The Shallow Water Model

Markus Uhlmann

November 2001, February 2008, last revised May 8, 2012

Abstract

We give an introduction to the system of shallow water equations. Emphasis is put upon its analytical properties and its implications for numerical solution strategies.

1 Introduction

1.1 Simulated Flow

The movement of an incompressible fluid ($\nabla \cdot \vec{v} = 0$) with constant density under the influence of a gravitational body force is considered. The description is basically inviscid except for the possible inclusion of a viscous bottom friction term.

Vertical accelerations of the fluid are neglected, which allows to integrate the remaining part of the vertical momentum equation and to obtain an expression for the pressure which in turn can then be eliminated from the system. The error associated with this approximation is of the order of $-p_{,x}/(\rho u_t) \approx h^2/l^2$ (h undisturbed water height, l characteristic length scale of the waves in x-direction). This estimate is equivalent to the so-called "long-wave limit" of wave motion, i.e. we are dealing with either very long waves or with shallow water. Physically, the horizontal velocity that is retained can be interpreted as a vertical average of the fluid velocity.

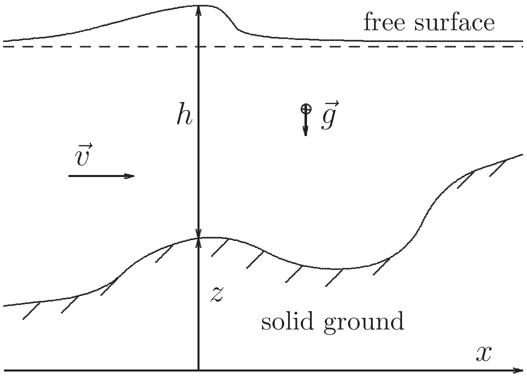


Figure 1: Schematic of the coordinates and variables of the shallow water model

The system of equations that governs the above class of flows can be conveniently written as follows:

$$\partial_t \vec{Q} + \partial_{x_i} \vec{F}_i = \vec{S} \quad , \tag{1}$$

where

$$\vec{Q} = \begin{pmatrix} h \\ uh \\ vh \end{pmatrix}, \quad \vec{F}_x = \begin{pmatrix} uh \\ u^2h + gh^2/2 \\ uvh \end{pmatrix}, \quad \vec{F}_y = \begin{pmatrix} vh \\ uvh \\ v^2h + gh^2/2 \end{pmatrix}, \quad \vec{S} = \begin{pmatrix} 0 \\ gh(-z_{,x} - S_{fx}) \\ gh(-z_{,y} - S_{fy}) \end{pmatrix}.$$

In the above notation, h signifies the water depth, u and v the horizontal velocity coordinates of the fluid, g the gravitational accelaration, z the vertical coordinate of the bottom taken from some reference point, S_{fx} and S_{fy} denote the two components of the bottom friction that can be expressed by the Manning formula

$$S_{fx} = \frac{n^2 u\sqrt{u^2 + v^2}}{h^{4/3}}, \quad S_{fy} = \frac{n^2 v\sqrt{u^2 + v^2}}{h^{4/3}} \tag{2}$$

where n is an empirical roughness coefficient.

1.2 Applications

Possible applications of the shallow water – or St. Venant – equations include various problems of open surface hydraulics: river hydraulics, man-made canals and irrigation systems as well as marine flows.

- A well-known case from the class of land-based or artificial flows is the prediction of the time evolution of the flood wave induced by a rupture of a containing dam [1, 2] which obviously has important implications for hasard assesssment.
- An example of an environmental flow study based on the shallow water approximation is the computation of the wave motion induced by a land slide into a lake [3, p. 173].
- A marine flow prediction of particular interest is connected to *tsunamis*: a given seismically induced wave propagates in the region of the continental shallows and shelf [4].

Many more applications exist such as wave motion in harbor basins or flow in urban sewage systems.

1.3 Limitations

The use of the St. Venant equations is above all limited by the previously mentioned assumption of long waves. When the gravitational waves can no longer be considered as “long” (i.e. $h/l \sim 1$), the hypothesis of two-dimensionality of the flow becomes invalid. Particularly, so-called “breaking and peaking” of waves near the shoreline cannot be realistically captured by this model. Surface tension (capillary effects) is equally excluded in the present description since these effects are associated to the regime of very short waves.

Furthermore, the effect of a partially free surface, e.g. in a combination of pipe and river flow, can only be taken into acount by a split of the problem into zones where the respective models would be applied.

In the case of very large domains (marine flows extending over a large range in latitude) Coriolis forces should be included in the equations.

The effect of wind on waves is not taken into account, i.e. the action of tangential stress at the water/air interface is neglected.

1.4 Present Approach

The system of equations (1) is of strictly hyperbolic type and as such admits discontinous solutions either as a consequence of discontinous initial data or in some cases through the evolution from initially smooth data (wave steepening).

The theory of characteristics allows to construct solutions to a number of elementary problems including discontinuities (such as the hydraulic jump). However, the analysis is restricted to simple cases such that a numerical solution is desirable. This is our present objective.

In the present note we will first develop in detail the analytical solution to the Riemann (discontinous initial-value) problem for the shallow water equations in the flat-bottom case. In this we will follow the general technique laid out by Smoller [5]. Besides its direct importance for dam-break flows the Riemann problem exhibits the essential physics encountered in more complex cases while keeping the geometric configuration simple and tractable. As such it has become a reference test case of choice for validating numerical methods. Moreover, the solution to the very Riemann problem has found its way into a good part of the modern numerical schemes conceived for the shallow water equations (and in fact for most hyperbolic systems).

In section 3 we will then present several more recent numerical approaches applicable to our system (1). We will especially discuss the difficulties associated with the numerical treatment of the source terms in the non-uniform bottom case which deprive the system of its conservation property.

2 Discussion of the Riemann problem

In the following we will concentrate on the time evolution of our flow model from an initial state that consists of two semi-infinite uniform zones which are separated by a discontinuity. One can imagine a realization of this situation by positioning a diaphragm ("infinitely thin dam") between the two fluid states and somehow rupture it at time $t = 0$. Our objective is to determine the resulting induced wave motion as a function of the initial state. This problem is geometrically one-dimensional in that the solution only depends on one space coordinate normal to the diaphragm, say $x\,\vec{e}_n$. In this section we will first restrict our analysis to one-dimensional motion, i.e. $\vec{v} = u\,\vec{e}_n$, and the flat-bottom case without bottom friction, viz.

$$\partial_t \vec{Q} + \partial_x \vec{F} = 0\,, \quad \vec{Q} = \begin{pmatrix} h \\ uh \end{pmatrix}, \quad \vec{F} = \begin{pmatrix} uh \\ u^2h + gh^2/2 \end{pmatrix}. \tag{3}$$

We will return to the two-dimensional case and the question of a non-uniform bottom at a later point.

2.1 Characterization of the system of equations

The jacobian matrix $\mathbf{J}$ of equations (3),

$$\mathbf{J} \equiv \frac{\partial F}{\partial Q} = \begin{pmatrix} 0 & 1 \\ -u^2 + gh & 2u \end{pmatrix} \quad , \tag{4}$$

has two distinct and real eigenvalues $\lambda_1 < 0 < \lambda_2$,

$$\lambda_1 = u - c\,, \quad \lambda_2 = u + c\,, \quad c \equiv \sqrt{gh} \quad , \tag{5}$$

such that we are dealing with a strictly hyperbolic system. We note that the celerity of gravitational waves $c = \sqrt{gh}$ takes the place the speed of sound has in gas dynamics and the Froude number $\mathrm{Fr} \equiv |u|/c$ is the analogue to the Mach number.

Let us reduce the system (3) to a diagonal form. For this we need the right eigenvectors $\vec{r^i}$ satisfying $(\mathbf{J} - \lambda_i \mathbf{I}) \cdot \vec{r^i} = 0$ which form the columns of the following matrix $\mathbf{R}$,

$$\mathbf{R} = \left(\begin{array}{cc} 1 & 1 \\ u-c & u+c \end{array} \right) \quad , \tag{6}$$

with the inverse

$$\mathbf{R}^{-1} = \left(\begin{array}{cc} u+c & -1 \\ c-u & 1 \end{array} \right) \cdot \frac{1}{2c} \quad . \tag{7}$$

The diagonalization of the jacobian $\mathbf{J}$ (denoting $\Lambda \equiv \mathrm{diag}(\lambda_1, \lambda_2)$) can be written as:

$$\partial_t \vec{Q} + \mathbf{R} \Lambda \mathbf{R}^{-1} \partial_n \vec{Q} = 0 \,. \tag{8}$$

In the *linear* case, i.e. when the Jacobian is constant, we have

$$\begin{aligned} \longrightarrow \mathbf{R}^{-1} \partial_t \vec{Q} + \Lambda \mathbf{R}^{-1} \partial_n \vec{Q} &= 0 \qquad (9) \\ \longrightarrow \partial_t \vec{W} + \Lambda \partial_n \vec{W} &= 0 \qquad (10) \end{aligned}$$

where $\mathbf{R}^{-1} \partial \vec{Q} = \partial \vec{W}$ defines the characteristic variables W_i which are advected along characteristic lines with respective wave speeds λ_i.

2.2 Analytic solution to Riemann's problem

Let us consider the initial-value problem with the following data

$$\vec{Q}(x, t=0) = \left\{ \begin{array}{ll} \vec{Q}_L & x < 0 \\ \vec{Q}_R & x > 0 \end{array} \right. \quad . \tag{11}$$

In our case, both characteristic fields are genuinely non-linear (GNL), since [5]

$$\frac{\partial \lambda_i}{\partial \vec{Q}} \cdot \vec{r^i} \neq 0 \qquad i = \{1, 2\} \quad . \tag{12}$$

This fact signifies that the two wave-fields can either be smooth solutions called simple waves ("rarefactions" in the terminology of gas dynamics) or (discontinous) shocks. In general it exists one more possible type of solution, that of a contact discontinuity, which however does not appear in the present one-dimensional case but is encountered when carrying both velocity components in the equations (cf. section **??**).

The solution to (11) will consist of constant states separated by either shock waves or rarefaction waves. The solution for $t > 0$ can be divided into zones as shown in figure 2. Zones "1" and "2" have zero width in the case of a respective shock and will otherwise consist of a smooth variation over a finite interval.

We begin by studying shock waves. The notion of a *weak solution* of a conservation law has been coined by Lax [6] and is explained in § A. A volume integral balance across a discontinuity propagating at speed σ gives for our system ($[\cdot]$ denoting a jump)

$$-\sigma \left[\vec{Q} \right] + \left[\vec{F} \right] = 0 \quad , \tag{13}$$

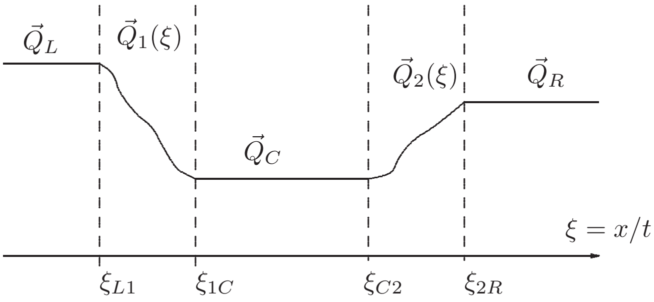


Figure 2: Zones and limiting coordinates of the solution to the Riemann problem for $t > 0$.

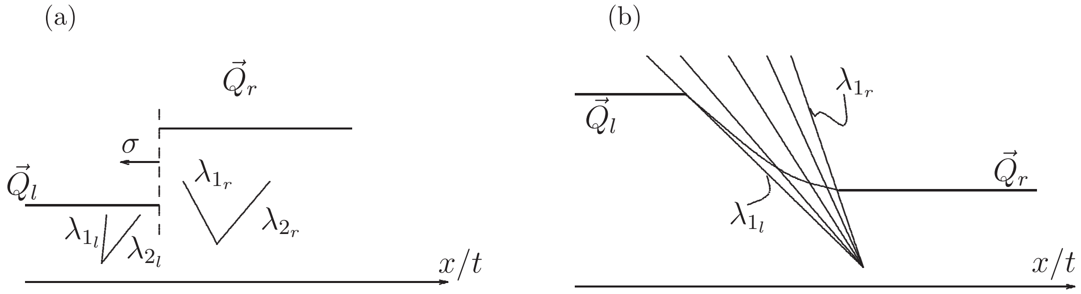


Figure 3: Schematic of (a) a 1-shock and (b) a 1-rarefaction wave.

such that we get explicitly

$$-\sigma\,[h] + [hu] \;=\; 0\,, \tag{14}$$

$$-\sigma\,[hu] + \left[hu^2 + gh^2/2\right] \;=\; 0\,. \tag{15}$$

The mathematical solution for a jump between left and right states $\vec{Q}_l$, $\vec{Q}_r$ satisfying $x_l < x_r$ is the following

$$\sigma = \frac{z\,u_r - u_l}{z-1}\,, \quad [u] = \pm\frac{c_l}{\sqrt{2}}(z-1)\sqrt{\frac{z+1}{z}}\,, \tag{16}$$

where the ratio $z = h_r/h_l$ is a paramter. Next we need to select the physically meaningful sign of the velocity jump in (16). From the shock conditions first derived by Lax [6] (also called "entropy conditions" with respect to gas dynamics) we know that the following inequalities hold for 1-shocks

$$\sigma < \lambda_1(\vec{Q}_l)\,, \qquad \lambda_1(\vec{Q}_r) < \sigma < \lambda_2(\vec{Q}_r) \quad . \tag{17}$$

By substitution of the jump conditions (16) one can verify that these inequalities can only be satisfied when $z > 1$ and the − sign is selected in (16) in the case of a 1-shock. Moreover, σ is seen to be negative and the inequalities $\sigma < u_r$, $\sigma < u_l$ hold. We thus have the situation shown in figure 3(a). The flow constantly feeds characteristic waves into the back of the shock while no information on the shock preceeds its own propagation. The fact that $z > 0$ shows a shock to be "compressive", i.e. that our height variable h always increases when a shock has passed. Finally, the bounds on σ prove that fluid particles pass a 1-shock from left to right.

The case of a 2-shock is analogous: we obtain $z < 1$ and need to select the + sign in (16).

Let us now turn to the rarefaction curves. These are so-called simple waves, i.e. regions of the solution in which the variables vary smoothly between two states. Since the k-characteristic speed always increases through a k-rarefaction curve ($\lambda_k(\vec{Q}(x/t))$ increases as x/t increases) the k-characteristic paths do not cross and form what is sometimes called an expansion fan (figure 3(b)). One has thus in a 1-rarefaction wave:

$$\lambda_1(\vec{Q}_r) \geq \lambda_1(\vec{Q}_l) \quad . \tag{18}$$

Associated to each k-simple wave there are $(n-1)$ k-Riemann invariants I_R^k that satisfy

$$\frac{\partial I_R^k}{\partial \vec{Q}} \cdot \vec{r}^k = 0 \quad , \tag{19}$$

and which remain constant within the interval covered by the k-simple wave. In our case (of an $n = 2$ dimensional system of equations), it can be verified that the following expressions are the two Riemann invariants:

$$I_R^1 = u + 2c \,, \qquad I_R^2 = u - 2c \,. \tag{20}$$

With respect to the 1-rarefaction curve, we thus have

$$u_l + 2c_l = u_r + 2c_r \quad . \tag{21}$$

Using (18) and (21) we can conclude that $1 \geq (h_r/h_l) > 0$, i.e. that the water height decreases in a rarefaction wave. Again, the situation is analogous in the case of a 2-rarefaction ($\lambda_2(\vec{Q}_r) \geq \lambda_2(\vec{Q}_l)$ and $1 \geq (h_l/h_r) > 0$). We can now introduce a dimensionless parameter $\chi \equiv -\log(h_r/h_l) \geq 0$ ($\chi \equiv \log(h_r/h_l) \geq 0$ for a 2-rarefaction) such that the rarefaction curves can be described by the following formulas:

$$\begin{array}{cc} \text{1-rarefaction curve} & \text{2-rarefaction curve} \\ \chi \geq 0 \left\{ \begin{array}{lcl} h_r/h_l & = & e^{-\chi} \\ \dfrac{u_r - u_l}{c_l} & = & 2\left(1 - e^{-\chi/2}\right) \end{array} \right. & \chi \geq 0 \left\{ \begin{array}{lcl} h_r/h_l & = & e^{\chi} \\ \dfrac{u_r - u_l}{c_l} & = & 2\left(e^{\chi/2} - 1\right) \,. \end{array} \right. \end{array} \tag{22}$$

Similarly, the shock curves can be parmetrized using $\chi \equiv -\log(z) \leq 0$ for the 1-shock and $\chi \equiv \log(z) \leq 0$ for the 2-shock, viz.

$$\begin{array}{cc} \text{1-shock curve} & \text{2-shock curve} \\ \chi \leq 0 \left\{ \begin{array}{lcl} h_r/h_l & = & e^{-\chi} \\ \dfrac{u_r - u_l}{c_l} & = & -\left(e^{-\chi} - 1\right)\sqrt{\dfrac{e^{-\chi}+1}{2e^{-\chi}}} \end{array} \right. & \chi \leq 0 \left\{ \begin{array}{lcl} h_r/h_l & = & e^{\chi} \\ \dfrac{u_r - u_l}{c_l} & = & +\left(e^{\chi} - 1\right)\sqrt{\dfrac{e^{\chi}+1}{2e^{\chi}}} \,. \end{array} \right. \end{array} \tag{23}$$

With these ingredients it is now possible to put together the one-parameter functions associated with each "family" of curves, i.e. for each wave field whatever its type might be:

1-family, χ real

$$\frac{h_r}{h_l} = e^{-\chi}\,, \qquad \frac{u_r - u_l}{c_l} = f_1(\chi) = \begin{cases} 2\left(1 - e^{-\chi/2}\right) & x \geq 0 \\ -\left(e^{-\chi} - 1\right)\sqrt{\dfrac{e^{-\chi}+1}{2e^{-\chi}}} & x \leq 0 \end{cases} \tag{24}$$

2-family, χ real

$$\frac{h_r}{h_l} = e^{\chi}\,, \qquad \frac{u_r - u_l}{c_l} = f_2(\chi) = \begin{cases} 2\left(e^{\chi/2} - 1\right) & x \geq 0 \\ \left(e^{\chi} - 1\right)\sqrt{\dfrac{e^{\chi}+1}{2e^{\chi}}} & x \leq 0\,. \end{cases}$$

The global solution to the Riemann problem is obtained by defining a set of transformations across each field in order to connect the left and right *initial* states. Before proceeding, we need the following three properties:

$$f_1'(\chi) > 0\,, \tag{25}$$
$$f_1(\mathrm{I\!R}) = [-\infty, 2]\,, \tag{26}$$
$$f_2(\chi) = f_1(\chi)\cdot e^{\chi/2}\,. \tag{27}$$

Particularly, the first inequality (25) can be verified by considering that the drivative f_1' is monotone in both the positive and negative branch of χ.

Defining the vector of primitive variables $\vec{V} = (h, u) = (v_1, v_2)$, we introduce now the general transformations T^i_χ, $i = 1, 2$:

$$\begin{aligned} T^{(1)}_\chi \cdot \vec{V} &= \left(e^{-\chi}\cdot v_1, v_2 + f_1(\chi)\sqrt{g}\sqrt{v_1}\right)\,, \\ T^{(2)}_\chi \cdot \vec{V} &= \left(e^{\chi}\cdot v_1, v_2 + f_2(\chi)\sqrt{g}\sqrt{v_1}\right)\,. \end{aligned} \tag{28}$$

Solving the Riemann problem comes down to determining the real numbers χ_1 and χ_2 for which:

$$\vec{V}_R = T^{(2)}_{\chi_2}\cdot T^{(1)}_{\chi_1}\cdot \vec{V}_L \quad . \tag{29}$$

Explicitly, the transformation given in (29) can be written as:

$$\begin{pmatrix} h_R \\ u_R \end{pmatrix} = \begin{pmatrix} e^{-\chi_1}\cdot e^{\chi_2}\cdot h_L \\ u_L + c_L\left(f_1(\chi_1) + f_2(\chi_2)\cdot e^{-\chi_1/2}\right) \end{pmatrix} \tag{30}$$

It is useful to define the *global* parameters

$$B \equiv \frac{h_R}{h_L}\,, \qquad C \equiv \frac{u_R - u_L}{c_L}\,. \tag{31}$$

From the first component of (30) we deduce

$$\log(B) = \chi_2 - \chi_1 \quad , \tag{32}$$

while we get from the second component

$$C = f_1(\chi_1) + f_2(\chi_2)\cdot e^{-\chi_1/2} \quad . \tag{33}$$

Using the property (27) and relation (32), we rewrite this as

$$C = f_1(\chi_1) + f_1\left(\chi_1 + \log(B)\right)\sqrt{B} \quad . \tag{34}$$

We can now: (i) find bounds on the admissiblility of the initial data and (ii) formulate criteria for the appearance of either rarefactions or shock curves in the solution depending on the initial data. To this end let us consider the rarefaction curves which present the critical part of the solution.

Suppose that the 1-family is a rarefaction curve, i.e. $\chi_1 \geq 0$. In the limit between a shock/-rarefaction $\chi_1 = 0$ and from (34) we have $C = f_1(\log(B))\sqrt{B}$. Since we know that $f_1' > 0$, we find that in the general case (whith $\chi_1 \geq 0$):

$$C > f_1(\log(B))\sqrt{B} \quad , \tag{35}$$

for the 1-family to be a rarefaction curve. The solution of (34) in this case can be given explicitly, viz.

$$\chi_1 = -2\log\left(\frac{1}{2}\left(1+\sqrt{B}\right) - \frac{1}{4}C\right) \quad , \tag{36}$$

from which arises as a further condition

$$2\left(1+\sqrt{B}\right) > C \quad , \tag{37}$$

for χ_1 to be real. If (37) is violated no real χ_1 exists and in fact a dry zone develops on the back side of the rarefaction. In that case, the velocity is not defined and we will arbitrarily set it to zero.

Looking at the 2-family, we obtain from (34) and using (32):

$$C > f_1\left(-\log(B)\right) \quad , \tag{38}$$

for the 2-family to be a simple wave. In that case the solution is

$$\chi_2 = -2\log\left(-\frac{C}{4\sqrt{B}} + \frac{1}{2\sqrt{B}} + \frac{1}{2}\right) \quad , \tag{39}$$

from which the admissibility criterion (beyond which a dry zone occurs):

$$2\left(1+\sqrt{B}\right) > C \tag{40}$$

as in the previous case. Note that the solution for the shock curves (i.e. χ_1 and/or χ_2 negative) is an implicit function for the χ's that has to be solved numerically. However, a solution of the shock curve is only necessary in the case that *both* families are shock curves since otherwise the problem is entirely defined by specifying the central state from the rarefaction relations and the variation across the shock is consistently obtained. More specifically, in the mentioned case of a double shock solution – returning to the height ratio z_1 as the unknown parameter – we can choose to solve the following formula that applies to the 1-shock:

$$-\frac{1}{\sqrt{2}}(z_1-1)\sqrt{\frac{z_1+1}{z_1}}-\frac{1}{\sqrt{2}}(z_1/B-1)\sqrt{\frac{z_1/B+1}{z_1/B}}\sqrt{B}-C=0 \quad , \tag{41}$$

by using a Newton iteration, say. With respect to admissibility, the shock curves do not introduce any further limitation on the initial data. We can now conclude:

The Riemann problem (11) associated with the system (3) has a unique solution if

$$u_R-u_L<2\,(c_L+c_R)\,, \tag{42}$$

otherwise a dry zone develops in between the initial states. The 1-family of waves is a simple wave if

$$f_1(\log(B))\sqrt{B}<C<2\,(1+\sqrt{B}) \tag{43}$$

and otherwise a shock wave. The 2-family is a simple wave if

$$f_1(-\log(B))<C<2\,(1+\sqrt{B}) \tag{44}$$

and otherwise a shock wave (C and B are defined in (31)).

In practice, one of the following four configurations can arise:

(i) rarefaction–shock,

(ii) rarefaction–rarefaction,

(iii) shock–rarefaction,

(iv) shock–shock,

which we can now decide from the initial data by applying (43) and (44). The central state $\vec{Q}_C$ can thus be deduced from equation (36) in cases (i)-(ii) and from equation (39) in case (iii) while iterating (41) is necessary in case (iv). What is now left to do in order to terminate the solution is the determination of the limit coordinates of the zones “1” and “2” (figure 2) and the explicit variation of variables through the rarefaction curves.

We first remark that the solution to the Riemann problem is completely self-similar with $\xi=x/t$ as the similarity variable and as such it can be expressed conveniently in one-dimensional ξ-space. In the case of a 1-shock (2-shock) the position of the discontinuity $\xi_{L1}=\xi_{1C}$ ($\xi_{C2}=\xi_{2R}$) is equal to the shock speed σ given by relation (16). Throughout the region covered by a k-rarefaction curve, we know that the solution is constant along characteristic curves $\mathrm{d}x/\mathrm{d}t=\lambda_k$ and that these characteristics are straight lines, i.e. $\mathrm{d}x/\mathrm{d}t=\xi$. Moreover, we know about the constancy of the associated k-Riemann invariant I_R^k. Together we obtain:

$$\begin{array}{rclrcl} u_1(\xi) &=& \frac{2}{3}\xi+\frac{1}{3}(u_L+2c_L)\,, & u_2(\xi) &=& \frac{2}{3}\xi+\frac{1}{3}(u_R-2c_R)\,,\\ h_1(\xi) &=& \frac{1}{9g}(u_L+2c_L-\xi)^2\,, & h_2(\xi) &=& \frac{1}{9g}(\xi-u_R+2c_R)^2\,. \end{array} \tag{45}$$

The zone limits are obtained by calculating the propagation speeds of the bounding characteristics:

$$\begin{aligned} \xi_{L1} &= \lambda_1(\vec{Q}_L) = u_L - c_L \, , & \xi_{C2} &= \lambda_2(\vec{Q}_C) = u_C + c_C \, , \\ \xi_{1C} &= \lambda_1(\vec{Q}_C) = u_C - c_C \, , & \xi_{2R} &= \lambda_2(\vec{Q}_R) = u_R + c_R \, . \end{aligned} \tag{46}$$

This completes the solution of the Riemann problem. An explicit algorithm is given in appendix B. Some examples for different initial data are shown in figure 4.

3 Finite volume method for the numerical solution of the flat-bottom case

3.1 Introduction

The most obvious – and perhaps the most successful – choice for the numerical treatment of our system of equations (1) is the finite volume method. This is because the numerical scheme incorporates the notion of "weak solutions" which include discontinuities. As such, a finite volume method is dealing with certain volume averages of the quantities and with fluxes across cell boundaries rather than with a point-to-point discretization of the differential operators (finite difference method). Hence, the starting point for the discretization is an integral (or weak) formulation of the equations. As in [7] we first integrate our basic one-dimensional, non-viscous relation (3) over a spatial domain (a, b),

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_a^b \vec{Q}(x,t) \, \mathrm{d}x = F(a,t) - F(b,t) \quad , \tag{47}$$

and then in time over $(n \cdot \Delta t, (n+1) \cdot \Delta t)$:

$$\int_a^b U^{n+1}(x) \mathrm{d}x - \int_a^b U^n(x) \mathrm{d}x = -\Delta t \left(\hat{f}(Q(b)) - \hat{f}(Q(a)) \right) \quad . \tag{48}$$

In the above relation (48) – which is exact – $\hat{f}$ corresponds to the time average of the flux during the period of integration. The implications are important: if we suppose some discretization of our spatial domain into finite volumes V_i and define the following spatial average between cell boundaries $(i - 1/2)$ and $(i + 1/2)$,

$$\overline{U}_i = \frac{1}{\Delta x_i} \int_{i-1/2}^{i+1/2} U(x,t) \, \mathrm{d}x \quad , \tag{49}$$

we obtain from (48):

$$\overline{Q}_i^{n+1} - \overline{Q}_i^n = -\Delta t / \Delta x \left(\hat{f}(Q_{i+1/2}) - \hat{f}(Q_{i-1/2}) \right) \quad . \tag{50}$$

It becomes clear that the temporal variation of the cell-averaged values is due to the time integral of the cell-flux difference. This statement of physical conservation can be considered as the foundation for numerical finite volume methods. The main part of the remaining task is to find physically meaningful numerical approximations to the fluxes $\hat{f}$.

3.2 Godunov's scheme

The idea behind Godunov's method (cf. [8]) consists in solving analytically the Riemann problems arising at each cell interface of a cell-wise constant finite-volume scheme (figure). The obtained solution is again cell-averaged before the following time step. As such, Godunov's scheme is a finite volume method

$$\vec{Q}_i^{n+1} - \vec{Q}_i^n = -\frac{\Delta t}{\Delta x_i} \left(\mathcal{F}^G_{i+1/2} - \mathcal{F}^G_{i-1/2} \right) \quad , \tag{51}$$

where the numerical flux function $\mathcal{F}^G$ is the physical flux computed from the exact solution $\vec{Q}^{(R)}$ of the individual Riemann problem, taken at the location of the cell-interface, viz.

$$\mathcal{F}^G_{i+1/2} = F(\vec{Q}^{(R)}(x_{i+1/2})) \quad . \tag{52}$$

In practice, this means that $\vec{Q}^{(R)}$ is the value of the solution of the respective Riemann problem at $\xi = 0$ in local coordinates. For the shallow water equations, this is simply the center state $\vec{Q}_c$ of the interaction (cf. figure 2) which we have calculated in §2.2. With a few additions, the routine given in appendix B can thus be transformed into a Godunov solver (see appendix C).

In Godunov's scheme a restriction of the permissible time step arises from the condition that neighboring Riemann problems do not interact, i.e. that the maximum signal velocities do not cover more than half a cell-dimension during a time interval, viz.

$$\Lambda_i \cdot \Delta t \leq \frac{\Delta x}{2} \qquad \forall \qquad i\,, \qquad \text{where} \qquad \Lambda_i = \max\left(|\lambda_i|, |\sigma_i|\right) \tag{53}$$

and σ_i only makes sense in the presence of a shock in the ith cell.

The Godunov method is overall monotone, i.e. does not give rise to oscillations in the presence of discontinuities, and it is of formal first order accuracy in time and space. Moreover, the scheme has the important property of respecting the entropy condition (cf. §2.2) that guarantees a physically meaningful solution. This fact is not surprising since the essential physics of the underlying system are built into the numerical method.

Why, then, is this method not largely utilized in practice? Basically, it can become quite time-consuming due to the iterative solution of each Riemann problem. Moreover, the great detail of the cell-wise solution is partially lost during the averaging stage so that a simplified approach is often sufficient.

3.3 Roe's scheme

A very pragmatic and successful approach has been taken by Roe [9]. The *exact* solution to the *linearized* Riemann problem

$$\partial_t \vec{Q} + \mathcal{A}(\vec{Q}_L, \vec{Q}_R) \cdot \partial_x \vec{Q} = 0 \quad , \tag{54}$$

is constructed. This solution consists of two simple waves since both characteristic fields are now linearly degenerate (i.e. the relation (12) becomes an equality). The interface flux can be expressed by incrementing across each wave either from the right or from the left state. Blending both formulas leads to the well known flux function:

$$\mathcal{F}^{Roe}_{i+1/2} = \frac{1}{2}\left(\vec{F}(\vec{Q}_L) + \vec{F}(\vec{Q}_R) - |\mathcal{A}(\vec{Q}_L, \vec{Q}_R)| \cdot (\vec{Q}_R - \vec{Q}_L)\right) \quad , \tag{55}$$

where

$$\mathcal{A}(\vec{Q}_L, \vec{Q}_R) = \mathcal{R}(\vec{Q}_L, \vec{Q}_R) \cdot |\Lambda(\vec{Q}_L, \vec{Q}_R)| \cdot \mathcal{R}^{-1}(\vec{Q}_L, \vec{Q}_R) \quad , \tag{56}$$

and $\mathcal{R}, \mathcal{R}^{-1}$ are the diagonalization matrices and Λ the diagonal eigenvalue matrix.

The key point of Roe's scheme is the definition of a sensible linearization (54). The consistency requirements can be put forth as three conditions

$$\begin{array}{ll} \text{(i)} & \mathcal{A}(\vec{Q}_L, \vec{Q}_R) \text{ is hyperbolic and a diagonal form exists.} \\ \text{(ii)} & \mathcal{A}(\vec{Q}_L, \vec{Q}_L) = J(\vec{Q}_L) \\ \text{(iii)} & \mathcal{A}(\vec{Q}_L, \vec{Q}_R)\,[\vec{Q}] = [\vec{F}] \end{array} \qquad . \tag{57}$$

Condition (iii), in particular, assures that the numerical flux (55) is exact in the case of a stationary shock situated in between nodes $()_L$ and $()_R$.

In order to find the matrix $\mathcal{A}$ according to criteria (i)-(iii), we define the following parameter vector

$$\vec{z} = \sqrt{h}\,(1, u)^T \quad , \tag{58}$$

which lets us rewrite the variable vector as

$$\vec{Q}(\vec{z}) = (z_1^2, z_1\, z_2)^T \quad , \tag{59}$$

and the flux vector as

$$\vec{F}(\vec{z}) = (z_1\, z_2, z_2^2 + \frac{g}{2}\, z_1^4)^T \quad . \tag{60}$$

Using the mean value theorem,

$$[a\, b] = \overline{a}\,[b] + \overline{b}\,[a] \quad , \tag{61}$$

the jump of variables can be expressed as $[\vec{Q}] = B\,[\vec{z}]$, where

$$B = \begin{pmatrix} 2\,\overline{z}_1 & 0 \\ \overline{z}_2 & \overline{z}_1 \end{pmatrix} \quad . \tag{62}$$

Analogous, the jump of the fluxes is written as $[\vec{F}] = C\,[\vec{z}]$, where

$$C = \begin{pmatrix} \overline{z}_2 & \overline{z}_1 \\ 2\,g\,\overline{z}_1\,\overline{z}_1^2 & 2\,\overline{z}_2 \end{pmatrix} \quad . \tag{63}$$

Inserting matrix (62) and (63) into condition (iii) gives:

$$\mathcal{A} = C \cdot B^{-1} = \begin{pmatrix} 0 & 1 \\ g\,\overline{z_1^2} - \dfrac{\overline{z}_2^2}{\overline{z}_1^2} & 2\dfrac{\overline{z}_2}{\overline{z}_1} \end{pmatrix} \quad , \tag{64}$$

By comparing (64) with equation (4) we notice that Roe's linear matrix for the shallow water equations is equivalent to the jacobian $\vec{J}$ of the continous system under the following change of variables:

$$\mathcal{A}(\vec{Q}_L, \vec{Q}_R) = \vec{J}(\vec{Q} \,\rightarrow\, \vec{Q}^{Roe}) \quad , \tag{65}$$

where $\vec{Q}^{Roe}$ is the vector of variables averaged as follows:

$$\begin{aligned} \vec{Q}^{Roe} &= (\widetilde{h}, \widetilde{u}\widetilde{h})^T \\ \widetilde{h} &= \overline{h} \\ \widetilde{u} &= \frac{\sqrt{h_L}\, u_L + \sqrt{h_R}\, u_R}{\sqrt{h_L} + \sqrt{h_R}} \quad . \end{aligned} \tag{66}$$

The above result has been obtained by Glaister [10]. The situation is similar in the case of the Euler equations, as noted in the original paper by Roe [9]. For more complex systems. e.g. those arising in the context of two-phase flows [11] or resulting from a statistical turbulence model [12], Roe's linear matrix cannot be obtained from the jacobian by a simple variable transformation. In these latter cases, it is sometimes appropriate to resort to a scheme not relying upon the somewhat restrictive condition (iii) of Roe. One prominent example is the following "VFRoe" scheme.

References

[1] H. Capart and D.L. Young. Formation of a jump by the dam-break wave over a granular bed. *J. Fluid Mech.*, 372:165–187, 1998.

[2] T. Buffard, T. Gallouet, and J.-M. Hérard. Un schéma simple pour les équations de Saint-Venant. *C. R. Acad. Sci. Paris*, t. 324, Série I:385–390, 1998.

[3] J.M. Townson. *Free-surface hydraulics.* Unwin Hyman, 1991.

[4] S.S. Voit. Tsunamis. *Annu. Rev. Fluid Mech.*, 19:217–236, 1987.

[5] J. Smoller. *Shock waves and reaction diffusion equations.* Springer, 1983.

[6] P.D. Lax. *Hyperbolic systems of conservation laws and the mathematical theory of shock waves.* CBMS Monographs. SIAM, Philadelphia, 1973.

[7] C.G. Hirsch. *Numerical computation of internal and external flows.* J. Wiley, 1990.

[8] S. Godunov, A. Zabrodine, M. Ivanov, A. Kraiko, and G. Prokopov. *Résolution numérique des problèmes multidimensionnels de la dynamique des gaz.* Editions MIR, Moscou, 1979.

[9] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[10] P. Glaister. A weak formulation of Roe's approximate Riemann solver applied to the St. Venant equations. *J. Comput. Phys.*, 116:189–191, 1995.

[11] L. Combe and J.-M. Hérard. Un schéma volumes finis pour la simulation d'un modèle bi-fluide d'écoulements diphasiques compressibles gaz-solide. *Revue Européenne des Eléments finis*, 5(2):197–231, 1997.

[12] G. Brun, J.-M. Hérard, D. Jeandel, and M. Uhlmann. An approximate Riemann solver for second-moment closures. *J. Comput. Phys.*, 151(2):990–996, 1999.

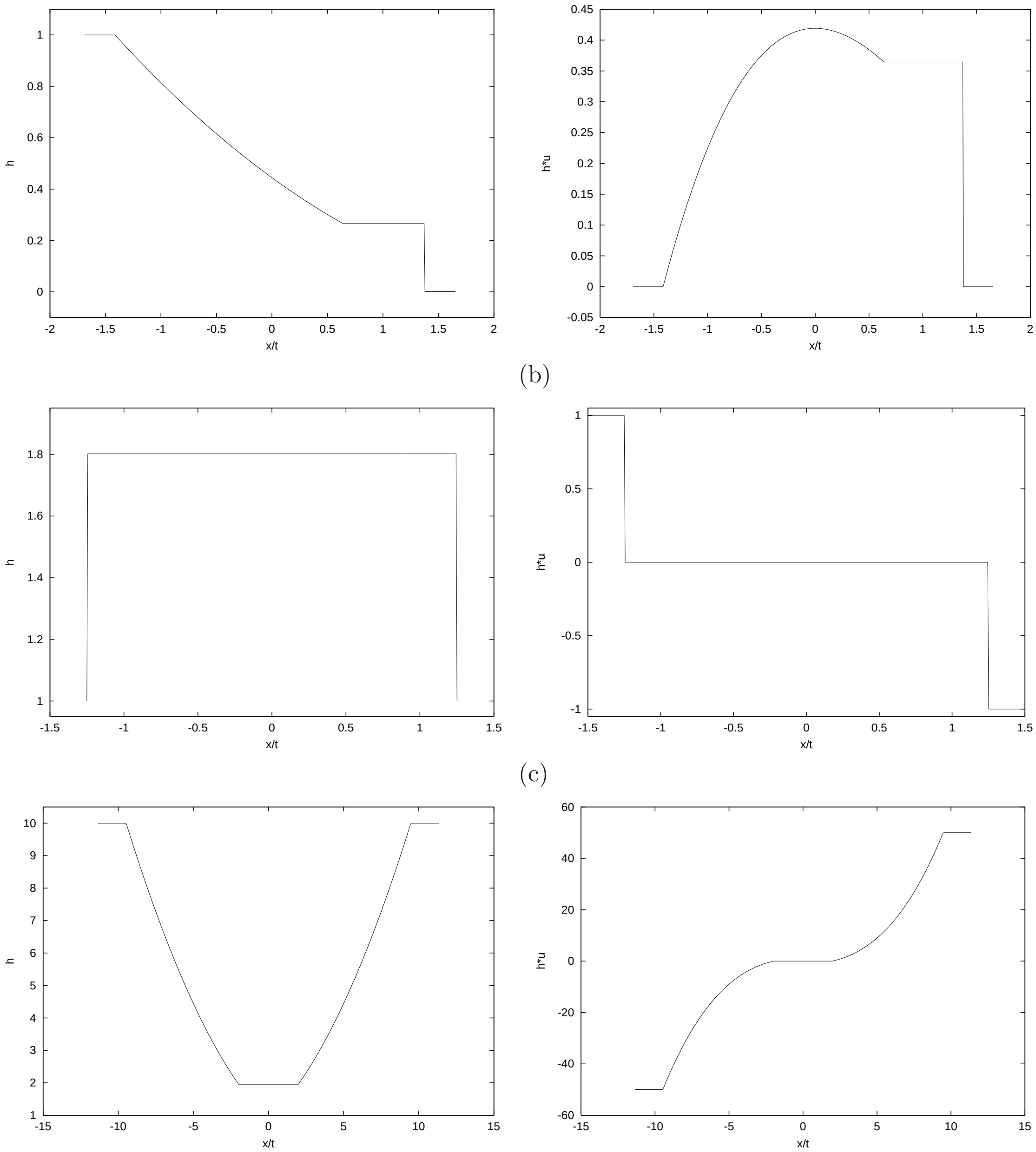
[13] G.B. Whitham. *Linear and nonlinear waves.* Wiley, 1974.

[14] P. Garcia-Navarro, M.E. Hubbard, and A. Priestley. Genuinely multidimensional upwinding for the 2d shallow water equations. *J. Comput. Phys.*, 121:79–93, 1995.

[15] A. Bermudez and M.E. Vazquez. Upwind methods for hyperbolic conservation laws with source terms. *Computers & Fluids*, 23(8):1049–1071, 1994.

[16] J.M. Greenberg and A.Y. Leroux. A well-balanced scheme for the numerical processing of source terms in hyperbolic equations. *SIAM J. Num. Anal.*, 33(1):1–16, 1996.

[width=100



(a)

(b)

(c)

Figure 4: Some example Riemann problems for g=2: (a) $\vec{Q}_L = (1, 0)$, $\vec{Q}_R = (.001, 0)$ – rarefacion/shock; (b) $\vec{Q}_L = (1, 1)$, $\vec{Q}_R = (1, -1)$ – double shock; (c) $\vec{Q}_L = (10, -50)$, $\vec{Q}_R = (10, 50)$ – double rarefaction. The variation of the height variable h and the flow rate hu is shown as a function of $\xi = x/t$. The solution has been obtained using the routine given in appendix B.

[width=100

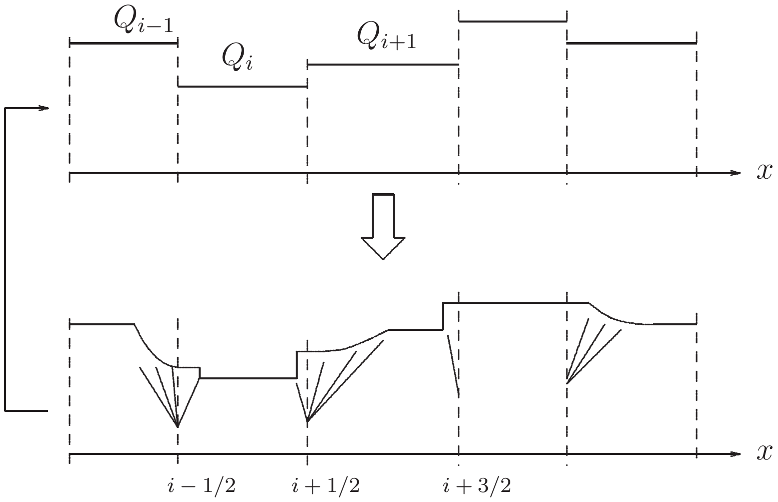


Figure 5: The steps of Godunov's scheme for solving a one-dimensional hyperbolic problem: discretization into cell-wise constant values; exact solution of the ensueing Riemann problems at the cell-interfaces (subject to a CFL-like condition); subsequent averaging over the cells.

A Weak solutions of conservation laws and the jump conditions

The present account follows that of Smoller [p. 246 5]. We consider the initial value problem for a conseration law

$$u_{,t} + \partial_x f(u) = 0\,, \qquad u(x,0) = u_0(x)\,. \tag{67}$$

Let us define a test function $\phi(x,t)$ which vanishes outside of a compact support and particularly on $t = T$, $x = a$ and $x = b$ (cf. figure 6). Multiplication of (67) by $\phi(x,t)$ and integration for $t > 0$ over a domain D gives:

$$\int_{t>0}\int (u_{,t} + f_{,x})\,\phi\,\mathrm{d}x\mathrm{d}t = \int_D\int (u_{,t} + f_{,x})\,\phi\,\mathrm{d}x\mathrm{d}t = \int_a^b\int_0^T (u_{,t} + f_{,x})\,\phi\,\mathrm{d}x\mathrm{d}t = 0\,. \tag{68}$$

Integration by parts of the time derivative in (68)

$$\int_D\int u_{,t}\phi\,\mathrm{d}x\mathrm{d}t = \int_a^b [u\phi]_0^T\,\mathrm{d}x - \int_a^b\int_0^T u\phi_{,t}\,\mathrm{d}x\mathrm{d}t = -\int_a^b u(t=0)\phi(t=0)\mathrm{d}x - \int_D\int u\phi_{,t}\,\mathrm{d}x\mathrm{d}t \tag{69}$$

and of the spatial derivative

$$\int_D\int f_{,x}\phi\,\mathrm{d}x\mathrm{d}t = \int_0^T \underbrace{[f\phi]_a^b}_{=0}\,\mathrm{d}t - \int_D\int f\phi_{,x}\,\mathrm{d}x\mathrm{d}t \tag{70}$$

leads to:

$$\int_D\int (u_{,t} + f_{,x})\,\phi\,\mathrm{d}x\mathrm{d}t + \int_a^b u_0(x)\phi(t=0,x)\,\mathrm{d}x = 0\,. \tag{71}$$

A bounded measurable function $u(x,t)$ is called a *weak solution* of the initial-value problem (67) with bounded and measurable initial data u_0 provided that (71) holds for all $\phi \in C_0^1$. If u is continuous C^1, then (71) describes a classical solution of (67).

Turning now to the jump conditions, let $\Gamma(x,t)$ be a smooth curve across which $u(x,t)$ has a jump discontinuity (i.e. a discontinuity in the zeroth derivative). Consider a domain D around some point P on Γ (cf. figure 6 b); $\phi(x,t)$ is a continuous test function with support centered at P and – as above – zero on the boundaries of D. A weak solution on the domain D satisfies

$$0 = \int_D\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t = \int_{D_1}\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t + \int_{D_2}\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t\,. \tag{72}$$

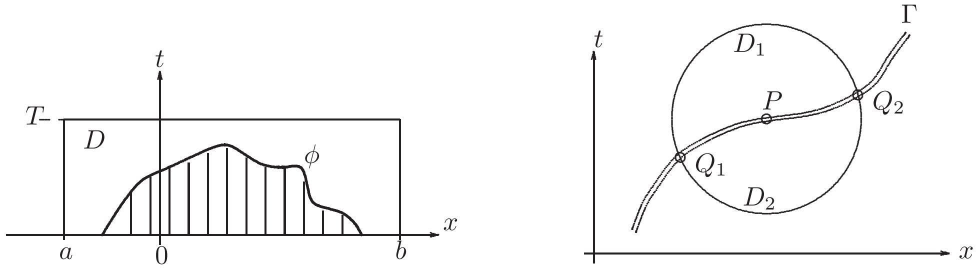


Figure 6: The test function ϕ used during the derivation of the class of weak solutions of system (67) (left) and the notation for the solution of a jump discontinuity around the point P on the curve Γ (right).

Since $u(x,t)$ is continuous on both sides of Γ, i.e. in D_1, D_2 we have:

$$\begin{aligned}
\int_{D_i}\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t &= \int_{D_i}\int ((u\phi)_{,t} + (f\phi)_{,x}) - \left(\phi\,\underbrace{(u_{,t} + f_{,x})}_{=0}\right)\,\mathrm{d}x\mathrm{d}t \\
&= \int_{D_i}\int \nabla\cdot\mathbf{A}\,\mathrm{d}\mathbf{v}\,, \qquad (73)
\end{aligned}$$

where:

$$\begin{aligned}
\mathbf{A} &= (f\phi, u\phi)^T \\
\nabla &= (\partial_x, \partial_t)^T \qquad (74)\\
\mathrm{d}\mathbf{v} &= \mathrm{d}x\,\mathrm{d}t\,. \qquad (75)
\end{aligned}$$

Applying the divergence theorem gives:

$$\begin{aligned}
\int_{D_i}\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t &= \oint_{\partial D_i} \mathbf{A}\cdot\mathbf{n}\mathrm{d}s \\
&= \oint_{\partial D_i} (A_1 n_1 + A_2 n_2)\mathrm{d}s \qquad (76)
\end{aligned}$$

where $\mathbf{n}$ is the outward normal vector along the closed curve ∂D_i. The geometric relation between $\mathrm{d}s$ and the normal vector $\mathbf{n}$ is such that (cf. figure 7):

$$\begin{aligned}
n_1\mathrm{d}s &= \mathrm{d}t \\
n_2\mathrm{d}s &= -\mathrm{d}x\,. \qquad (77)
\end{aligned}$$

Therefore we obtain:

$$\int_{D_i}\int (u\phi_{,t} + f\phi_{,x})\,\mathrm{d}x\mathrm{d}t = \oint_{\partial D_i} \phi(f\,\mathrm{d}t - u\,\mathrm{d}x)\,. \qquad (78)$$

We can now evaluate the r.h.s. of equation (72) by evaluating expression (78) for the two paths on the boundaries ∂D_1 and ∂D_2; we note that—since $\phi = 0$ on ∂D the line integral is non-zero only along Γ. Then we have:

$$\begin{aligned}
0 &= \oint_{\partial D_1} \phi(f\,\mathrm{d}t - u\,\mathrm{d}x) + \oint_{\partial D_2} \phi(f\,\mathrm{d}t - u\,\mathrm{d}x) \\
&= \oint_{Q_1}^{Q_2} \phi(f(u_l)\,\mathrm{d}t - u_l\,\mathrm{d}x) + \oint_{Q_2}^{Q_1} \phi(f(u_r)\,\mathrm{d}t - u_r\,\mathrm{d}x) \\
&= \oint_{Q_1}^{Q_2} \phi(f(u_l)\,\mathrm{d}t - u_l\,\mathrm{d}x) - \oint_{Q_1}^{Q_2} \phi(f(u_r)\,\mathrm{d}t - u_r\,\mathrm{d}x) \\
&= \oint_{Q_1}^{Q_2} \phi([f(u)]\,\mathrm{d}t - [u]\,\mathrm{d}x)\,, \qquad (79)
\end{aligned}$$

where $u_l = u(x(t) - 0, t)$ and $u_r = u(x(t) + 0, t)$ are the values of u on the left and right of the discontinuity, and the jumps are denoted by $[u] = u_l - u_r$ and $[f(u)] = f(u_l) - f(u_r)$.

Since the integral in (79) needs to vanish for any ϕ, one can conclude:

$$\sigma[u] = [f(u)]\,, \qquad (80)$$

where $\sigma = \mathrm{d}x/\mathrm{d}t$ is the speed of the discontinuity.

B Algorithm for analytic solution of flat-bottom Riemann problem

```
      program kneedeep
c----67--------------------------------------------------------flowtec-|
c     solves the riemann problem for st.venant equations (flat bottom)
c     analytically by considering the following zones:
c     (e.g. for a rarefaction-shock configuration)
c
c      (L)     (1)        (C)         (2)        (R)
c      ------
c            \\\\\\                       ----------
c             \\\\\                     |
c              \\\\                     |
c               \\\                     |       ^ state q
c                \\                     |       |
c                 \                     |       |
c                  ---------------------        -----> (x/t)
c----67----------------------------------------------------------------|
c     12.02.99
c     BUGS: -?-
c----67----------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      parameter(npoimx=500)
      dimension x(npoimx),u(npoimx),fh(npoimx)
      logical lvacuum,l1simple,l2simple,lldry,lrdry
      parameter (f23=2.d0/3.d0,f13=1.d0/3.d0,f19=1.d0/9.d0)
      character*20 outfile
      parameter (iunit=12,outfile='riemann.dat')
c----67----------------------------------------------------------------|
      parameter (g=2.,time=1.d0)
      parameter (nnwtmx=20,small=1.d-6,width=1.2d0)
c----67----------------------------------------------------------------|

c     /* set the two initial states L/R of diaphragm    */
      write(*,10)g
      write(*,*)'hL= ?'
      read(*,*)fhL
      write(*,*)'uL= ?'
      read(*,*)uL
      write(*,*)'hR= ?'
      read(*,*)fhR
```

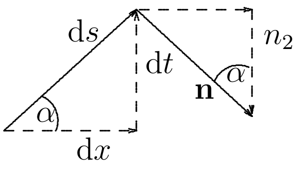


Figure 7: The geometrical relation between the inifinitesimal line segment ds and the outward normal vector **n**.

```
      write(*,*)'uR= ?'
      read(*,*)uR
      cL=dsqrt(g*fhL)
      cR=dsqrt(g*fhR)
      if(fhl.ne.0.d0)then
         if(fhR.eq.0.d0)then
            write(*,*)'dry bed on the right'
            lrdry=.TRUE.
         endif
         B=fhR/fhL
         C=(uR-uL)/cL
      else
         write(*,*)'dry bed on the left'
         lldry=.TRUE.
         B=0.d0
         C=0.d0
      endif

c     /* check for vacuum (i.e. dry bed)               */
      lvacuum=.FALSE.
      if(2.d0*(1.d0+dsqrt(B)).le.C)lvacuum=.TRUE.
      write(*,*)'appearance of vacuum:',lvacuum
      if(lvacuum)write(*,*)'will set velocity arbitrarily to zero'
c     /* check of which type are the 1- and 2-families */
      l1simple=.FALSE.
      l2simple=.FALSE.
      if(B.gt.0.d0)
     $     check1=f1(dlog(B))*dsqrt(B)
      if(check1.lt.C.or.lrdry)l1simple=.TRUE.
      if(B.gt.0.d0)
     $     check2=f1(-dlog(b))
      if(check2.lt.C.or.lldry)l2simple=.TRUE.
      write(*,*)'1-family:'
      if(l1simple)write(*,*)'simple wave'
      if(.not.l1simple.and..not.lldry)write(*,*)'shock'
      write(*,*)'2-family:'
      if(l2simple)write(*,*)'simple wave'
      if(.not.l2simple.and..not.lrdry)write(*,*)'shock'


c     /* solve the 1-family                            */
      if(l1simple)then
         if(lvacuum.or.lrdry)then
            fhC=0.d0
            uc=0.d0
         else
            x1=-2.d0*dlog((1.d0+dsqrt(B))/2.-C/4.d0)
            fhC=fhL*dexp(-x1)
            CC=dsqrt(g*fhc)
            uC=uL+cL*2.d0*(1.d0-dexp(-x1/2.d0))
            write(*,*)'1-rarefaction: hC=',fhC,' uC=',uC
         endif
      else
         if(l2simple)then
```

```
            if(lvacuum.or.lldry)then
               fhC=0.d0
               uc=0.d0
            else
               x2=-2.d0*dlog(-C/(dsqrt(B)*4.d0)+1.d0/(2.d0*dsqrt(B))+
     $              .5d0)
               fhC=fhR*dexp(-x2)
               CC=dsqrt(g*fhc)
               uC=uR-cC*2.d0*(dexp(x2/2.d0)-1.d0)
               write(*,*)'2-rarefaction: hC=',fhC,' uC=',uC
            endif
         else
c     /* we have in fact two shocks: need to iterate              */
            z1=1.1d0
            do n=1,nnwtmx
               zold=z1
               z1=zold-funcz1(zold,B,C)/dfuncz1(zold,B)
               if(dabs(z1-zold).le.small)goto 111
            enddo
            write(*,*)'newton iteration not converged',z1,zold
            stop
 111        continue
            fhC=z1*fhL
            cc=dsqrt(fhc*g)
            uC=uL-cL*(z1-1.d0)*dsqrt((z1+1.d0)/(2.d0*z1))
         endif
      endif
c     /* we now know about the center state (c)                   */

c     /* calculate limits of zone (1) and (2)                      */
c     /* note: "positions" x** are actually given in "xi/t",i.e. velocities */
      if(l1simple)then
         xL1=uL-cL
         if(.not.lvacuum.and..not.lrdry)then
            x1C=uC-cC
         else
c     /* limit zone (1) such that vacuum is just reached through    */
c     /* the expansion */
            x1C=uL+2.d0*cL
         endif
      else
c     /* find shock position: calc shock speed from jump cond.       */
c     /* note: in this case, the zone has zero width                 */
         if(.not.lldry)then
            z=fhC/fhL
            write(*,*)'1-shock: hL/hC=',z
            sigma=(uC*z-uL)/(z-1.d0)
            xl1=sigma
            x1C=sigma
         else
c     /* leave this bound undefined */
         endif
      endif
      if(l2simple)then
```

```
         x2R=uR+cR
         if(.not.lvacuum.and..not.lldry)then
            xC2=uC+cC
         else
            xC2=uR-2.d0*cR
         endif
      else
         if(.not.lrdry)then
            z=fhR/fhC
            write(*,*)'2-shock: hR/hC=',z
            sigma=(uR*z-uC)/(z-1.d0)
            x2R=sigma
            xC2=sigma
         else
c     /* since nothing happens in the dry zone, extend it a bit from (1)(C) */
            x2R=x1C+dabs(.1d0*x1c)
            xC2=x2R
         endif
      endif
c     /* get back to the case of a dry bed on the left */
      if(lldry)then
         x1C=xC2-dabs(.1d0*xC2)
         xl1=x1c
      endif

c     /* set a discrete grid, a bit wider than interesting zone */
      flx=(x2R-xL1)
      foverlap=(width-1.d0)/2.d0
      do i=1,npoimx
         x(i)=(xL1-foverlap*flx)+
     $        dfloat(i-1)*flx*width/dfloat(npoimx-1)
      enddo

c     /* set the variables at grid points                       */
      do i=1,npoimx
         if(x(i).le.xL1)then
            u(i)=uL
            fh(i)=fhl
         elseif(x(i).le.x1C)then
            u(i)=f23*x(i)+f13*(uL+2.d0*cL)
            fh(i)=f19/g*(uL+2.d0*cL-x(i))**2
         elseif(x(i).le.xC2)then
            u(i)=uC
            fh(i)=fhC
         elseif(x(i).le.x2R)then
            u(i)=f23*x(i)+f13*(uR-2.d0*cR)
            fh(i)=f19/g*(x(i)-uR+2.d0*cR)**2
         else
            u(i)=uR
            fh(i)=fhR
         endif
      enddo

c     /* output                                                  */
```

```
      open(iunit,file=outfile)
      write(iunit,112)
      do i=1,npoimx
         if(fh(i).ne.0.d0)then
            froude=u(i)/dsqrt(g*fh(i))
         else
            froude=0.d0
         endif
         write(iunit,*)x(i),x(i)*time,fh(i),u(i),fh(i)*u(i),
     $        dsqrt(fh(i)*g),
     $        u(i)+2.d0*dsqrt(fh(i)*g),u(i)-2.d0*dsqrt(fh(i)*g),
     $        froude
      enddo
      close(iunit)

c     /* format statements */
 10   format('Riemann problem:   (L)   |   (R) ; gravitational ',
     $     'accel. = ',e8.3)
 112  format('#1:x/t 2:x 3:h 4:u 5:h*u 6:c 7:u+2c 8:u-2c 9:Froude')

c     /* finalize                                                  */
      stop
      end
c----67----------------------------------------------------------------|
      double precision function f1(x)
      implicit double precision (a-h,o-z)
c
      if(x.lt.0.d0)then
         f1=-(dexp(-x)-1.d0)*dsqrt((dexp(-x)+1)/(2.d0*dexp(-x)))
      elseif(x.gt.0.d0)then
         f1=2.d0*(1.d0-dexp(-x/2.d0))
      else
         f1=0.d0
      endif
      end
c----67----------------------------------------------------------------|
      double precision function f2(x)
      implicit double precision (a-h,o-z)
c
      if(x.lt.0.d0)then
         f2=(dexp(x)-1.d0)*dsqrt((dexp(x)+1.d0)/(2.d0*dexp(x)))
      elseif(x.gt.0.d0)then
         f2=2.d0*(dexp(x/2.d0)-1.d0)
      else
         f2=0.d0
      endif
      end
c----67----------------------------------------------------------------|
      double precision function funcz1(z,B,C)
      implicit double precision (a-h,o-z)
c
      funcz1= -(z-1.D0)*dsqrt(2.D0)*dsqrt((z+1.D0)/z)/2.D0-
     $     (z/B-1.D0)*dsqrt(2.D0)*dsqrt((z/B+1.D0)/z*B)*
     $     dsqrt(B)/2.D0-C
```

```
c
      end
c----67----------------------------------------------------------------|
      double precision function dfuncz1(z,B)
      implicit double precision (a-h,o-z)
c
      dfuncz1=  -dsqrt(2.D0)*dsqrt((z+1.D0)/z)/2.D0-
     $     (z-1.D0)*dsqrt(2.D0)/dsqrt((z+1.D0)/z)*
     $     (1.D0/z-(z+1.D0)/z**2)/4.D0-1.D0/dsqrt(B)*
     $     dsqrt(2.D0)*dsqrt((z/B+1.D0)/z*B)/2.D0-
     $     (z/B-1.D0)*dsqrt(2.D0)/dsqrt((z/B+1.D0)/z*B)*
     $     dsqrt(B)*(1.D0/z-(z/B+1.D0)/z**2*B)/4.D0
c
      end
c----67----------------------------------------------------------------|
```

C Algorithm for numerical solution of flat-bottom Riemann problem by different finite volume methods of Godunov type

```
      program godunov
c----67--------------------------------------------------------flowtec-|
c     solves the flat-bottom st.venant equations numerically in
c     one dimension (dam-break problem) by a finite-volume
c     method using:
c     ischeme           fluxes
c     1                 godunov  (exact riemann solver)
c     3                 roe      (lin. riemann solver, flux-diff-split)
c     2                 gallouet (lin. riemann solver, var-diff-split)
c----67--------------------------------------------------------flowtec-|
c     12.05.99
c     BUGS: -?-
c----67----------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      parameter(nmax=400,g=2.d0)
      dimension q(nmax,2),rhs(nmax,2),flux(2)
      parameter(cfl=.4d0)
c----67----------------------------------------------------------------|
      parameter(hll=1.d0,ull=0.d0,hrr=5.d-1,urr=0.d0)
      parameter(flx=1.d0,tfin=.1d0,nitmx=50000)
      parameter(ischeme=3)
c----67----------------------------------------------------------------|
      time=0.d0
      nit=0
      dx=flx/dfloat(nmax)
      do i=1,nmax
         rhs(i,1)=0.d0
         rhs(i,2)=0.d0
      enddo
c     /* set the initial state (diaphragm at center)*/
      do i=1,nmax/2
```

```
         q(i,1)=hll
         q(i,2)=ull*hll
      enddo
      do i=nmax/2+1,nmax
         q(i,1)=hrr
         q(i,2)=urr*hrr
      enddo

c     /* main time loop */
 11   continue
      nit=nit+1

         dt=1.d35

c     /* flux balance for each cell-interface */
         do 20 icell=1,nmax-1

c     /* solve interfacial riemann problem analytically */
            if(ischeme.eq.1)then

               call riemann(q(icell,1),q(icell,2)/q(icell,1),
     $              q(icell+1,1),q(icell+1,2)/q(icell+1,1),
     $              g,
     $              fhc,uc,flamx)

            elseif(ischeme.eq.2)then

               call gallouet(q(icell,1),q(icell,2),
     $              q(icell+1,1),q(icell+1,2),
     $              g,
     $              fhc,uc,flamx)

            elseif(ischeme.eq.3)then

               call roeflux(q(icell,1),q(icell,2),
     $              q(icell+1,1),q(icell+1,2),
     $              g,
     $              flux,flamx)
            else
               write(*,*)'small problem here...',ischeme
            endif

c     /* calculate maximum GLOBAL time step */
            dt=dmin1(dt,dx*cfl/flamx)
            if(dt.eq.0.d0.or.flamx.eq.0.d0)then
               write(*,*)'icell =',icell,dx,cfl,flamx
               stop
            endif

c     /* accumulate rhs fluxes */
            if (ischeme.eq.1.or.ischeme.eq.2)then
               do ic=1,2
                  add=dflux(fhc,uc,g,ic)
                  rhs(icell,ic)=rhs(icell,ic)+add
                  rhs(icell+1,ic)=rhs(icell+1,ic)-add
               enddo
            else
               do ic=1,2
                  rhs(icell,ic)=rhs(icell,ic)+flux(ic)
                  rhs(icell+1,ic)=rhs(icell+1,ic)-flux(ic)
               enddo
            endif

 20         continue

            time=time+dt
            write(*,*)'time = ',time, ' dt= ',dt, ' tfin = ',tfin
            if(time.gt.tfin.or.nit.gt.nitmx)then
               call output(q,time,flx,nmax,g)
               goto 10
            endif

c     /* update the variables */
            do ival=2,nmax-1
               do ic=1,2
                  q(ival,ic)=q(ival,ic)-dt/dx*rhs(ival,ic)
                  rhs(ival,ic)=0.d0
               enddo
            enddo

c     /* check for non-physical values */
            do i=1,nmax
               if(q(i,1).lt.0d0)then
                  WRITE(*,*)'negative height: ',i,q(i,1),q(i,2)
                  stop
               endif
            enddo

            goto 11
 10      continue
      stop
      end
c----67---------------------------------------------------------flowtec-|
      subroutine output(q,time,flx,ni,g)
      implicit double precision (a-h,o-z)
      dimension q(ni,2)
      open(12,file='sol.dat')
      write(12,100)time,ni
      do i=1,ni
         write(12,*)dfloat(i-1)/dfloat(ni-1)*flx,
     $        q(i,1),q(i,2)/q(i,1),q(i,2),sqrt(g*q(i,1))
      enddo
      close(12)
 100  format('# 1:x 2:h 3:u 4:h*u 5:c')
      return
      end
c----67---------------------------------------------------------flowtec-|
      subroutine riemann(fhl,ul,fhr,ur,g,fhc,uc,flamx)
```

```
c----67--------------------------------------------------------flowtec-|
c     solves the riemann problem for st.venant equations (flat bottom)
c     analytically by considering the following zones:
c     (e.g. for a rarefaction-shock configuration)
c
c      (L)     (1)        (C)         (2)        (R)
c      ------
c            \\\\\\                       ----------
c             \\\\\                     |
c              \\\\                     |
c               \\\                     |       ^ state q
c                \\                     |       |
c                 \                     |       |
c                  ---------------------        -----> (x/t)
c----67----------------------------------------------------------------|
c     12.02.99
c     BUGS: -?-
c----67----------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      logical lvacuum,l1simple,l2simple,lldry,lrdry
      parameter (f23=2.d0/3.d0,f13=1.d0/3.d0,f19=1.d0/9.d0)
c----67----------------------------------------------------------------|
c      parameter (g=2.,time=1.d0,x0=2.5d0)
      parameter (nnwtmx=20,small=1.d-6,width=1.2d0)
c----67----------------------------------------------------------------|
      lrdry=.FALSE.
      lldry=.FALSE.
      cL=dsqrt(g*fhL)
      cR=dsqrt(g*fhR)
      if(fhl.ne.0.d0)then
         if(fhR.eq.0.d0)then
c            write(*,*)'dry bed on the right'
            lrdry=.TRUE.
         endif
         B=fhR/fhL
         C=(uR-uL)/cL
      else
c         write(*,*)'dry bed on the left'
         lldry=.TRUE.
         B=0.d0
         C=0.d0
      endif
c     /* check for equal states on left and right */
      if(fhr.eq.fhl.and.ur.eq.ul)then
         fhc=fhr
         uc=ur
         flamx=dmax1(dabs(ur+sqrt(g*fhr)),dabs(ur-sqrt(g*fhr)),
     $        dabs(ul+sqrt(g*fhl)),dabs(ul-sqrt(g*fhl)))
         goto 112
      endif

c     /* check for vacuum (i.e. dry bed)               */
      lvacuum=.FALSE.
      if(2.d0*(1.d0+dsqrt(B)).le.C)lvacuum=.TRUE.
```

```
c      write(*,*)'appearance of vacuum:',lvacuum
c      if(lvacuum)write(*,*)'will set velocity arbitrarily to zero'
c     /* check of which type are the 1- and 2-families */
      l1simple=.FALSE.
      l2simple=.FALSE.
      if(B.gt.0.d0)
     $     check1=f1(dlog(B))*dsqrt(B)
      if(check1.lt.C.or.lrdry)l1simple=.TRUE.
      if(B.gt.0.d0)
     $     check2=f1(-dlog(b))
      if(check2.lt.C.or.lldry)l2simple=.TRUE.
c      write(*,*)fhl,ul,fhr,ur
c      write(*,*)'1-family:'
c      if(l1simple)write(*,*)'simple wave'
c      if(.not.l1simple.and..not.lldry)write(*,*)'shock'
c      write(*,*)'2-family:'
c      if(l2simple)write(*,*)'simple wave'
c      if(.not.l2simple.and..not.lrdry)write(*,*)'shock'

c     /* solve the 1-family                             */
      if(l1simple)then
         if(lvacuum.or.lrdry)then
            fhC=0.d0
            uc=0.d0
         else
            x1=-2.d0*dlog((1.d0+dsqrt(B))/2.-C/4.d0)
            fhC=fhL*dexp(-x1)
            CC=dsqrt(g*fhc)
            uC=uL+cL*2.d0*(1.d0-dexp(-x1/2.d0))
c            write(*,*)'1-rarefaction: hC=',fhC,' uC=',uC
         endif
      else
         if(l2simple)then
            if(lvacuum.or.lldry)then
               fhC=0.d0
               uc=0.d0
            else
               x2=-2.d0*dlog(-C/(dsqrt(B)*4.d0)+1.d0/(2.d0*dsqrt(B))+
     $              .5d0)
               fhC=fhR*dexp(-x2)
               CC=dsqrt(g*fhc)
               uC=uR-cC*2.d0*(dexp(x2/2.d0)-1.d0)
c               write(*,*)'2-rarefaction: hC=',fhC,' uC=',uC
            endif
         else
c     /* we have in fact two shocks: need to iterate               */
            z1=1.1d0
            do n=1,nnwtmx
               zold=z1
               z1=zold-funcz1(zold,B,C)/dfuncz1(zold,B)
               if(dabs(z1-zold).le.small)goto 111
            enddo
            write(*,*)'newton iteration not converged',z1,zold
            stop
```

```
 111        continue
            fhC=z1*fhL
            cc=dsqrt(fhc*g)
            uC=uL-cL*(z1-1.d0)*dsqrt((z1+1.d0)/(2.d0*z1))
         endif
      endif
c     /* we now know about the center state (c)                    */

c     /* calculate limits of zone (1) and (2)                      */
c     /* note: "positions" x** are actually given in "xi/t",i.e. velocities */
      if(l1simple)then
         xL1=uL-cL
         if(.not.lvacuum.and..not.lrdry)then
            x1C=uC-cC
         else
c     /* limit zone (1) such that vacuum is just reached through    */
c     /* the expansion */
            x1C=uL+2.d0*cL
         endif
      else
c     /* find shock position: calc shock speed from jump cond.       */
c     /* note: in this case, the zone has zero width                 */
         if(.not.lldry)then
            z=fhC/fhL
c            write(*,*)'1-shock: hL/hC=',z
            if(dabs(z-1.d0).gt.small)then
               sigma=(uC*z-uL)/(z-1.d0)
            else
               sigma=ul-sqrt(g*fhl)
            endif
            xl1=sigma
            x1C=sigma
         else
c     /* leave this bound undefined */
         endif
      endif
      if(l2simple)then
         x2R=uR+cR
         if(.not.lvacuum.and..not.lldry)then
            xC2=uC+cC
         else
            xC2=uR-2.d0*cR
         endif
      else
         if(.not.lrdry)then
            z=fhR/fhC
c            write(*,*)'2-shock: hR/hC=',z
            if(dabs(z-1.d0).gt.small)then
               sigma=(uR*z-uC)/(z-1.d0)
            else
               sigma=ur+sqrt(g*fhr)
            endif
            x2R=sigma
            xC2=sigma
         else
c     /* since nothing happens in the dry zone, extend it a bit from (1)(C) */
            x2R=x1C+dabs(.1d0*x1c)
            xC2=x2R
         endif
      endif
c     /* get back to the case of a dry bed on the left */
      if(lldry)then
         x1C=xC2-dabs(.1d0*xC2)
         xl1=x1c
      endif
c     /* compute maximum wave speed from limiting velocities xL1,x2R */
      flamx=dmax1(dabs(xL1),dabs(x2R))

c     /* finalize                                                 */
 112  continue
      return
      end
c----67--------------------------------------------------------flowtec-|
      subroutine gallouet(fhl,ql,fhr,qr,g,fhc,uc,flamx)
c----67--------------------------------------------------------flowtec-|
c     solves the riemann problem for st.venant equations (flat bottom)
c     in its linearized form, cf Buffard, Gallouet, Herard, CRAcadSci,
c     t.326,Serie I,p.385-390, 1998
c----67----------------------------------------------------------------|
c     12.02.99
c     BUGS: -?-
c----67----------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      logical lvacuum
      parameter (f23=2.d0/3.d0,f13=1.d0/3.d0,f19=1.d0/9.d0)
      parameter (small=1.d-6)
c----67----------------------------------------------------------------|
      if(fhl.ne.0.d0)then
         ul=ql/fhl
      else
         ul=0.d0
      endif
      if(fhr.ne.0.d0)then
         ur=qr/fhr
      else
         ur=0.d0
      endif
      cl=dsqrt(g*fhl)
      cr=dsqrt(g*fhr)
      um=(ul+ur)/2.d0
      fhm=(fhr+fhl)/2.d0
      cm=(cl+cr)/2.d0

c     /* check for existence of dry bottom */
      if((ul-ur).ge.2.d0*(cl+cr))then
         lvacuum=.true.
         uc=0.d0
         cc=0.d0
         flamx=small
         goto 111
      endif

c     /* intermediate state in non-vacuum case */
      uc=(um-(cr-cl))
      cc=(cr+cl)/4.d0*(2.d0-(ur-ul)/(cr+cl))
      flamx=dmax1(dabs(um+cm),dabs(um-cm))

 111  continue
      fhc=cc*cc/g
c
      return
      end
c----67----------------------------------------------------------------|
      subroutine roeflux(fhl,ul,fhr,ur,g,rflux,flamx)
c----67--------------------------------------------------------flowtec-|
c     solves the linearized riemann problem for st.venant equations
c     (flat bottom) by using Roe's method (J.Comp.Physics, vol.43,1981)
c     * fluxes are passed upon exit in vector rflux(1:2)
c----67----------------------------------------------------------------|
c     21.05.99
c     BUGS: -?-
c----67----------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      dimension a(2,2),cflux(2),uflux(2),diffv(2),rflux(2)
      logical lvacuum
      parameter (f23=2.d0/3.d0,f13=1.d0/3.d0,f19=1.d0/9.d0)
      parameter (small=1.d-6)
c----67----------------------------------------------------------------|
      cl=dsqrt(g*fhl)
      cr=dsqrt(g*fhr)

c     /* roe's average for shallow water */
      fhc=(fhl+fhr)/2.d0
      uc=(dsqrt(fhr)*ur+dsqrt(fhl)*ul)/(dsqrt(fhr)+dsqrt(fhl))
      cc=dsqrt(g*fhc)

c     /* diagonalization |A|=R*|lambda|*R1 */
      call aroe(fhc,uc,cc,a)

c     /* variable difference vector */
      diffv(1)=fhr-fhl
      diffv(2)=fhr*ur-fhl*ul

c     /* centered flux */
      do i=1,2
         cflux(i)=dflux(fhr,ur,g,i)+dflux(fhl,ul,g,i)
      enddo

c     /* evaluate roe's flux: f=(fl+fr-|a|*(qr-ql))/2 */
      do i=1,2
         rflux(i)=cflux(i)
         do j=1,2
```

```
            rflux(i)=rflux(i)-a(i,j)*diffv(j)
         enddo
         rflux(i)=rflux(i)/2.d0
      enddo

c     /* estimate the maximum signal velocity */
      flamx=dmax1(dabs(uc+cc),dabs(uc-cc))
c
      return
      end
c----67---------------------------------------------------------------|
      subroutine aroe(hc,uc,cc,a)
c----67-------------------------------------------------------flowtec-|
c     calculates roe's matrix
c----67---------------------------------------------------------------|
      implicit double precision (a-h,o-z)
      dimension a(2,2),r(2,2),r1(2,2),flambda(2)
c----67---------------------------------------------------------------|
c     /* diagonalization matrix r */
      r(1,1)=1.d0
      r(1,2)=1.d0
      r(2,1)=uc-cc
      r(2,2)=uc+cc

c     /* inverse r1 */
      r1(1,1)=(uc+cc)/(2.d0*cc)
      r1(1,2)=-1.d0/(2.d0*cc)
      r1(2,1)=(cc-uc)/(2.d0*cc)
      r1(2,2)=1.d0/(2.d0*cc)

c     /* eigenvalues */
      flambda(1)=uc-cc
      flambda(2)=uc+cc

c     /* matrix*diag*matrix multiply */
      do i=1,2
         do j=1,2
            a(i,j)=0.d0
            do k=1,2
               a(i,j)=a(i,j)+r(i,k)*dabs(flambda(k))*r1(k,j)
            enddo
         enddo
      enddo
c
      return
      end
c----67---------------------------------------------------------------|
      double precision function f1(x)
      implicit double precision (a-h,o-z)
c
      if(x.lt.0.d0)then
         f1=-(dexp(-x)-1.d0)*dsqrt((dexp(-x)+1)/(2.d0*dexp(-x)))
      elseif(x.gt.0.d0)then
         f1=2.d0*(1.d0-dexp(-x/2.d0))
      else
         f1=0.d0
      endif
      end
c----67--------------------------------------------------------------|
      double precision function f2(x)
      implicit double precision (a-h,o-z)
c
      if(x.lt.0.d0)then
         f2=(dexp(x)-1.d0)*dsqrt((dexp(x)+1.d0)/(2.d0*dexp(x)))
      elseif(x.gt.0.d0)then
         f2=2.d0*(dexp(x/2.d0)-1.d0)
      else
         f2=0.d0
      endif
      end
c----67--------------------------------------------------------------|
      double precision function funcz1(z,B,C)
      implicit double precision (a-h,o-z)
c
      funcz1= -(z-1.D0)*dsqrt(2.D0)*dsqrt((z+1.D0)/z)/2.D0-
     $     (z/B-1.D0)*dsqrt(2.D0)*dsqrt((z/B+1.D0)/z*B)*
     $     dsqrt(B)/2.D0-C
c
      end
c----67--------------------------------------------------------------|
      double precision function dfuncz1(z,B)
      implicit double precision (a-h,o-z)
c
      dfuncz1=  -dsqrt(2.D0)*dsqrt((z+1.D0)/z)/2.D0-
     $     (z-1.D0)*dsqrt(2.D0)/dsqrt((z+1.D0)/z)*
     $     (1.D0/z-(z+1.D0)/z**2)/4.D0-1.D0/dsqrt(B)*
     $     dsqrt(2.D0)*dsqrt((z/B+1.D0)/z*B)/2.D0-
     $     (z/B-1.D0)*dsqrt(2.D0)/dsqrt((z/B+1.D0)/z*B)*
     $     dsqrt(B)*(1.D0/z-(z/B+1.D0)/z**2*B)/4.D0
c
      end
c----67--------------------------------------------------------------|
      double precision function dflux(h,u,g,i)
      implicit double precision (a-h,o-z)
c
      if (i.eq.1)then
         dflux=h*u
      elseif(i.eq.2)then
         dflux=u*u*h+g*h*h/2.d0
      else
         write(*,*)'problem here:..',i
         stop
      endif
c
      end
c----67--------------------------------------------------------------|
```

Index

Contents