathbf{F}^k(\mathbf{x}_i) &\approx \hat{\mathbf{F}}^k(\mathbf{x}_i) = N^{ij}\left(\mathbf{F}_j^k\right)^h \\
\mathbf{G}^k(\mathbf{x}_i) &\approx \hat{\mathbf{G}}^k(\mathbf{x}_i) = N^{ij}\left(\mathbf{G}_j^k\right)^h
\end{aligned} \tag{5.6}$$

It is important to note that the nodal parameter, i.e. $(\bullet)^h$, do not coincide with the approximated parameter $(\hat{\bullet})$ because in the WLSQ approximation, the shape function is not satisfy Kronecker delta property. Hence, if the fully explicit scheme time integration is performed to solve for the discretized system of (5.1), we obtain the approximated parameter $\hat{\mathbf{U}}(\mathbf{x}_i)$ at star point, which implies that additional linear system must be solved in order to get the nodal parameter $\mathbf{U}_j^h$ at star point. Fortunately, this equation system has excellent properties and can be solved by a few iterations of a Gauss-Seidel method or another similar scheme[23]. Henceforth, the markers $(\bullet)^h$ and $(\hat{\bullet})$ will be omitted for the sake of simplicity.

5.1. Discretization of convective fluxes

From expressions (2.14) of the WLSQ approximation procedure, we obtain the following expression for the divergence of the convective flux function:

$$\mathcal{F}^i := \frac{\partial \mathbf{F}^l(\mathbf{x}_i)}{\partial x_l} \approx D_l^{ij}\mathbf{F}_j^l \tag{5.6}$$

with D_l^{ij} are defined in equation (2.14). Then, take advantage of the PNs property of the shape function derivatives, viz.

$$D_l^{ii} + D_l^{ij}\,|_{j\neq i} = 0 \rightarrow D_l^{ii} = -D_l^{ij}\,|_{j\neq i} \tag{5.7}$$

where the special linear operator are defined, $D_l^{ij}\,|_{j\neq i}(\bullet) := \sum_{j\neq i} D_l^{ij}(\bullet)$. Substitute (5.7) into equation (5.6), yields

[23] Enrique Ortega (2007). A finite point method for three-dimensional compressible flow, *Thesis Project*, CIMNE, Barcelona.

$$\mathcal{F}^i = D_l^{ij}\,|_{j\neq i}\left(\mathbf{F}_j^l - \mathbf{F}_i^l\right) \tag{5.8}$$

In general, flux in equation (5.8) is unstable and needs to be stabilized. Equivalent form of (5.8) can be constructed by replace $\mathbf{F}_j^l$ with their midpoint value, denoted by $\mathbf{F}_{ij}^l$, and then extrapolate them via multiplying factor of 2, obtained

$$\mathcal{F}^i = 2D_l^{ij}\,|_{j\neq i}\left(\mathbf{F}_{ij}^l - \mathbf{F}_i^l\right) \tag{5.9}$$

The flux $\mathbf{F}_{ij}^l$, may be imagined as the numerical flux vector at the midpoint of the line segment connecting the star point $\mathbf{x}_i$ to another point $\mathbf{x}_j \in \Omega_i$ in Figure 4. To obtain stabilized solution of (5.1), this numerical flux will be redefined and calculated by an approximated Riemann solver[24], which naturally provides the required dissipation for the semi-discrete expression.

5.1.1. Roe solver

The Cartesian components of the midpoint numerical flux, derived by Roe [25] and applied to Riemann solver [26], are obtained by

$$\mathbf{F}_{ij}^k := \frac{1}{2}\left[\left(\mathbf{F}_{\mathrm{L}}^k + \mathbf{F}_{\mathrm{R}}^k\right) - \left|\tilde{A}\left(\mathbf{U}_{\mathrm{L}}, \mathbf{U}_{\mathrm{R}}\right)\right| \cdot \left(\mathbf{U}_{\mathrm{R}} - \mathbf{U}_{\mathrm{L}}\right)\hat{n}^k\right] \tag{5.10}$$

where $\tilde{A}$ is the flux Jacobian matrix of the fluxes along direction $\mathbf{l}_{ji} := \mathbf{x}_j - \mathbf{x}_i$, see in Figure 4, evaluated by Roe's average variables, and $\hat{n}^k$ is a *k*-th component of unit vector $\hat{\mathbf{n}}$ in the direction of vector $\mathbf{l}_{ji}$. Subscripts R and L refer to the right and left stage of midpoint between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the direction $\mathbf{l}_{ji}$, respectively.

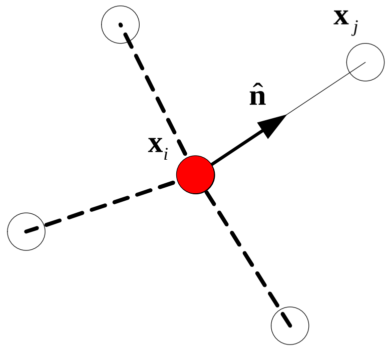

Figure 4. The unit vector along the line-segment in two-dimensional.

[24] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors and difference schemes, *Journal of Computational Physics*, 43: 357-372.

[25] Ibid.

[26] Ibid. 2.

Generally, the flux Jacobian matrix A is given by

$$A = \frac{\partial(\mathbf{F}\cdot\hat{\mathbf{n}})}{\partial\mathbf{U}} = \begin{bmatrix} 0 & \hat{n}^1 & \hat{n}^2 \\ \left(c^2 - u_1^2\right)\hat{n}^1 - u_1 u_2 \hat{n}^2 & 2u_1\hat{n}^1 + u_2\hat{n}^2 & u_1\hat{n}^2 \\ \left(c^2 - u_2^2\right)\hat{n}^2 - u_1 u_2 \hat{n}^1 & u_2\hat{n}^1 & 2u_2\hat{n}^2 + u_1\hat{n}^1 \end{bmatrix} \tag{5.11}$$

where $c = \sqrt{gh}$ is the wave celerity, and note that $\mathbf{F}\cdot\hat{\mathbf{n}} := \mathbf{F}_{ij}^l \hat{n}^l$.

The Roe's average value of u_1, u_2 and c are defined by $\tilde{u}_{ij}^1, \tilde{u}_{ij}^2$ and $\tilde{c}_{ij}$, respectively. These values can be expressed as

$$\begin{aligned} \tilde{u}_{ij}^k &= \frac{u_\mathrm{R}^k\sqrt{h_\mathrm{R}} + u_\mathrm{L}^k\sqrt{h_\mathrm{L}}}{\sqrt{h_\mathrm{R}} + \sqrt{h_\mathrm{L}}} \\ \tilde{c}_{ij} &= \sqrt{\frac{g\left(h_\mathrm{R} + h_\mathrm{L}\right)}{2}} \end{aligned} \tag{5.12}$$

The average Roe's velocities $\tilde{u}_{ij}^k$ can be computed in a more efficient way[27] with the help of an *auxiliary variable* $r := \sqrt{h_\mathrm{R}/h_\mathrm{L}}$. Introducing this variable into first row of equations (5.12), obtain

$$\tilde{u}_{ij}^k = \frac{r u_\mathrm{R}^k + u_\mathrm{L}^k}{r+1} \tag{5.13}$$

Finally, to complete the computing procedure for equation (5.10), we will use the knowledge from modal analysis of flux Jacobian matrix $\tilde{A}$ to compute the last term of this equation efficiently. The eigenvalues of $\tilde{A}$ are given by

$$\begin{aligned} \tilde{\lambda}^1 &= \tilde{u}_{ij}^l \hat{n}^l + \tilde{c}_{ij} \\ \tilde{\lambda}^2 &= \tilde{u}_{ij}^l \hat{n}^l \\ \tilde{\lambda}^3 &= \tilde{u}_{ij}^l \hat{n}^l - \tilde{c}_{ij} \end{aligned} \tag{5.14}$$

and its eigenvectors are given by

$$\tilde{\mathbf{e}}^1 = \begin{bmatrix} 1 \\ \tilde{u}_{ij}^1 + \tilde{c}_{ij}\hat{n}^1 \\ \tilde{u}_{ij}^2 + \tilde{c}_{ij}\hat{n}^2 \end{bmatrix},\ \tilde{\mathbf{e}}^2 = \begin{bmatrix} 0 \\ -\tilde{c}_{ij}\hat{n}^2 \\ \tilde{c}_{ij}\hat{n}^1 \end{bmatrix},\ \tilde{\mathbf{e}}^3 = \begin{bmatrix} 1 \\ \tilde{u}_{ij}^1 - \tilde{c}_{ij}\hat{n}^1 \\ \tilde{u}_{ij}^2 - \tilde{c}_{ij}\hat{n}^2 \end{bmatrix}. \tag{5.15}$$

Then the numerical flux (5.10) can be evaluated as

27 Ibid. 23.

$$\mathbf{F}_{ij}^k := \frac{1}{2}\left[\left(\mathbf{F}_\mathrm{L}^k + \mathbf{F}_\mathrm{R}^k\right) - \left(\sum_{\phi=1}^{3} \tilde{\alpha}^\phi \left|\tilde{\lambda}^\phi\right| \tilde{\mathbf{e}}^\phi\right)\hat{n}^k\right] \tag{5.16}$$

The variables $\tilde{\alpha}^\phi$ are the coefficients of the decomposition in the basis of eigenvectors of matrix $\tilde{A}$, which are derived from the properties

$$\mathbf{U}_\mathrm{R} - \mathbf{U}_\mathrm{L} = \sum_{\phi=1}^{3} \tilde{\alpha}^\phi \tilde{\mathbf{e}}^\phi. \tag{5.17}$$

Hence, the values of $\tilde{\alpha}^\phi$ depend on the jumps $\Delta(\bullet) := (\bullet)_\mathrm{R} - (\bullet)_\mathrm{L}$ and are given by

$$\begin{aligned}
\tilde{\alpha}^{1,3} &= \frac{\Delta h}{2} \pm \frac{1}{2\tilde{c}_{ij}}\left[\Delta\left(hu_l\right)\hat{n}^l - \left(\tilde{u}_{ij}^l \hat{n}^l\right)\Delta h\right], \\
\tilde{\alpha}^2 &= \frac{1}{\tilde{c}_{ij}}\left\{\left[\Delta\left(hu_2\right) - \tilde{u}_{ij}^2 \Delta h\right]\hat{n}^1 - \left[\Delta\left(hu_1\right) - \tilde{u}_{ij}^1 \Delta h\right]\hat{n}^2\right\}.
\end{aligned} \tag{5.18}$$

Above equations, (5.10) to (5.18), are general descriptions and computational procedures to any order of accuracy for Roe's fluxes. However, when only the *first order* fluxes in (5.10) are considered, the right (R) and left (L) stage values are replaced simply by values at node *j* and *i*, respectively, i.e.

$$\mathbf{F}_{ij}^k := \frac{1}{2}\left[\left(\mathbf{F}_i^k + \mathbf{F}_j^k\right) - \left|\tilde{A}\left(\mathbf{U}_i, \mathbf{U}_j\right)\right| \cdot \left(\mathbf{U}_j - \mathbf{U}_i\right)\hat{n}^k\right] \tag{5.19}$$

Next, we will discuss how to obtain the fluxes in (5.10) more accurate than first order, i.e. equation (5.19).

5.1.2. Higher order schemes

The low-order scheme, equation (5.20), in the previous section is useless in practice. In order to achieve a scheme of order higher than one, the amount of dissipation must be reduced. This implies reducing the magnitude of the difference $\mathbf{U}_j - \mathbf{U}_i$ by replacing the zero-order extrapolation of the variables $\left(\mathbf{U}_\mathrm{L} := \mathbf{U}_i \text{ and } \mathbf{U}_\mathrm{R} := \mathbf{U}_j\right)$ at the midpoint $\mathbf{x}_{ij} := \left(\mathbf{x}_i + \mathbf{x}_j\right)/2$ by a higher-order extrapolation.

The MUSCL (Monotone Upstream-centered Schemes for Conservation Laws) methodology[28] allows achieving second and third-order accuracy of fluxes in (5.10) using linear and quadratic reconstruction (extrapolation) of the conservative variables respectively. The assumption is made that the function behaves smoothly in the vicinity of point $\mathbf{x}_{ij}$ along the edge connected point *i* and *j*. This allows the reconstruction of alternate

[28] Van Leer, B. (1979). Towards the ultimate conservative difference scheme: V. A second order sequel to Godunov's method, *Journal of Computational Physics*, 32: 101-136.

values of right and left stages, denoted by $\mathbf{U}_j^-$ and $\mathbf{U}_i^+$, respectively. This lead to a flux function of the form

$$\mathbf{F}_{ij}^k := \frac{1}{2}\left[\left(\mathbf{F}_i^{k+} + \mathbf{F}_j^{k-}\right) - \left|\tilde{A}\left(\mathbf{U}_i^+, \mathbf{U}_j^-\right)\right| \cdot \left(\mathbf{U}_j^- - \mathbf{U}_i^+\right)\hat{n}^k\right] \tag{5.20}$$

where

$$\mathbf{F}_i^{k+} = \mathbf{F}^k\left(\mathbf{U}_i^+\right) \text{ and } \mathbf{F}_j^{k-} = \mathbf{F}^k\left(\mathbf{U}_j^-\right) \tag{5.21}$$

The upwind-biased approximations for $\mathbf{U}_i^+$ and $\mathbf{U}_j^-$ are defined by

$$\begin{aligned} \mathbf{U}_i^+ &= \mathbf{U}_i + \frac{1}{4}\left[(1-\kappa)\Delta_i^-\mathbf{U} + (1+\kappa)\left(\mathbf{U}_j - \mathbf{U}_i\right)\right] \\ \mathbf{U}_j^- &= \mathbf{U}_j - \frac{1}{4}\left[(1-\kappa)\Delta_j^+\mathbf{U} + (1+\kappa)\left(\mathbf{U}_j - \mathbf{U}_i\right)\right] \end{aligned} \tag{5.22}$$

where the forward and backward operators are given by

$$\begin{aligned} \Delta_j^+\mathbf{U} &= \mathbf{U}_{j+1} - \mathbf{U}_j := 2\mathbf{l}_{ji} \cdot \nabla\mathbf{U}_j - \left(\mathbf{U}_j - \mathbf{U}_i\right) \\ \Delta_i^-\mathbf{U} &= \mathbf{U}_i - \mathbf{U}_{i-1} := 2\mathbf{l}_{ji} \cdot \nabla\mathbf{U}_i - \left(\mathbf{U}_j - \mathbf{U}_i\right) \end{aligned} \tag{5.23}$$

In equations (5.23), we note that the variables $\mathbf{U}_{j+1}$ and $\mathbf{U}_{i-1}$ are *presumed* variables, to the purpose of derivation with respect to one-dimensional case. Hence, in present work, gradients of conservative variable, i.e. $\nabla\mathbf{U}_j$ and $\nabla\mathbf{U}_i$, are used to compute the forward and backward difference in equation (5.23). The parameter κ can be chosen to control the degree of approximation. Set the value of parameter κ to be -1 and +1 lead to a second-order rightward-biased and centered approximations respectively, while $\kappa = \frac{1}{3}$ results in a third-order scheme[29].

5.1.3. Limiting

The Godunov's theorem[30] states that no linear scheme of order higher than one is free of oscillations. This implies that with the higher order extension in previous subsection, some form of limiting in conservative variable will be required. Limiting is still an area of active research and numerous approaches can be found in the literature, see for instance[31]. In present work, two approaches for limiting the extrapolation slopes in the MUSCL

[29] Hirsch, C. (1991). *Numerical Computation of Internal and External Flows*, vol. II: *Computational Methods for Inviscid and Viscous Flows*, Wiley, New York.

[30] Ibid.

[31] Ibid. 2, 27 and 28.

technique are proposed. The first approach is the minmod limiter[32], and the second one uses the Van Albada limiter[33].

5.1.3.1. The minmod limiter. The reconstruction given by an equation (5.22) do not generate oscillations if $\mathbf{U}_i \le \mathbf{U}_i^+ \le \mathbf{U}_j$, and $\mathbf{U}_i \le \mathbf{U}_j^- \le \mathbf{U}_j$ with the differences $\Delta_i^-\mathbf{U}$, $\Delta_j^+\mathbf{U}$ and $\left(\mathbf{U}_j - \mathbf{U}_i\right)$ have the same sign. This can be expressed into the following relations:

$$\left|\Delta_i^- U^a\right| \le \frac{3-\kappa}{1-\kappa}\left|U_j^a - U_i^a\right| \; ; \; a = 1,2,3. \tag{5.24}$$

and

$$\left|\Delta_j^+ U^a\right| \le \frac{3-\kappa}{1-\kappa}\left|U_j^a - U_i^a\right| \; ; \; a = 1,2,3. \tag{5.25}$$

To ensuring all these conditions, the reconstruction of the variables is performed in the following manner

$$\begin{aligned} U_i^{a+} &= U_i^a + \frac{1}{4}\Big\{(1-\kappa)\phi_m\left[\Delta_i^- U^a, b\left(U_j^a - U_i^a\right)\right] + \\ &\qquad (1+\kappa)\phi_m\left[\left(U_j^a - U_i^a\right), b\Delta_i^- U^a\right]\Big\} \; ; \; a = 1,2,3 \\ U_j^{a-} &= U_j^a - \frac{1}{4}\Big\{(1-\kappa)\phi_m\left[\Delta_j^+ U^a, b\left(U_j^a - U_i^a\right)\right] + \\ &\qquad (1+\kappa)\phi_m\left[\left(U_j^a - U_i^a\right), b\Delta_j^+ U^a\right]\Big\} \; ; \; a = 1,2,3 \end{aligned} \tag{5.26}$$

where *a* is used to represent the number of unknowns (i.e. conservative variables), and *b* is so-called *compression parameter* and is defined according to $1 \le b \le (3-\kappa)/(1-\kappa)$, the choice of $b = 1$ leads to a more dissipative scheme. The minmod function, denoted by ϕ_m, can be expressed as

$$\phi_m(x,y) = \mathrm{sgn}(x)\cdot\left[0, \min\left(|x|, \mathrm{sgn}(x)\cdot y\right)\right] \tag{5.27}$$

This function will give an argument, either *x* or *y*, which has a smallest absolute value if they have the same sign, otherwise returns a zero value.

5.1.3.2. The Van Albada limiter. The upwind-biased interpolations in (5.22) are replaced by

[32] Ibid. 23.

[33] Ibid. 2.

$$U_i^{a+} = U_i^a + \frac{s_i^a}{4}\left[\left(1-\kappa s_i^a\right)\Delta_i^- U^a + \left(1+\kappa s_i^a\right)\left(U_j^a - U_i^a\right)\right] \; ; \; a = 1,2,3$$
$$U_j^{a-} = U_j^a - \frac{s_j^a}{4}\left[\left(1-\kappa s_j^a\right)\Delta_j^+ U^a + \left(1+\kappa s_j^a\right)\left(U_j^a - U_i^a\right)\right] \; ; \; a = 1,2,3 \tag{5.28}$$

with no sum on *a*. The flux limiters s_i^a and s_j^a are defined by

$$s_i^a = \max\left\{0, \frac{2\Delta_i^- U^a \cdot \left(U_j^a - U_i^a\right) + \varepsilon}{\left(\Delta_i^- U^a\right)^2 + \left(U_j^a - U_i^a\right)^2 + \varepsilon}\right\} \; ; \; a = 1,2,3$$
$$s_j^a = \max\left\{0, \frac{2\Delta_j^+ U^a \cdot \left(U_j^a - U_i^a\right) + \varepsilon}{\left(\Delta_j^+ U^a\right)^2 + \left(U_j^a - U_i^a\right)^2 + \varepsilon}\right\} \; ; \; a = 1,2,3 \tag{5.29}$$

where ε is a very small number to prevent division by zero in smooth region of the flow. Notice that the values of limiters in (5.29) are bounded between 0 and 1. When the values of limiters are equal to zero, equations (5.28) represent the first order scheme, while limiters equal to unity recover the higher order schemes.

Naturally, the minmod limiter leads to more dissipative results than the Van Albada limiter. Hence, in some cases when the Van Albada limiter allows the oscillation to be appeared, adoption of the minmod limiter should be more suitable.

5.1.4. Lower bound of wave speeds

The acoustic eigenvalues $\tilde{\lambda}^1$ or $\tilde{\lambda}^3$ are vanished at critical flow point (i.e. Froude number equal to unity), and the eigenvalue $\tilde{\lambda}^2$ goes to zero at stagnation point. Vanishing of eigenvalues could cause misbehavior of the dissipation terms in (5.17), and leads to numerical instabilities for particular flow conditions.

A simple way to overcome this problem is limiting the minimum absolute value of all eigenvalues to a fraction of the Jacobian matrix spectral radius $\rho\left(\tilde{A}\right)$, viz.

$$\left|\tilde{\lambda}^1\right| \leftarrow \max\left[\left|\tilde{\lambda}^1\right|, \alpha\rho\left(\tilde{A}\right)\right]$$
$$\left|\tilde{\lambda}^2\right| \leftarrow \max\left[\left|\tilde{\lambda}^2\right|, \beta\rho\left(\tilde{A}\right)\right] \tag{5.30}$$
$$\left|\tilde{\lambda}^3\right| \leftarrow \max\left[\left|\tilde{\lambda}^3\right|, \alpha\rho\left(\tilde{A}\right)\right]$$

where $\rho\left(\tilde{A}\right) = \left|\tilde{u}_{ij}^l \hat{n}^l\right| + \tilde{c}_{ij}$, the parameters $\alpha \approx 0.2$ and $\beta \approx 0.1$. Note that this limiting is corresponding with entropy correction scheme in Euler equations[34].

[34] Ibid. 23.

5.2. Discretization of diffusive fluxes and source terms

Discrete form of diffusive fluxes and source terms are simpler than the convective fluxes. They no need to be stabilized. Again, from expressions (2.14) of the WLSQ approximation procedure, we obtain the following expression for the divergence of the diffusive (viscous) flux function:

$$\mathcal{G}^i := \frac{\partial \mathbf{G}^l(\mathbf{x}_i)}{\partial x_l} \approx D_l^{ij}\mathbf{G}_j^l \tag{5.31}$$

with D_l^{ij} are defined in equation (2.14). Also, the source terms are expressed in the form below,

$$\mathcal{S}^i := \mathbf{S}(\mathbf{x}_i) \approx N^{ij}\mathbf{S}_j. \tag{5.32}$$

Now, we have all information to perform time integration, which solves for conservative variables **U**.

5.3. Discretization in time

From equations (5.1), and the discretization of the fluxes in expressions (5.20), (5.31) and (5.32), we obtain the following system of ODEs for the conservative variables:

$$\frac{\partial \hat{\mathbf{U}}_i}{\partial t} = \mathbf{R}^i := \mathcal{G}^i - \mathcal{F}^i + \mathcal{S}^i \tag{5.33}$$

This system of ODEs is integrated explicitly in time with a multistage Runge-Kutta scheme. Assume that the vector $\mathbf{U}_i^h$ of conservative variables are known for all star point at time $t = t^n$. The right hand side vector $\mathbf{R}^i$ is computed for each star point with current stage of conservative variables $\mathbf{U}^h = \underset{i=1}{\overset{n}{\mathbf{A}}}\mathbf{U}_i^h$ (assemble of $\mathbf{U}_i^h$ for all star points), i.e. $\mathbf{R}^i := \mathscr{R}^i(\mathbf{U}^h)$. The vector of conservative variables at time t^{n+1} is obtained by q-stage Runge-Kutta scheme, viz.

$$\hat{\mathbf{U}}_i^{(s)} = \hat{\mathbf{U}}_i^n + \alpha_s \Delta t_i \mathbf{R}^{i(s-1)}, \quad s = 1, 2, ..., q \tag{5.34}$$

where $\hat{\mathbf{U}}_i^{(0)} \leftarrow \hat{\mathbf{U}}_i^n$ and $\hat{\mathbf{U}}_i^{n+1} \leftarrow \hat{\mathbf{U}}_i^{(q)}$. Time step Δt_i is evaluated at the star point $\mathbf{x}_i$, and α_s are coefficient that depend on the numbers of stages (q). For two, three and four stages schemes these coefficients are set as follows:

$q = 2 \to \alpha_1 = 1/2$ and $\alpha_2 = 1.0$
$q = 3 \to \alpha_1 = \alpha_2 = 3/5$ and $\alpha_3 = 1.0$
$q = 4 \to \alpha_1 = 1/4,\ \alpha_2 = 1/3,\ \alpha_3 = 1/2$ and $\alpha_1 = 1.0$

Each stage in (5.34) have to be ended up with computation of nodal parameter $\mathbf{U}^h$, which will be used to calculate the right hand side vector $\mathbf{R}^i$ of current stage. The following linear system has to be solved

$$\mathbf{M}_c \cdot \mathbf{U}^h = \hat{\mathbf{U}} \qquad (5.35)$$

where $\mathbf{M}_c$ is the equivalent consistent *mass* matrix of the system, which results from the assemble of the shape function matrix, $\mathbf{N}^{ij} = diag\left(N^{ij} \quad N^{ij} \quad N^{ij}\right)$ and $\hat{\mathbf{U}} = \underset{i=1}{\overset{n}{\mathbf{A}}} \hat{\mathbf{U}}_i$. This equation system can be solved by simple iteration scheme easily. The rate of convergence is excellent; require only a few iterations[35].

5.3.1. The time step calculation and stability requirements

Since the multistage Runge-Kutta scheme presented above is explicit, the local time step Δt_i for each star point $\mathbf{x}_i$ must be restricted by a CFL (Courant-Friedriches-Lewy) condition. The time step constraint is of the form

$$\Delta t_i = Cr \min_{j \neq i}\left(\Delta t_j^{\mathrm{ad}}, \Delta t_j^{\mathrm{dif}}\right), \quad \forall \mathbf{x}_j \in \Omega_i \qquad (5.36)$$

with

$$\begin{aligned} \Delta t_j^{\mathrm{ad}} &= \frac{\left\|\mathbf{l}_{ji}\right\|}{\max\left(\lambda_j^{\max}, \lambda_i^{\max}\right)} \\ \Delta t_j^{\mathrm{dif}} &= \frac{\left\|\mathbf{l}_{ji}\right\|^2}{2v} \end{aligned} \qquad (5.37)$$

where $\mathbf{l}_{ji} := \mathbf{x}_j - \mathbf{x}_i$ is the vector link between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ (Figure 4), v is the kinematic viscosity coefficient, and

$$\begin{aligned} \lambda_i^{\max} &= \left|u_i^k \hat{n}^k\right| + c_i \\ \lambda_j^{\max} &= \left|u_j^k \hat{n}^k\right| + c_j \end{aligned} \qquad (5.38)$$

are the maximum wave speeds in the direction of $\mathbf{l}_{ji}$. The Courant number Cr is restricted to $Cr < 1$. The adoption of a local time step Δt_i in equation (5.34) is suitable for stationary problem. If we need an accurately transient solution, a global time step must be used instead, i.e. $\Delta t_i \leftarrow \Delta t_g$. This global time step is defined by

$$\Delta t_g = \min_i\left(\Delta t_i\right), \quad \forall \mathbf{x}_i \in \Omega. \qquad (5.39)$$

[35] Ibid. 2.

5.4. Boundary conditions

In this work three type of boundary conditions are concerned. The first kind is so called *far-field conditions*, applied on outer boundary denoted by Γ_∞, the second kind is the *normal velocity conditions*, applied on boundary denoted by Γ_w, and the third kind is the *no-slip boundary condition*, denoted by Γ_o. Note that these three types of boundary cover whole problem domain, i.e. $\Gamma = \Gamma_\infty \cup \Gamma_\mathrm{w} \cup \Gamma_\mathrm{o}$.

5.4.1. Far-field boundary conditions

For each star point on the outer boundary $\mathbf{x}_i \in \Gamma_\infty$ and their unit outward normal vector $\hat{\mathbf{n}}_\infty$, the normal component of convective flux is modified according to far-field state $\mathbf{U}_\infty$. Given the original convective flux $\mathbf{F}_i^k$, their normal component can be obtained,

$$\mathbf{F}_n = \mathbf{F}_i^k \hat{n}_\infty^k \tag{5.40}$$

Also, the far-field state normal flux vector is given by

$$\mathbf{F}_\infty = \mathbf{F}^k \left(\mathbf{U}_\infty\right) \hat{n}_\infty^k . \tag{5.41}$$

Then, the new normal flux vector is expressed by the solution of the approximate Riemann problem between $\mathbf{U}_i$ and $\mathbf{U}_\infty$, viz.

$$\mathbf{F}_n^* := \frac{1}{2}\left[\left(\mathbf{F}_n + \mathbf{F}_\infty\right) - \left|\tilde{A}\left(\mathbf{U}_\infty, \mathbf{U}_i\right)\right| \cdot \left(\mathbf{U}_\infty - \mathbf{U}_i\right)\right] \tag{5.42}$$

Note that the Roe matrix on above equation is calculated in the direction of outward unit normal vector $\hat{\mathbf{n}}_\infty$ on the outer boundary. Finally, the modified flux vectors in Cartesian coordinate at $\mathbf{x}_i \in \Gamma_\infty$ are obtained

$$\overline{\mathbf{F}}_i^k = \mathbf{F}_i^k + \left(\mathbf{F}_n^* - \mathbf{F}_n\right) \hat{n}_\infty^k \tag{5.43}$$

The modified flux vectors given above are computed for all $\mathbf{x}_i \in \Gamma_\infty$ in each time step. Replace them to original convective flux, i.e. $\mathbf{F}_i^k \leftarrow \overline{\mathbf{F}}_i^k$.

5.4.2. Normal velocity boundary conditions

The most well-known of this boundary condition is so-called slip wall boundary condition, which set the normal component of velocity to be zero. However, for generalized them, in present work we set normal velocity to be any prescribed value, namely

$$\mathbf{u}^\mathrm{T} \cdot \hat{\mathbf{n}}_\mathrm{w} \equiv u_l \hat{n}_\mathrm{w}^l = \overline{u}, \quad \forall \mathbf{x}_i \in \Gamma_\mathrm{w} . \tag{5.44}$$

where $\hat{\mathbf{n}}_{\mathrm{w}}$ is the unit normal outward vector to the boundary point $\mathbf{x}_i \in \Gamma_{\mathrm{w}}$, and $\bar{u}$ is the prescribed value of normal velocity. For each time step, the velocity value at $\mathbf{x}_i \in \Gamma_{\mathrm{w}}$ are modified as follows

$$\mathbf{u} \leftarrow \mathbf{u} + \left(\bar{u} - \mathbf{u}^{\mathrm{T}} \cdot \hat{\mathbf{n}}_{\mathrm{w}}\right)\hat{\mathbf{n}}_{\mathrm{w}}, \quad \forall \mathbf{x}_i \in \Gamma_{\mathrm{w}}. \tag{5.45}$$

where the velocity vectors on the right hand side are an original one.

5.4.3. No-slip boundary conditions

This type of boundary conditions will be considered when viscosity effect has been represented. The implementation of no-slip boundary condition is very simple. All components of velocity are set to be zero, i.e.

$$\mathbf{u} \leftarrow \mathbf{0}, \quad \forall \mathbf{x}_i \in \Gamma_{\mathrm{o}}. \tag{5.46}$$

Conditions (5.46) are equivalent to set the normal and tangential velocities to be zeros.

6. NUMERICAL EXAMPLES

Three test problems are considered to assess the schemes present in Section 5. The Van Albada limiter is adopted for all tests. The time step is limited by formulas (5.37) to (5.40). The Courant number is set to 0.5. Two stage Runge-Kutta is used to perform time integration in equation (5.35).

6.1. Oblique hydraulic jump

The geometry is a 40 m long flat-bottomed channel where the upstream entry width of 30 m is narrowed to the exit by a converging wall deflected through an angle $\theta = 8.96^{\circ}$ (Figure 11). The initial conditions are chosen to be $h = 1$ m, $u_1 = 8.57$ m/sec and $u_2 = 0$. Supercritical boundary conditions are imposed on the upstream boundary (left-side) with the same value as initial conditions. Friction along the channel walls (top and bottom) is ignored. Local time step is used to observe the steady state result.

The global cloud of 1309 points is used to discretized the problem domain. The result is presented as a depth contour plot in Figure 12. The angle of shock front β, is measured and its value approximately equal to 30°, which close to analytical solution[36].

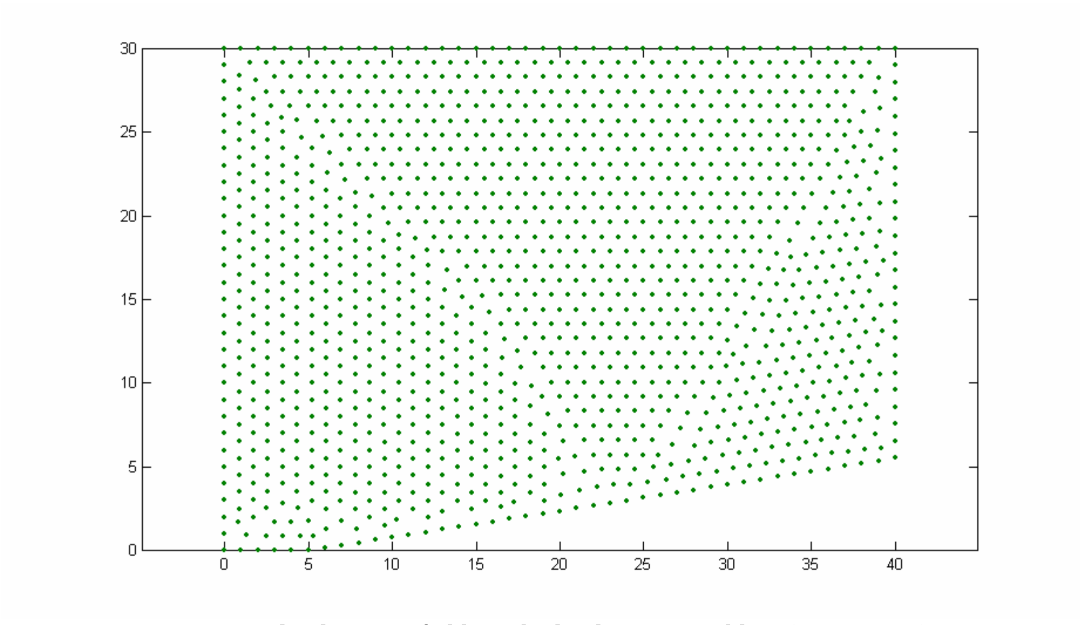

Figure 11. Cloud points of oblique hydraulic jump problem (1309 points).

[36] Ibid. 21.

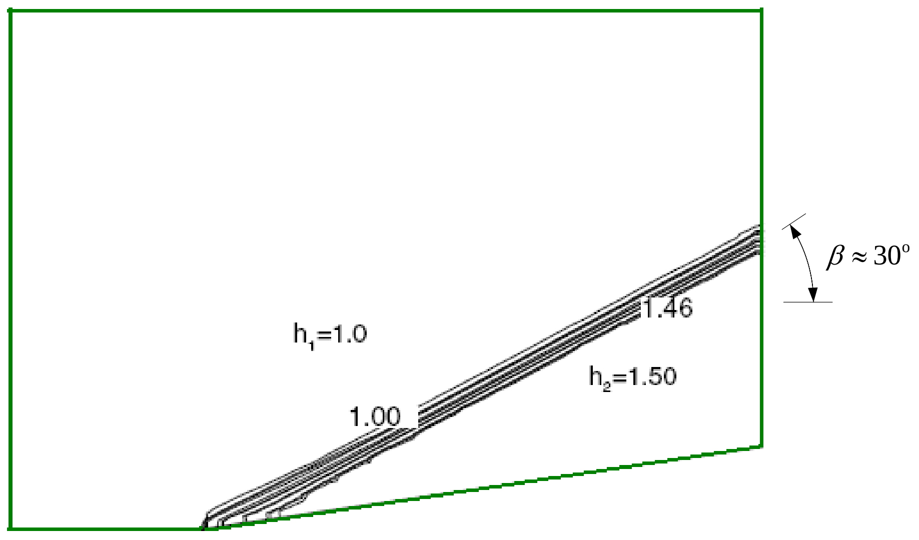

Figure 12. Depth contour plot for oblique hydraulic jump.

6.2. 1D dam break

In this test, we consider a straight channel modeled in 2D, with the headwater and tail water separated by a rigid diaphragm (dam). The viscous effect is not considered. Initially, the water has a different depth on each side of the dam, D_L and D_R (Figure 5) and assumed to be at rest. At initial time $t = 0$, the dam is instantly removed. It creates a bore wave moving from left to right (in x_1-direction), and a depression wave propagating towards the left. In this case, the length of a fixed rectangular region is set to 1.0 m in x_1-direction and 0.5 m in x_2-direction with a barrier at x_1 = 0.5 m. The model discretized by 3772 cloud points (Figure 6). The global time step is used to investigate the transient solution.

The numerical results are taken from the profile cut along x_2 = 0.25 m. Exact solution (solid line) obtained from Wu et al. [37]. The results for two different initial water depth ratio (D_L/ D_R) are plot in Figures 7 and 8.

[37] Wu, C., Huang, G. and Zheng, Y. (1999). Theoretical solution of dam-break shock wave, *Journal of Hydraulic Engineering*, November: 1210-1215.

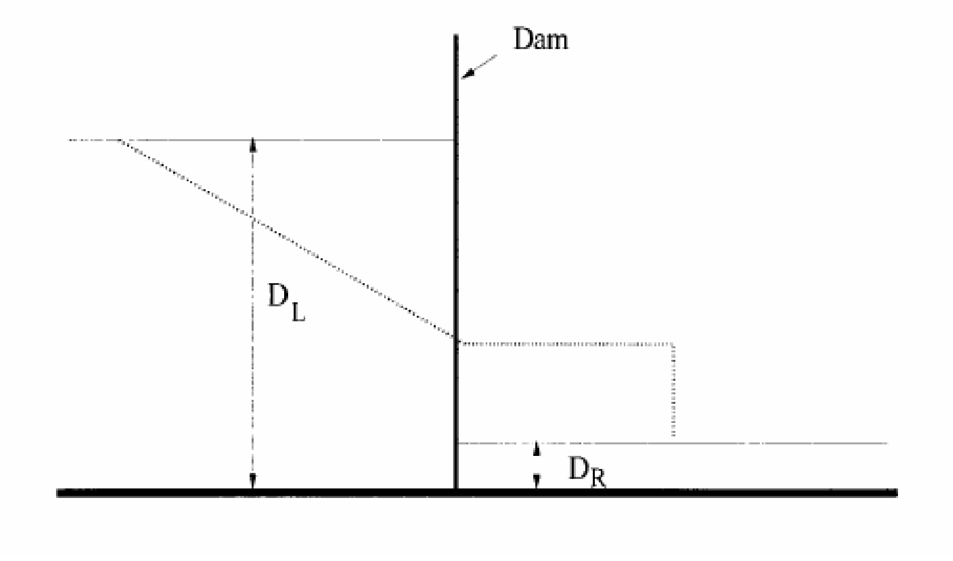

Figure 5. 1D dam break problem

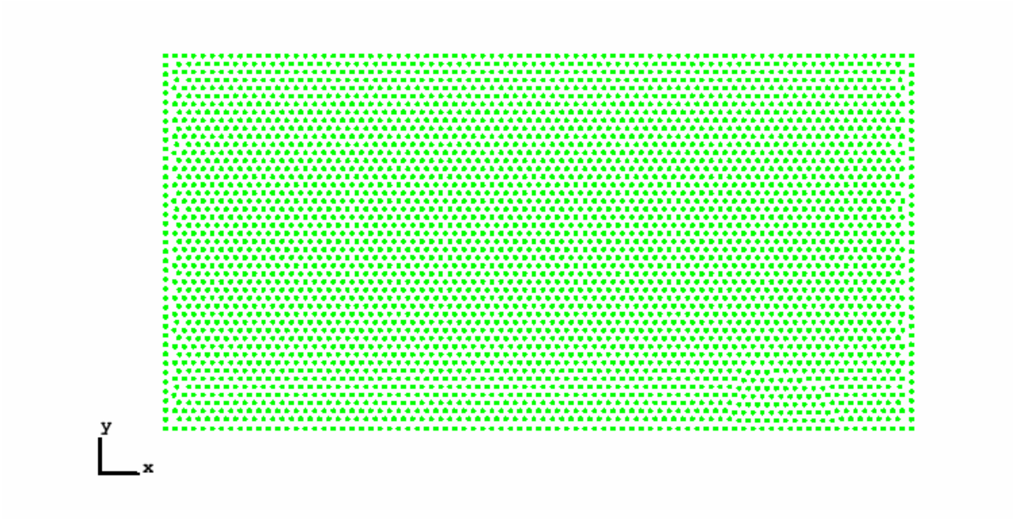

Figure 6. Cloud point of 1D dam break problem (3772 points).

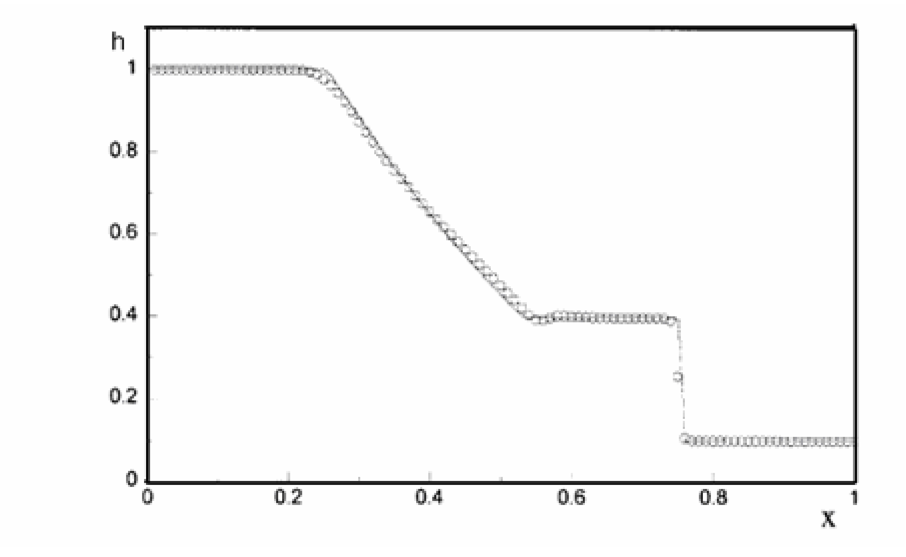

Figure 7. Water elevation after dam break with initial water depth ratio $D_L/D_R = 10$ ($t = 0.25$ s).

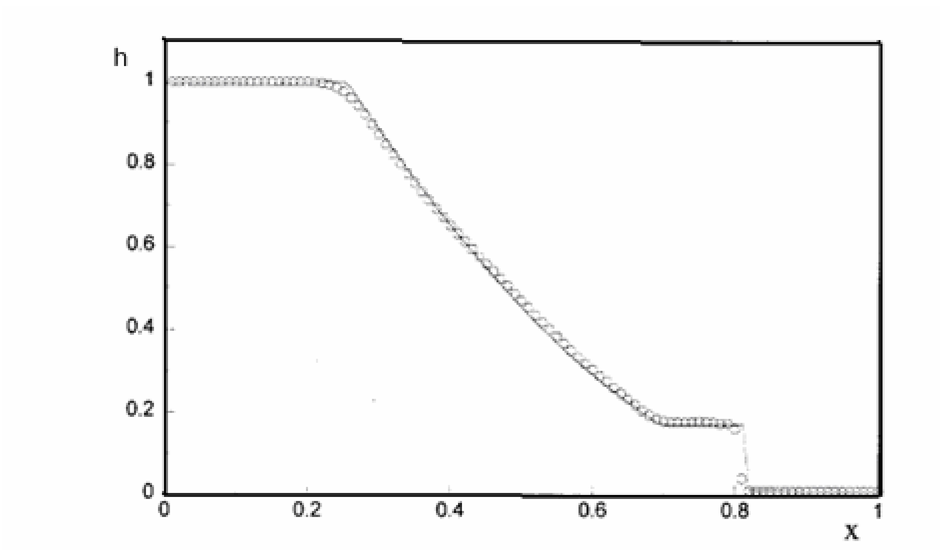

Figure 8. Water elevation after dam break with initial water depth ratio D_L/D_R = 100 (t = 0.25 s).

6.3. Flow past a backward step

A steady flow past a backward step is examined and the reattachment length of the flow field is calculated. This problem has been studied numerically by Wang and Liu[38]. The step dimension is 1 m, the upstream boundary is located at a distance 12 m from the step and the downstream boundary is at a distance 13 m form the step as shown below in Figure 9.

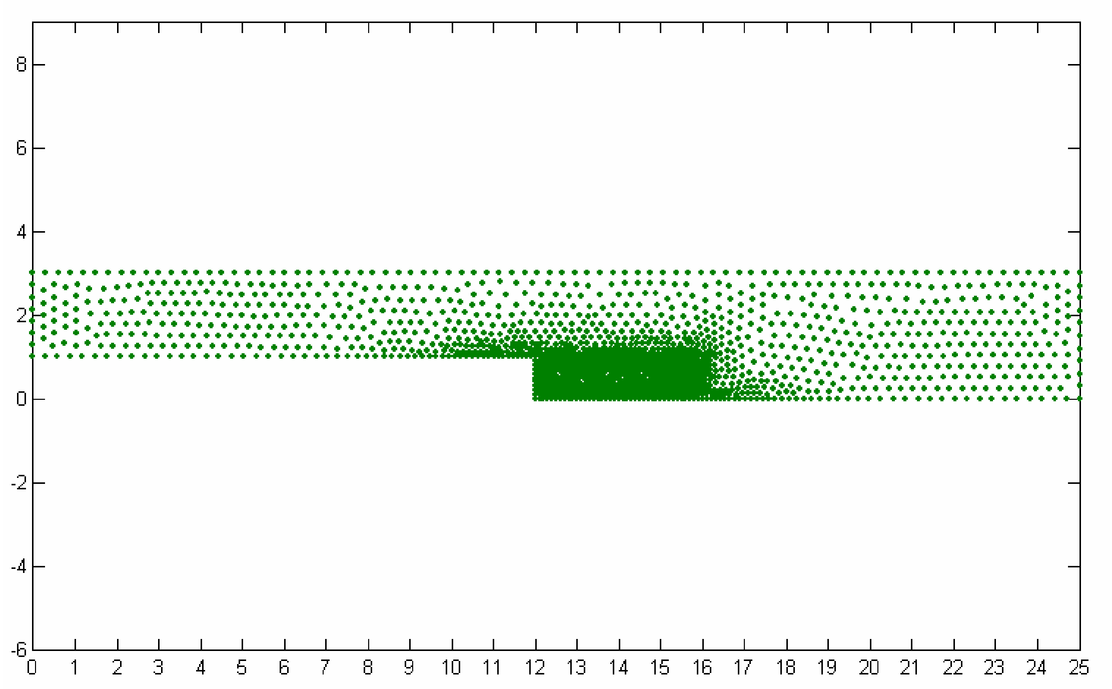

Figure 9. Cloud point of backward facing step problem (1577 points).

38 Ibid. 21.

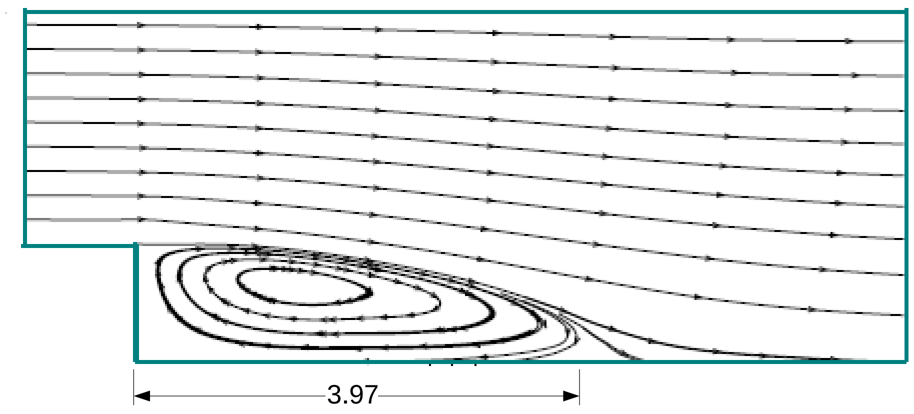

Figure 10. Streamline plot for flow past a backward step.

The initial water depth is taken to be unity. A no-slip wall boundary condition is used. At the inflow boundary, a velocity 0.5 m/sec is prescribed. A still water level of 1 m is prescribed at the outflow boundary. The kinetic viscosity coefficient is set to be 0.00685 m^2/sec. In this problem the local time step is used to investigate the steady solution. The computed streamline near the step is shown in Figure 10. The measured value for the length of the recirculation region is about 3.97.m which is very close to 3.95 m given in Wang and Liu.[39]

6.4. Flow in a sloping channel with varying width

Consider the flow in a channel with a smooth constriction and a sloping bottom surface. The channel is 10000 m long, and the breadth varies from 1000 m to 500 m, and to 1000 m (Figure 13) with a cosine function. The bed slope is taken to be a constant value of 1:100 (downward). At the left end the mass flow $Q = 2000 \ \text{m}^3/\text{sec}$, is specified, and at the right end the depth is specified by extrapolation from the interior. Initial water depth is specified to be 1 m. Local time step is used.

The results are taken from the center line of channel. Water depth profile in Figure 14 varies in similar with a cosine function. The mass flow Q in Figure 15 (solid line) is closely to the value of $2000 \ \text{m}^3/\text{sec}$, which refers by blue-dot line.

[39] Ibid.

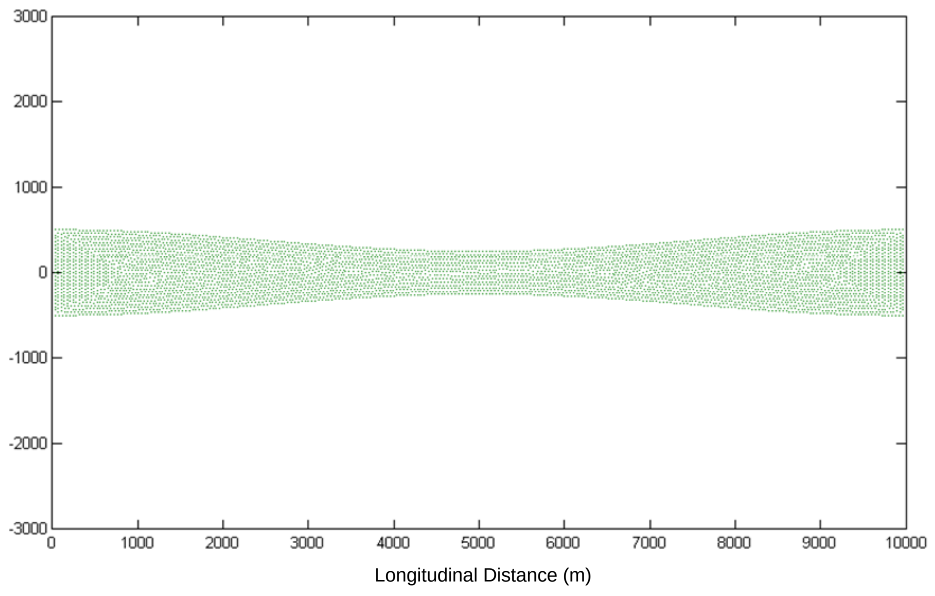

Figure 13. Cloud points of a channel with varying width (5584 points).

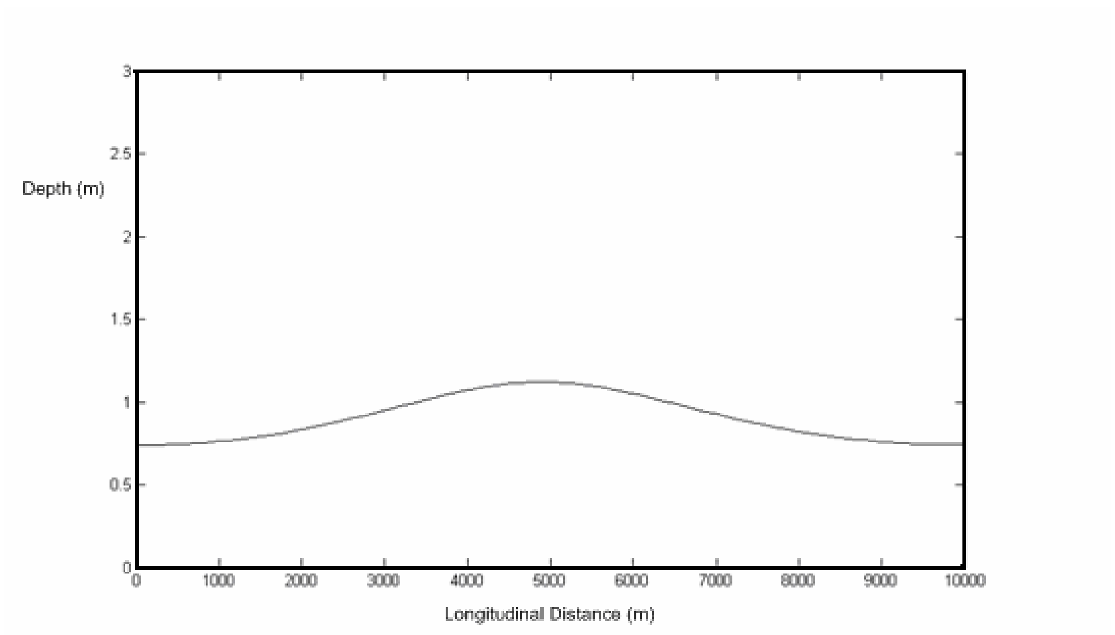

Figure 14. A plot of water depth along the center line of channel.

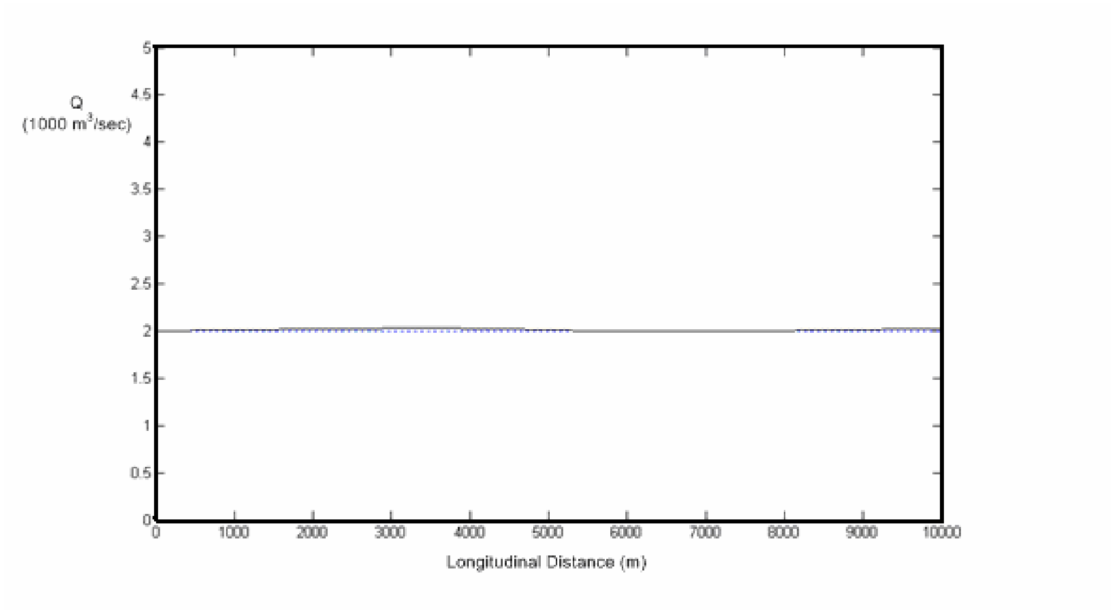

Figure 15. A plot of mass flow along the channel.

7. CONCLUSIONS

The finite point method for shallow water flow has been developed. Starting form a global cloud of points, we used a robust iterative scheme to compute a local approximation for shape functions and gradients. These approximation factors are used to incorporate with approximate Riemann solvers. The results obtained show accuracy close to available analytical and experimental solutions.

Simulation of shallow water flow in large scale may suffer due to the complication of the source term, e.g. the bathymetry slope and base shear terms. These sometimes cause the misbehavior and solutions are not converged. The future research should be study on how to improve the solution in the case of complex source term. Also, another interesting topic is an implementation of adaptive FPM to the wave climb on the sloping beach. Naturally, this problem needs the robust scheme to regeneration of cloud points when the borders of ocean are moved into the dry bed region.

ACKNOWLEDGEMENTS

A considerable part of this work was carried out while the author was training at the International Center for Numerical in Engineering (CIMNE), Barcelona, Spain, in the August 2008. The support for this training by Prof. Eugenio Oñate and Prof. Worsak Kanok-Nukulchai is gratefully acknowledged. Also, the author has to deeply thankful to a very great comment, discussion and kindly help from Sr. Enrique Ortega.