Chapter 6
Hydraulic Modeling Development and Application in Water Resources Engineering

Francisco J.M. Simões

Contents

Abstract The use of modeling has become widespread in water resources engineering and science to study rivers, lakes, estuaries, and coastal regions. For example, computer models are commonly used to forecast anthropogenic effects on the environment, and to help provide advanced mitigation measures against catastrophic events such as natural and dam-break floods. Linking hydraulic models to vegetation and habitat models has expanded their use in multidisciplinary applications to the riparian corridor. Implementation of these models in software packages on personal desktop computers has made them accessible to the general engineering community, and their use has been popularized by the need of minimal training due to intuitive graphical user interface front ends. Models are, however, complex and nontrivial, to the extent that even common terminology is sometimes ambiguous and often applied incorrectly. In fact, many efforts are currently under way in order

F. J.M. Simões (✉)
US Geological Survey Geomorphology and Sediment Transport Laboratory, 4620 Technology Drive, Suite 400, Golden, CO 80403, USA

C. T. Yang, L. K. Wang (eds.), *Advances in Water Resources Engineering*,
Handbook of Environmental Engineering, Volume 14, DOI 10.1007/978-3-319-11023-3_6

to standardize terminology and offer guidelines for good practice, but none has yet reached unanimous acceptance. This chapter provides a view of the elements involved in modeling surface flows for the application in environmental water resources engineering. It presents the concepts and steps necessary for rational model development and use by starting with the exploration of the ideas involved in defining a model. Tangible form of those ideas is provided by the development of a mathematical and corresponding numerical hydraulic model, which is given with a substantial amount of detail. The issues of model deployment in a practical and productive work environment are also addressed. The chapter ends by presenting a few model applications highlighting the need for good quality control in model validation.

Keywords Water resources · Hydraulics · Modeling · Shallow-water equations · Numerical model · Graphical user interface

Nomenclature

$\mathbf{A}$	Jacobian matrix (–)
C	Chezy's roughness coefficient ($L^{1/2}/T$)
C_D	Wind drag coefficient (dimensionless)
C_f	Skin friction coefficient (dimensionless)
D_i	Diffusion coefficient in the ith Cartesian direction ($i=1, 2$) (L^2/T)
$\mathbf{E}$	Matrix containing the inviscid terms (–)
f	Coriolis parameter (1/T)
Fr	Froude number (dimensionless)
F_x, F_y	Components of the body forces in the Cartesian directions, same as F_i ($i=1, 2, 3$) (L^2/T^2)
g	Acceleration due to gravity (L/T^2)
h	Water depth (L)
h_{dry}, h_{wet}	Threshold for drying and wetting of computational cells (L)
h_{eff}	Effective water depth (L)
l	Length of control volume edge (L)
n	Manning's roughness coefficient ($T/L^{1/3}$)
n_x, n_y	Components of the unit normal vector to control volume edges (dimensionless)
p	Pressure ($M/(LT^2)$)
q_x, q_y	Unit discharge in the Cartesian directions (L^2/T)
$\vec{r}$	Position vector (L)
R, $\overline{R}$	Residual and its average, respectively (–)
$\mathbf{R}$	Matrix containing the viscous terms (–)
$\mathbf{S}$	Matrix containing source/sink terms (–)
$S0_i$	Component of the bed gradient in the ith Cartesian direction ($i=1, 2$) (dimensionless)
t	Time (T)
u, v, w	Components of the velocity vector in Cartesian coordinates, same as u_i ($i=1, 2, 3$) (L/T)

U, V	Components of the depth-averaged velocity vector in Cartesian coordinates (L/T)
u_w, v_w	Components of the wind vector in Cartesian coordinates (L/T)
u_*	Magnitude of the shear velocity (L/T)
x, y, z	Orthogonal Cartesian coordinate directions (L)
z_b	Elevation of channel bed above datum (L)
α_{ij}, β_i	Coefficients of the SSPRK scheme (dimensionless)
δ_e	A very small number (L/T)
δ_{ij}	Kronecker delta (dimensionless)
δ_w	Parameter used in dry computational cells (L)
Δt	Time step size (T)
ε_t	Parameter in eddy viscosity formula (dimensionless)
Φ	Limiter in the MUSCL reconstruction (dimensionless)
η	Water-surface elevation above datum (L)
λ_i	Eigenvalues of the Jacobian matrix (–)
ν	Kinematic molecular viscosity (L^2/T)
ν_t	Turbulent eddy viscosity (L^2/T)
θ	CFL coefficient (dimensionless)
θ_c	Flume contraction angle (dimensionless)
θ_s	Angle of the shock (dimensionless)
ρ, ρ_0	Density of water (M/L^3)
ρ_a	Density of air (M/L^3)
Ω_i	Area of control volume i (L^2)
τ_{bx}, τ_{by}	Components of the bed shear stress vector in the Cartesian directions ($M/(LT^2)$)
τ_{ij}	Radiation stresses (L^2/T^2)
τ_{wx}, τ_{wy}	Components of the wind stress at the free surface ($M/(LT^2)$)

1 Introduction

The study of natural river changes and the interference of man in natural water bodies is a difficult but important activity, as increasing and shifting populations place more demands on the natural sources of freshwater, and the impacts of a potential climate change bring new unknowns and renewed urgencies. Although the basic mechanical principles for these studies are well established, the complexity of the fluvial system involves not only the river corridor proper but also the surrounding geographic complex of hillslopes and floodplains, and their respective physiography (e.g., soil composition and properties, vegetation cover, etc.). To understand the fluvial system as a whole, therefore, one must understand not only how the individual parts of the system behave in isolation but also how they interact with each other. These interactions and feedbacks, which may be deterministic or random, can be very complex and extend over wide scales in time and space. Table 6.1 presents

Table 6.1 Synopsis of the variables involved in modeling fluvial systems, showing the complex dependencies between the different forcing phenomena and concomitant subsystem adjustments

	Scale	Variables	Dependencies
Independent variables	Global	Climate	
		Geology	
	Basin	Physiography	
		Vegetation	Climate, geology
		Soil type	Climate, geology
		Land use	Vegetation, soil type, biophysical agents
	Channel	Valley slope	Basin physiography
		Flow discharge	Basin physiography, climate
		Sediment load	Basin physiography, climate, vegetation, basin soil type, land use, bank material
		Bank material composition	Geology, vegetation, soil, land use
Dependent variables	Basin	Meander wavelength and sinuosity	Bank material, discharge, sediment load, channel width and depth
	Channel and subchannel	Bed slope	Valley slope, flow discharge, sediment load, sediment transport rate, bed material size
		Width and depth	Flow discharge, flow velocity, sediment load, bed material size, bank material composition
		Flow velocity	Discharge, channel width and depth, frictional resistance
		Frictional resistance	Bedform geometry, channel width and depth, bed material size
		Bedform geometry	Sediment transport rate, channel width and depth
		Sediment transport rate	Sediment load, flow velocity and turbulence, bed material size, channel slope
		Bed material size	Sediment load, sediment transport rate

a synoptic view of scales, variables, and processes involved in fluvial hydrology, which helps to paint a view of the dynamic dependencies of the morphology of natural river channels. Modeling is one of the tools available to study the behavior of environmental systems, and has become a crucial vehicle to represent and understand—and even communicate—the effects of the interactions of real-world variables in the fluvial corridor.

What is a model? A common definition is that a model is an abstraction of reality [1]. In other words, it is the methodical organization of data and knowledge—as well as assumptions—about the specific system of interest. Modeling is, therefore, a creative process that allows a systematic analysis of a problem. It can be as simple as a schematic diagram sketched in the back of an envelope, or as complex as those used in large supercomputers for weather forecasting, containing millions of lines of code and using detailed data at the planetary scale. Indeed, there are many types of models, of which computer models are a subset. Programmable computers offer a powerful tool to implement physically and mathematically sophisticated models, which nowadays are used routinely in aeronautics, acoustics, medicine, astrophysics, etc.

In hydrology and fluvial hydraulics, the use of computer models has increased significantly in the past couple of decades, due to the availability of affordable computer power, and due to advancements in large-scale data collection techniques, such as remote sensing using laser scanning. The availability of data to build, calibrate, and test a model has also become less of a limiting factor in the application and development of hydrological models. These factors have led to the proliferation of numerous software packages that are increasingly used by the nonmodeling community (i.e., by those that use models, but that do not develop them), which has brought an increased potential for misuse and misunderstanding in the capabilities of the models and in the interpretation of their results. Models are, after all, always limited by the scales, purposes, and assumptions used in their development and, consequently, always have limited application. This chapter illustrates the difference between modeling and software development by following the steps necessary to go from conceptual modeling all the way to computer software implementation. The next section presents a synoptic view of the steps involved in the development of a hydraulic model, which is followed by the detailed description of the development of a model to simulate free surface flows in rivers, reservoirs, and estuaries. Finally, this chapter concludes with the examples of application of the model to laboratory and to real-world problems.

2 Concepts and Development

The terms *model, numerical model,* and *modeling software* are often used interchangeably to mean the same thing. This, however, constitutes an imprecise, even incorrect use of the terms. To understand the difference between these terms, we must look at the whole discipline of modeling. A model is a representation of a portion of the physical world, i.e., it is an abstraction of it. This is the top level in modeling: It is the most general use of the term *model* which, in this sense, means the knowledge we have of the physical phenomena of interest, as is represented by the laws of physics and by data. Oftentimes, there are gaps in the knowledge of the system, caused by lack of data or by incomplete understanding of its physical nature, or both. These gaps must be sated with consistent simplifications, assumptions, and hypothesis. In fluvial hydraulics, a model is often used to represent a segment of a river, a lake, or an estuary, and the basic laws of Newtonian fluid dynamics are employed, which describe the behavior of fluids, flow turbulence, sediment transport mechanisms, mixing processes, etc. The data consist of boundary conditions, such as bathymetry, water discharges, sediment particle-size distributions, vegetation types, sizes and locations, etc. Common assumptions used include flow resistance laws, sediment transport capacity relations, linear dependency of turbulent diffusion on bed shear, etc.

The next level of modeling involves the conceptualization of the physical phenomena into its mathematical formulation, which results in the mathematical model. Stepping from the physical phenomena to the mathematical model involves a simplification, whereby the complexity of natural environments is reduced to a limited set of relations between the dependent and the independent variables[1]. This is accomplished by a process of successive elimination, where the relative importance of the different phenomena is compared and those with the smallest influence are eliminated—or alternatively by starting with an oversimplified view and adding those with the largest influence. Examples of simplifying approximations are steady versus unsteady and one- versus two- versus three-dimensional (3D) formulations; simplifying descriptions of turbulence; etc. This process depends on the modeling objectives, on our knowledge of the physical system, on the availability and quality of data, and even on the perception of the modeler(s). The mathematical model is, therefore, composed by a number of mathematical formulae (governing equations) that must be solved under particular sets of initial and/or boundary conditions.

In fluvial hydraulics, one usually arrives to the setup of a boundary-value problem whose governing equations contain partial differential equations with nonlinear terms. These are based on the 3D Navier–Stokes equations—for example, see [2] for the derivation and presentation of the equations—or in a simplification thereof. Useful and frequent simplifications of the governing equations are the assumption of hydrostatic pressure and the reduction of the order of the governing equations, whereby the 3D flow description is simplified to two-dimensional (2D) or one-dimensional (1D) descriptions by the use of averaging operations. Some models, however, do not use the flow equations. One alternative is provided by artificial neural network techniques, a numerical modeling technique that attempts to reproduce relations between sets of input and output data by learning from observing experimental and/or prototype tests—see [3], for example. Another alternative originated in artificial intelligence is the use of genetic programming, which is another machine-learning technique based on a set of instructions and a fitness function to measure how well a task is performed by a computer. In Ref. [4], this technique is applied to predict flow discharge in compound channels.

The processes employed to solve the governing equations result in the numerical model, which is the next level of modeling. This step is necessary because computers can only perform the most basic arithmetic operations, therefore, partial differential equations must be transformed into the simple algebraic equations that can be programmed in a computer. Furthermore, computers can only execute computations between finite numbers with a limited amount of correct digits and within a certain range. This operation has two important consequences: First, it transforms continuous equations into discrete ones and second, it transforms infinitesimal operators into finite ones. Consequently, instead of solving the original differential equations, computers only solve approximations thereof. The numerical model is, therefore, a set of algebraic equations whose solutions (hopefully) mimic those of the original differential equations.

[1] The independent variables are the inputs and the forcing quantities, such as boundary conditions and spatial coordinates; the dependent variables are the outputs and effects, such as flow velocity and water surface elevation.

The numerical model can be very simple—the direct solution of the Manning's equation on the back of an envelope, for example—or very complex. The latter often arises from the need to solve partial differential equations that have no known closed-form analytic solutions, and which require complex numerical techniques and algorithms to solve. Finite differences, finite volumes, and finite elements are some of the methodologies available and commonly used in numerical modeling, and detailed description of these can be found in basic computational fluid dynamics (CFD) textbooks, such as in Ref. [5]. Because these techniques and algorithms are approximate and do not provide the exact solution to the equations, the quality of the approximation depends on several factors, such as order of discretization, approaches to deal with nonlinear terms, discretization refinement (more about this later), choice of numerical parameters, boundary conditions, and even the experience of the modeler.

The next level of modeling involves the implementation of the numerical model in a computer, and it is called the computer model. The computer model is the result of coding the numerical model in a computer language and is, therefore, software. Often, the terms *numerical model* and *computer model* are used interchangeably, but that is not correct: The same numerical model can be coded differently by different programmers and, therefore, yield different results. This may be due to the order in which algebraic operations are implemented, resulting in different truncation errors, or it may be due to the use of numerical constants of different resolutions (say, by hard-coding $\pi = 1.1415962$ instead of using a function such as 4 tg^{-1} 1). Compilers may use different libraries and/or different optimization options. In fact, the mathematical associative property does not always hold for the Institute of Electrical and Electronics Engineers (IEEE) floating-point representation of numbers by a computer: $(a+b)+c$ may not be equal to $a+(b+c)$. For example, $2^{-63}+1–1$ is equal to 2^{-63}, obviously. In a computer using double-precision floating-point numbers, however, there are two possible answers: One is $(2^{-63}+1)+(-1) \approx 1+(-1)=0$, because $(2^{-63}+1)$ rounds to 1; the other, through a modification in the order of operations, is $2^{-63}+(1+(-1)) \approx 2^{-63}+0=2^{-63}$. Both the results are correct from the computer point of view and, in such a case, the outcome depends not only on how the lines of code are written by the programmer but also on how the particular compiler used performs code optimization [6]. Different operating systems and different hardware configurations may also be the cause for numerical differences among software. And there are coding errors, something that can easily to creep into modern computer models containing thousands of lines of code.

The different steps in modeling are summarized in Fig. 6.1, where the boxes to the left represent the levels of models described above and the ones to the right represent some of the factors influencing the outcome when going from one level to the next. It is easy to see that the different levels, or types, of models have different objectives. For example, a physical model may be built to gain knowledge about a system, while computer software is implemented to perform calculations—i.e., to provide a service. In fact, the final step in modeling involves the use of the computer model (software) by an operator. Software operators, or users, often are distinct from code developers and have different objectives: While code developers are interested in implementing a computer model of some conceptualized reality,

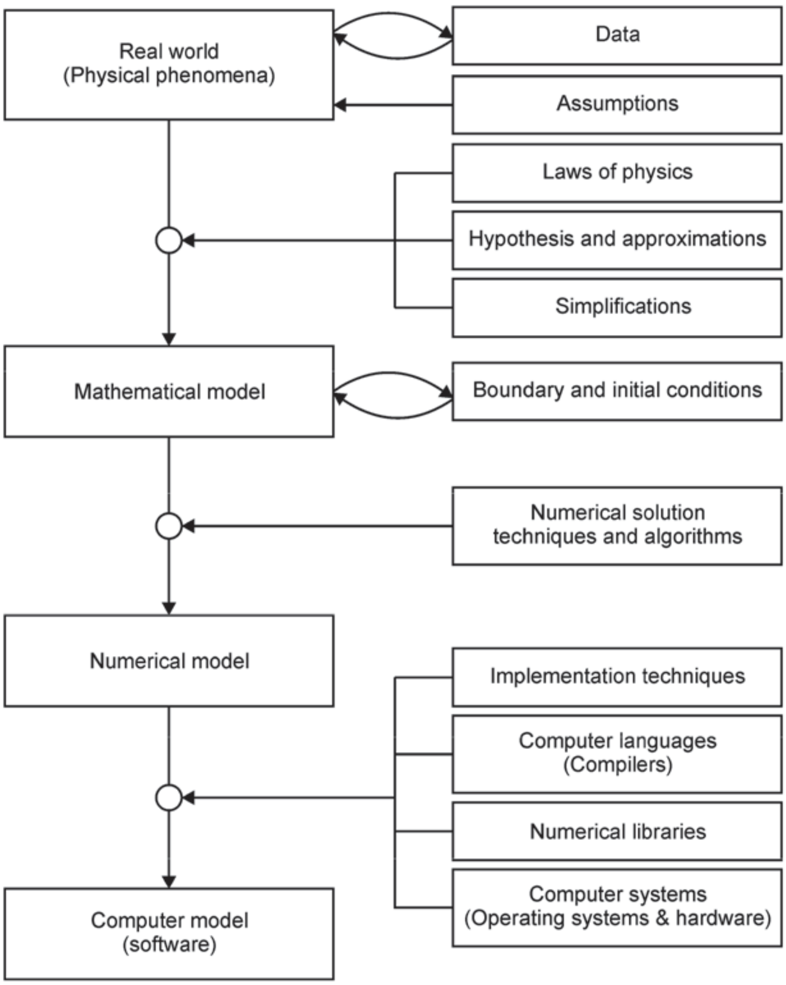
Real world
(Physical phenomena)
Data
Assumptions
Laws of physics
Hypothesis and approximations
Simplifications
Mathematical model
Boundary and initial conditions
Numerical solution techniques and algorithms
Numerical model
Implementation techniques
Computer languages (Compilers)
Numerical libraries
Computer systems (Operating systems & hardware)
Computer model (software)

Fig. 6.1 The different modeling levels and the factors contributing to each level change

software users are usually seeking answers to management problems. This raises the issues of communication between these two groups of researchers, including documentation, terminology, best-practice guidelines, and quality assurance.

Detailed and clear model documentation should present the elements of the conceptual model and its purpose, and must use a terminology that is unambiguous and understood by all. Current practices vary widely, but efforts exist to unify and standardize terminology and procedure. Detailed terminology is proposed by [7], who also review a number of modeling guidelines and present key views of the scientific community. A comprehensive set of guidelines for model development, model evaluation, and model application is given by the US Environmental Protection Agency's Council for Regulatory Environmental Modeling [8]. Nonetheless, it is still difficult to take advantage of the structural complexity of a model, and many approximations and simplifications are introduced in this stage. Model calibration is usually achieved using time series; therefore, finding the best set of model parameters is limited by available data. An incomplete knowledge of the physical system is also common, such as imprecise bathymetry, unknown water and/or sediment discharges, friction factors, etc. Finally, introducing additional human operators in the system also raises the issues of training (i.e., the results of the simulation project depend on operator skill) and increase the likelihood of human error.

3 Developing a Surface-Water Model for Environmental Flows

One of the most important considerations when developing (and applying) a model is model purpose. In scientific and engineering applications to human and natural environmental systems and interactions, the most important modeling purposes are exploratory analysis and prognostication: While prognostication refers to the ability of a model to predict the systems' response to specific external forcing factors, exploratory analysis is concerned with mapping the space of possible development trajectories of a system, usually with the purpose to explore the possibility of unexpected behaviors and thresholds triggering abrupt change. Closely replicating physical phenomena is, therefore, the major goal when developing a hydraulic model for environmental flows.

CFD methods have been in use for over 90 years, with the first application having been developed for numerical weather prediction by L. Richardson in 1922 and published in a book titled "Weather Prediction by Numerical Process." Since then, CFD has been expanded by mathematicians and engineers for the application to flow problems in the area of industrial engineering. As the numerical techniques became more efficient and the computational power (digital computers) increasingly more affordable, CFD has expanded to many other areas of fluid dynamics. Of interest here is the realm of environmental fluid mechanics, which constitutes the study of naturally occurring fluid flows on the planet, such as air, water, and debris flows. In environmental hydraulics, we are mainly concerned with the flow of surface water within a very narrow range of temperatures and pressures. The fundamental laws and equations of fluid motion under these circumstances have long been known—conservation of mass and of momentum—and will be used here as the basis for a real-world model for computing the flow in natural bodies of water.

3.1 The Mathematical Model

Flow phenomena in natural rivers are 3D, especially those at or near meander bends, local expansions and contractions, or at hydraulic structures. Turbulence is an essentially 3D phenomenon, and 3D models are particularly useful for the

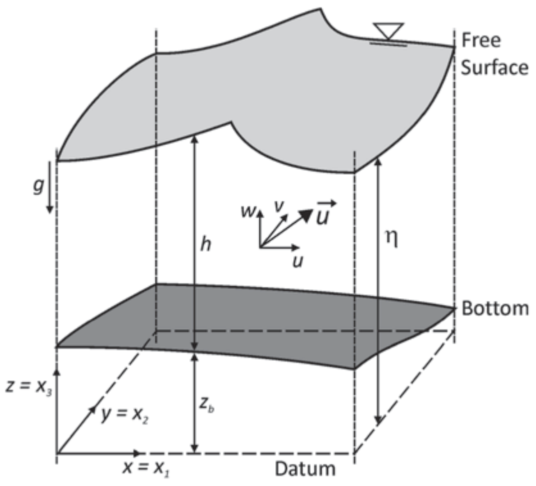

Fig. 6.2 Coordinate system used and the definition of some variables. Note that $u = u_1$, $v = u_2$, and $w = u_3$

simulation of turbulent heat and mass transport. These models are usually based on the Reynolds-averaged form of the Navier–Stokes equations, known since the nineteenth century, using additional equations of varied degree of complexity for the turbulence closure. Their derivation can be found in many basic textbooks on fluid dynamics, therefore, they will only be presented here without further consideration. The Navier–Stokes equations represent the statement of Newton's second law for fluids, i.e., the conservation of momentum, and in the Cartesian coordinate system and for incompressible fluids they can be written as

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = \frac{F_i}{\rho} - \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j} - \overline{u_i' u_j'} \right) \tag{6.1}$$

where i is the Cartesian directions ($i = 1$ for x, $i = 2$ for y, and $i = 3$ for z) as in Fig. 6.2, u_i is the Cartesian component of the velocity along the x_i direction ($i = 1, 2, 3$), ρ is the fluid density, p is the pressure, F_i is the component of the body forces per unit volume in the ith direction, ν is the kinematic molecular viscosity, $-\rho\overline{u_i' u_j'}$ are the turbulent stresses; and the indexed summation convention is used. Equation (6.1) constitutes a system of equations, one for each coordinate direction, i.e., for $i = 1, 2$, and 3. The body forces include gravitational, buoyancy, and Coriolis forces, or any other body forces that may be present (such as magnetic forces in magnetohydrodynamic fluids).

Conservation of mass is expressed by the continuity equation for incompressible fluids:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{6.2}$$

Table 6.2 Typical scales for the application of surface hydraulics models. The number of nodes provides a measure of the numerical burden associated with discretizing the governing equations

Dimension	Number of nodes	Spatial scale (m)	Time scale	Number of equations
1D	$\sim 10^2$	10^3–10^5	Up to decades	2
2D	10^3–10^5	10^3	Days	3 (+2 for turbulence)
3D	10^4–10^6	10^1–10^3	Minutes to hours	4 (+8 for turbulence)

The turbulence terms ($-\rho\overline{u_i'u_j'}$) result from averaging the original Navier–Stokes equations using the Reynolds decomposition (see [9] for a more detailed explanation of the technique) and require additional closure equations. One of the commonly used closure techniques is given by the k–ε model, but there are many other alternative choices. The reader is directed to the turbulence modeling monograph [10] for further details about this subject.

When shallow-water flows are nearly horizontal, the 3D effects are not essential, and in the long-wave approximation (i.e., waves with long characteristic lengths compared to the water depth, such as tidal waves in seas and flood waves in rivers), considerable simplifications can be made to the governing equations and efficient mathematical models can be built, where depth-averaged quantities replace their fully 3D counterparts. This process is called order reduction and results in models that are useful in many practical applications, where 3D detail is not needed or when the extent of the problem is too large for the available computing power. On the other hand, 2D modeling constitutes a step-up from the much simpler 1D approach, albeit at a cost of significant additional data requirements, but the recent advancements in low-cost data acquisition techniques with high resolution and accuracy, such as light detection and ranging (LIDAR) and multi-beam echo sounding, has facilitated the fulfillment of those requirements. Table 6.2 shows the scales of applicability of the different types of modeling in view of the commonly used current-day computing facilities. In Table 6.2, the number of partial differential equations of the mathematical model is given, with additional differential equations for turbulence modeling in parenthesis.

As the moniker implies, 2D depth-averaged modeling results from averaging the 3D flow Eq. (6.1) and (6.2) along the vertical direction, and replacing the point velocities and other dependent variables with their depth-averaged counterparts. For example, for the velocity components we have:

$$U = \frac{1}{h}\int_{z_b}^{\eta} u dz \quad \text{and} \quad V = \frac{1}{h}\int_{z_b}^{\eta} v dz \tag{6.3}$$

where U and V are the depth-averaged components of the velocity in the x and y directions, respectively (see Fig. 6.2 for definition of the remaining variables). The continuity equation is the most straightforward to integrate and results in

$$\frac{\partial \eta}{\partial t} + \frac{\partial hU}{\partial x} + \frac{\partial hV}{\partial y} = 0 \tag{6.4}$$

The momentum equations can be averaged in the same way, but this time nonlinear terms appear—the full mathematical derivation of the equations is outside of the scope of this text, but the interested reader is directed to [11] for the complete details. Including the Coriolis and pressure terms, whose integration is trivial, the depth-averaged momentum equations become

$$\frac{\partial(hU)}{\partial t}+\frac{\partial(hU^2)}{\partial x}+\frac{\partial(hUV)}{\partial y} = F_x + fhV - gh\frac{\partial\eta}{\partial x}-\frac{gh^2}{2\rho_0}\frac{\partial\rho}{\partial x}-\frac{\tau_{bx}}{\rho_0}+\frac{\partial(h\tau_{xx})}{\partial x}+\frac{\partial(h\tau_{xy})}{\partial y} \tag{6.5}$$

$$\frac{\partial(hV)}{\partial t}+\frac{\partial(hUV)}{\partial x}+\frac{\partial(hV^2)}{\partial y} = F_y - fhU - gh\frac{\partial\eta}{\partial y}-\frac{gh^2}{2\rho_0}\frac{\partial\rho}{\partial y}-\frac{\tau_{by}}{\rho_0}+\frac{\partial(h\tau_{xy})}{\partial x}+\frac{\partial(h\tau_{yy})}{\partial y} \tag{6.6}$$

where f is the Coriolis parameter ($=2\Omega\sin\Phi$, with Ω = angular rate of earth's revolution), Φ is the geographic latitude, F_i is the driving forces ($i=x, y$), ρ_0 is the density of a reference state, and τ_{bi} is the bottom stresses ($i=x, y$).

The above equations are sometimes called the shallow-water equations or the depth-averaged Navier–Stokes equations. The cross-stresses τ_{ij} include viscous friction, turbulent friction, and the nonlinear terms resulting from the vertical averaging process, which are usually called the radiation stresses:

$$\tau_{ij}=\frac{1}{h}\int_{z_b}^{\eta}\left[\nu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\overline{u_i'u_j'}+(u_i-U_i)(u_j-U_j)\right]dz \tag{6.7}$$

In most natural bodies of water, the molecular viscosity terms can be safely neglected in comparison with the turbulence terms. The radiation stresses are often neglected, but they represent important physical phenomena. For example, in bends they are at least partly responsible for shifting the high velocity part of the flow profile from the inner bank at the upstream region to the outer bank at the downstream region of the bend—see [12]. In general, however, the terms of Eq. (6.7) are collapsed in the form of diffusion coefficients and written as

$$\tau_{ij}=D_i\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}-\frac{1}{2}\delta_{ij}\frac{\partial U_k}{\partial x_k}\right) \tag{6.8}$$

where δ_{ij} is the Kronecker delta (=1 if $i=k$, 0 otherwise) and D_i is the diffusion coefficient in the ith direction (in general, $D_1=D_2=D_H$). In turbulent flow, the diffusion coefficients can be prescribed or computed from any of the many existing turbulence models (see [10] for more details), and the bottom shear stresses are assumed to have the same direction of the depth-mean velocity and proportional to the square of its magnitude:

$$\frac{\tau_{bx}}{\rho_0} = C_f u\sqrt{u^2+v^2} \quad \text{and} \quad \frac{\tau_{by}}{\rho_0} = C_f v\sqrt{u^2+v^2} \tag{6.9}$$

where C_f is the standard skin friction coefficient ($C_f \approx 0.003$). Note that Eq. (6.9) can also be written in terms of the Manning's roughness coefficient, *n,* or in terms of Chézy's roughness coefficient, *C:*

$$\frac{\tau_{bx}}{\rho_0} = \frac{gn^2}{h^{1/3}}u\sqrt{u^2+v^2} = \frac{g}{C^2}u\sqrt{u^2+v^2} \quad \text{and} \quad \frac{\tau_{by}}{\rho_0} = \frac{gn^2}{h^{1/3}}v\sqrt{u^2+v^2} = \frac{g}{C^2}v\sqrt{u^2+v^2} \tag{6.10}$$

The driving forces remaining in Eq. (6.5) and (6.6) include such effects as atmospheric pressure gradients, wind stresses, density gradients, and tidal stresses. The shallow-water equations can be written in many possible forms. Those forms may include different terms than the ones considered above (corresponding to other physical effects), or may be written in terms of curvilinear coordinates, for example. Many other aspects that are of interest, but that are outside the scope of this chapter, are described with much greater detail in Ref. [13].

3.2 The Numerical Model

Analytical solutions for the shallow-water equations, Eq. (6.4, 6.5, and 6.6), are not known except for a very few simplified cases, therefore, application to the complex topology usually encountered in natural topography requires the application of numerical mathematics solution methods. When used in CFD, these methods transform the problem of finding the solution of a system of continuous partial differential equations into that of finding the solution of a much larger set of algebraic equations that can be solved in a computer. The algebraic equations are only approximations of the original differential equations that result from discretizing the continuous space into a finite set of points using approximation theory. There are many different methods to accomplish the discretization, and different choices result in distinct sets of algebraic equations.

The choice of a discretization technique is determined by many different factors, some of which may be subjective. For example, a mathematician may favor finite elements using weighted residual methods, while a physicist may prefer to use control volumes. Residual methods are based on minimizing some sort of error, or residual, of the governing equations and are mathematically very solid and well established, but their derivation is more laborious and loses physical meaning very early in the discretization stage, making them more difficult to implement and to debug the code. The control volume formulation, on the other hand, employs the same thought process generally used to derive the governing equations (i.e., vanishing control volumes) and the resulting algebraic equations maintain terms with physical significance all the way to the programming level, making the entire process easier to grasp and to program correctly without code bugs.

In surface hydraulics, there are other considerations to take into account. For example, in lakes and many rivers, the flow field is generally smooth and without abrupt changes; in fast rivers, however, the flow field changes rapidly and may have discontinuities, such as in areas with hydraulic jumps and regime change. Particularly in catastrophic flooding, such as that caused by dam- and dike-break events, abrupt- and fast-moving wave fronts are produced over dry and irregular beds in the downstream plains and valleys. The numerical techniques needed to compute accurately this latter type of flows are usually more complex than those needed for slow flows because they must yield low numerical dissipation, produce wiggle-free solutions at the discontinuity areas, and be robust enough to survive the regions of vanishing water depth. The numerical model presented in this section was designed having these features as the primary concern.

The control volume formulation is used here to discretize the governing equations. To simplify the mathematics, Eqs. (6.4, 6.5, and 6.6) are first simplified (uniform and constant density, no Coriolis terms) and recast in the conservative form:

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}(\mathbf{U})}{\partial x}+\frac{\partial \mathbf{G}(\mathbf{U})}{\partial y}-\left(\frac{\partial \mathbf{P}(\mathbf{U})}{\partial x}+\frac{\partial \mathbf{Q}(\mathbf{U})}{\partial y}\right)=\mathbf{S}(\mathbf{U}) \tag{6.11}$$

where **U** is the vector containing the conservative variables, **F** and **G** are the inviscid fluxes, **P** and **Q** are the viscous fluxes, and **S** is the vector containing the forcing (source) terms which are defined as

$$\mathbf{U}=\begin{bmatrix} h \\ hu \\ hv \end{bmatrix},\quad \mathbf{F}=\begin{bmatrix} hu \\ hu^2+gh^2/2 \\ huv \end{bmatrix},\quad \mathbf{G}=\begin{bmatrix} hu \\ huv \\ hv^2+gh^2/2 \end{bmatrix},$$

$$\mathbf{P}=\begin{bmatrix} 0 \\ \nu_t h\dfrac{\partial u}{\partial x} \\ \nu_t h\dfrac{\partial v}{\partial x} \end{bmatrix},\quad \mathbf{Q}=\begin{bmatrix} 0 \\ \nu_t h\dfrac{\partial u}{\partial y} \\ \nu_t h\dfrac{\partial v}{\partial y} \end{bmatrix},\quad \mathbf{S}=\begin{bmatrix} 0 \\ gh\left(S_{0x}-S_{fx}\right)+\dfrac{\tau_{wx}}{\rho_0} \\ gh\left(S_{0y}-S_{fy}\right)+\dfrac{\tau_{wy}}{\rho_0} \end{bmatrix}$$

Here, lowercase letters are now used to define the components of the depth-averaged velocity, v_t is the eddy viscosity coefficient (D_i in Eq. (6.8)), S_0 ($S_{0i} = -\partial z_b / \partial x_i$) is the bed slope, and S_f $\left(ghS_{fi} = c_d u_i \sqrt{u_1^2 + u_2^2}\right)$ is the bottom friction. τ_{wx} and τ_{wy} are the wind stresses at the free surface in the x- and y-directions and are estimated from

$$\tau_{wx} = \rho_a C_D u_w \sqrt{u_w^2 + v_w^2} \quad \text{and} \quad \tau_{wy} = \rho_a C_D v_w \sqrt{u_w^2 + v_w^2}$$

where ρ_a is the density of air, u_w and v_w are the components of the wind speed at a certain prescribed height above the free surface (usually 10 m), and C_D is a dimensionless drag coefficient. The wind forcing terms may be neglected for the cases where the effects of bottom slope and friction are dominant, such as in rivers and laboratory channels, but are usually kept for lakes and reservoirs where wind-driven circulation is important.

For shallow flows, the effective eddy viscosity can be estimated well by the turbulent boundary layer theory, yielding a depth-mean value of $v_t = \varepsilon_t h u_*$, where u_* is the shear velocity $(= \sqrt{\tau_b / \rho})$ and ε_t is a parameter with a theoretical value of 0.068. Usually, $\varepsilon_t \in 0.1, 10$, but in practice, it can vary by several orders of magnitude outside this range—see [14]—and must be calibrated for each situation. Alternatively, a constant value of the turbulent eddy viscosity ε_t can also be prescribed.

The system's conservative formulation as written above, which remains valid across discontinuities in the flow variables such as hydraulic jumps, provides an ideal basis for the integral formulation over a control volume Ω used by traditional control volume methods and is adopted here. Defining $\mathbf{E} = (\mathbf{F}, \mathbf{G})^T$ and $\mathbf{R} = (\mathbf{P}, \mathbf{Q})^T$, Eq. (6.11) is now written in integral form as

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{U} \mathrm{d}\Omega + \oint_{\partial\Omega} (\mathbf{E} \cdot \mathbf{n}) \mathrm{d}s - v_t \oint_{\partial\Omega} (\mathbf{R} \cdot \mathbf{n}) \mathrm{d}s = \int_\Omega \mathbf{S} \mathrm{d}\Omega \tag{6.12}$$

where $\mathbf{n}$ is the outward-pointing unit vector normal to the control volume boundary $\partial\Omega$, and after applying Gauss' theorem to the flux integral. Equation (6.12) is the form used to discretize the shallow-water equations in the following manner:

$$\frac{\partial \Omega_i \mathbf{U}_i}{\partial t} = \sum_{k=1}^{n_C} \mathbf{E}_{ik} \Delta l_{ik} - v_t \sum_{k=1}^{n_C} \mathbf{R}_{ik} \Delta l_{ik} + \mathbf{S}_i \Omega_i \tag{6.13}$$

where $\mathbf{U}_i$ are the average values of the conserved variables over cell i, $\mathbf{E}_{ik}$ and $\mathbf{R}_{ik}$ are the inviscid and the viscous fluxes through edge k, $\mathbf{S}_i$ contains the source terms, Ω_i is the cell's area, and n_C is the number of sides of the polygonal control volume. The scheme adopted here discretizes the domain in nonoverlapping triangles ($n_C = 3$) and the values of the conserved variables are located at the geometric center of the control volumes. The linear reconstruction operators used, which must satisfy the cell averaging requirements that guarantee cell-wise discrete conservation properties, exploit the fact that the cell average is also a pointwise value of any

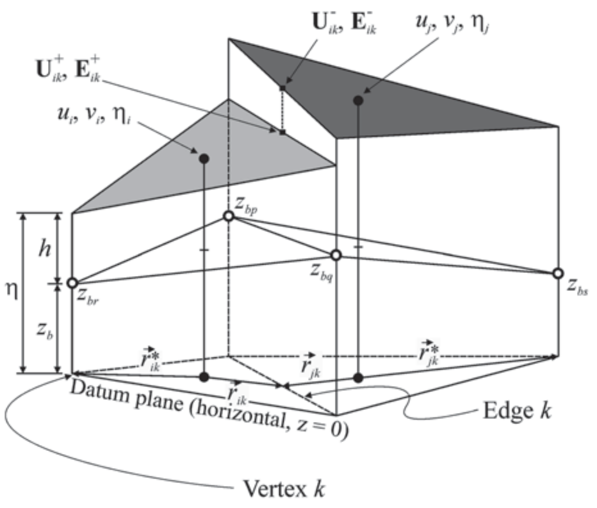

Fig. 6.3 General definition of the control volume geometry and location of the conserved variables. The dependent variables are defined at each triangle's centroid. $\vec{r}_{ik}$ is a vector that points from the centroid of triangle *i* to the midpoint of edge *k*, and $\vec{r}^*_{ik}$ is a similar vector that points to vertex *k*

permitted linear reconstruction evaluated at the centroid of the triangle (this is only true for fully wet triangles). The bed elevation above datum, however, is defined at the triangle vertices. Each triangle defines a control volume, which is used to solve Eq. (6.12). The general configuration of the discretization apparatus is shown in Fig. 6.3, where η is the free surface elevation ($=h+z_b$).

Following the principles of Godunov-type methods, the inviscid fluxes $\mathbf{E}_{ik}$ are numerical fluxes arising from a local Riemann problem at each cell edge. There are several approximate Riemann solvers that were specifically proposed for the shallow-water equations. In this work, $\mathbf{E}_{ik}$ are computed using Roe's [15] flux function at the cell edges:

$$\mathbf{E}_{ik} = \frac{1}{2}\left[(\mathbf{E}^+_{ik} + \mathbf{E}^-_{ik}) - \Gamma(\mathbf{U}^+_{ik} - \mathbf{U}^-_{ik})\right] \tag{6.14}$$

where the "+" quantities are reconstructed at the midpoint of the edge *k* using data from control volume *i* and the "–" quantities from control volume *j* (see Fig. 6.3). There are many forms of the upwinding factor $\mathbf{\Gamma}$, and an exhaustive exposition can be found in [16]. The approach of [17] is used here, which is based on the 1D Riemann problem in the direction normal to the cell edge, and which is included below for completeness. In this approach, $\mathbf{\Gamma}=|\mathbf{A}|=\mathbf{R}|\mathbf{\Lambda}|\mathbf{L}$, where $\mathbf{A}$ is the Jacobian of the flux evaluated at Roe's average state:

$$\mathbf{A} = \frac{\partial(\mathbf{E}\cdot\mathbf{n})}{\partial\mathbf{U}} = \begin{bmatrix} 0 & n_x & n_y \\ \left(c^2-u^2\right)n_x - uvn_y & 2un_x + vn_y & un_y \\ \left(c^2-v^2\right)n_y - uvn_x & vn_x & un_x + 2vn_y \end{bmatrix} \tag{6.15}$$

The quantities **R**, **L**, and **Λ** are the right and left eigenvectors and eigenvalues of **A**, respectively. The eigenvalues are given by

$$\lambda_1 = un_x + vn_y, \quad \lambda_2 = \lambda_1 - cn, \quad \lambda_3 = \lambda_1 + cn \tag{6.16}$$

where n_x and n_y are the components of the edge's normal vector **n**, n is its magnitude (=1 for a unit normal), and $c=(gh)^{1/2}$ is the wave celerity. The right and left eigenvector matrices are given by

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 1 \\ n_y & n_y - ucn_x/n & u + cn_x/n \\ -n_x & v - cn_y/n & v + cn_y/n \end{bmatrix},$$
$$\mathbf{L} = \begin{bmatrix} -\left(un_y - vn_x\right)/n & ny/n^2 & -n_x/n^2 \\ \left(un_x + vn_y\right)/2cn + 1/2 & -n_x/2cn & -n_y/2cn \\ -\left(un_x + vn_y\right)/2cn + 1/2 & n_x/2cn & n_y/2cn \end{bmatrix} \tag{6.17}$$

and Roe's average state is defined as

$$u = \frac{u^+\sqrt{h^+} + u^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}}, \quad v = \frac{v^+\sqrt{h^+} + v^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}}, \quad c = \sqrt{\frac{1}{2}g(h^+ + h^-)} \tag{6.18}$$

where all the quantities are evaluated at the edge midpoint.

Following [18], matrix **Λ** is evaluated as

$$|\Lambda| = \begin{vmatrix} f(\lambda_1) & & \\ & f(\lambda_2) & \\ & & f(\lambda_3) \end{vmatrix} \tag{6.19}$$

where $f(\lambda)$ is a function of the eigenvalues that incorporates the entropy fix, and is defined as

$$f(\lambda) = \begin{cases} |\lambda|, & |\lambda| \geq \delta_e \\ \dfrac{\lambda^2 + \delta_e^2}{2\delta_e}, & |\lambda| < \delta_e \end{cases} \tag{6.20}$$

where δ_e is a very small number. This prevents any of the eigenvalues to vanish, which causes the dissipation for that component to vanish also at that location and may lead to numerical instability. It also eliminates expansion shocks and makes Roe's flux function differentiable. The above method is particularly appropriate for discontinuous flows, where sharp gradients are important and must be accurately calculated, such as across hydraulic jumps, wet–dry fronts, and in dam-break flows.

Second-order accuracy is achieved using a piecewise linear model for the cell variables with the usual modified upwind scheme for conservation laws (MUSCL) reconstruction of [19], with limiting to enforce monotonicity near sharp gradients and discontinuities of the dependent variables:

$$V_{ik} = V_i + \Phi_i \nabla V_i \cdot \vec{r}_{ik} \tag{6.21}$$

where V_i is the variable (u, v, h) defined at the center of the control volume *i*, V_{ik} is the same quantity at the midpoint of the edge *k* (computed from the side of element *i*), ∇V_i is the gradient of *V* (piecewise linear) over the control volume, and $\vec{r}_{ik}$ is a position vector located at the centroid of the control volume and pointing to the midpoint of edge *k*. The flux limiter Φ_i has the objective of preventing the formation of local extrema at the flux integration points. Barth and Jespersen [20] were the first to propose a truly multidimensional limiter for unstructured grids. This limiter has been used by many, but it introduces nondifferentiability to the computation of the reconstructed function and, consequently, to the fluxes, impacting adversely the convergence properties of the solver. Venkatakrishnan [21] resolved this issue by introducing a modification to Barth and Jaspersen's limiter that makes it continuously differentiable. This limiter is applied to triangular cell-centered grids in the following manner:

1. Find the largest negative $(\delta V_{min} = \min(V_i - V_j))$ and positive $(\delta V_{max} = \max(V_i - V_j))$ difference between the solution at the centroid of the triangle, V_i, and that of all its neighbors that share an edge with it, V_j (j=1, 2, 3).
2. Compute the unconstrained reconstruction value at the midpoint of each edge, $V_{ik}^* = V_i + \nabla V_i \cdot \vec{r}_{ik}$.
3. For each edge, compute the maximum allowed value of Φ_{ik}:

$$\Phi_{ik} = \begin{cases} \dfrac{1}{\delta'}\left[\dfrac{\left(\Delta V_{max}^2 + \varepsilon^2\right)\delta' + 2\delta'^2 \Delta V_{max}}{\Delta V_{max}^2 + 2\delta'^2 + \delta' \Delta V_{max} + \varepsilon^2}\right] & \text{if}\{\delta > 0 \\ \dfrac{1}{\delta'}\left[\dfrac{\left(\Delta V_{min}^2 + \varepsilon^2\right)\delta' + 2\delta'^2 \Delta V_{min}}{\Delta V_{min}^2 + 2\delta'^2 + \delta' \Delta V_{min} + \varepsilon^2}\right] & \text{if } \delta' < 0 \end{cases}$$

4. Choose $\Phi_i = \min(\Phi_{ik})$.

In step 3, $\Delta V_{max} = \delta V_{max} - V_i$, $\Delta V_{min} = \delta V_{min} - V_i$, δ' is given by $\delta' = sign(V_{ik}^* - V_i)(\left|V_{ik}^* - V_i\right| + \delta)$, and δ is a very small number to avoid division by zero. ε is a parameter that controls over- and undershoots and depends on the estimate of a length scale Δ*x*, $\varepsilon = (K\Delta x)^3$. In the present work, *K* is set to 0.075 (found by numerical experimentation) and Δ*x* is a local mesh length scale set to the average length of the edges of the element.

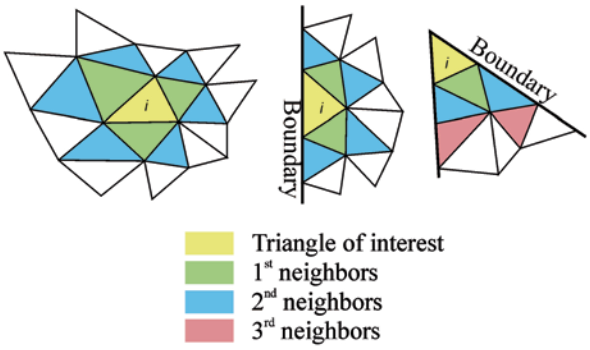

Fig. 6.4 Depiction of first, second, and third neighbors to a computational cell (cell *i,* in *yellow*) for different geometries. The *colored* area shows the stencil used in each case and the empty cells, which are third neighbors or higher, do not contribute to the computational cell

Computation of the gradients ∇V_i is done using a least squares technique which requires the solution of a two-by-two system for each cell *i* of the computational domain:

$$\begin{cases} aV_x + bV_y = d \\ bV_x + cV_y = e \end{cases} \tag{6.22}$$

where V_x and V_y are the Cartesian components of ∇V_i and

$$a = \sum_{j=1}^{n_T} w_j^2 (x_j - x_i)^2, \quad b = \sum_{j=1}^{n_T} w_j^2 (x_j - x_i)(y_j - y_i)$$

$$c = \sum_{j=1}^{n_T} w_j^2 (y_j - y_i)^2, \quad d = \sum_{j=1}^{n_T} w_j^2 (V_j - V_i)(x_j - x_i)$$

$$e = \sum_{j=1}^{n_T} w_j^2 (V_j - V_i)(y_j - y_i)$$

where n_T is the number of triangles belonging to the computational molecule. All of the above terms depend solely on the grid geometry; therefore, they can be precomputed and stored for later use by the iterative solution cycle. The system is solved directly using Cramer's rule and the inverse distance weighting is used $(w_j = ((x_j - x_i)^2 + (y_j - y_i)^2)^{-1/2})$.

The least-squares procedure thus defined requires a minimum of three control volumes, including *i,* to form a determined system. It is desirable to exceed that number by building stencils by successively adding first, second, and third neighbors until a desired size is reached. In particular, the stencils near boundaries may require farther neighbors. The additional data contained in the expanded stencil allows filtering out noise at the expense of little additional computational cost. Some of the typical stencils used are shown in Fig. 6.4.

The viscous fluxes $\mathbf{R}_{ik}$ are discretized using a central differencing approach. The viscous flux terms are computed at the edge midpoints, where *u, v,* and their gradients must be calculated. The velocity components at the edge midpoints are simply $V_k = (V_{k1} + V_{k2})/2$, where V_k is the velocity of interest (*u* or *v*) at the midpoint of edge *k,* and V_{k1} and V_{k2} are its values at the edge's extremities (the triangle vertices). The values of the dependent variables at the triangle vertices are obtained using least squares second-order interpolation procedure [22] that uses the information of all the cells sharing the vertex:

$$V_l^V = \sum_{Neighb} c_i V_i \tag{6.23}$$

where V_l^V is the conserved variable at vertex *l,* V_i denotes the conserved variable at the centroid of cell *i* that shares vertex *l,* and the sum is carried over all the cells sharing vertex $\left(x_l^V, y_l^V\right)$. c_i is the dimensionless coefficient that is defined by

$$c_i = \frac{w_i}{\sum\limits_{Neighb} w_i} \tag{6.24}$$

with

$$w_i = 1 + \lambda_x\left(x_i - x_l^V\right) + \lambda_y\left(y_i - y_l^V\right)$$
$$\lambda_x = \left(I_{xy}R_y - I_{yy}R_x\right)/\left(I_{xx}I_{yy} - I_{xy}^2\right), \quad \lambda_y = \left(I_{xy}R_x - I_{xx}R_y\right)/\left(I_{xx}I_{yy} - I_{xy}^2\right)$$
$$R_x = \sum_{Neighb}\left(x_i - x_l^V\right), \quad R_y = \sum_{Neighb}\left(y_i - y_l^V\right)$$
$$I_{xx} = \sum_{Neighb}\left(x_i - x_l^V\right)^2, \; I_{yy} = \sum_{Neighb}\left(y_i - y_l^V\right)^2, \quad I_{xy} = \sum_{Neighb}\left(x_i - x_l^V\right)\left(y_i - y_l^V\right)$$

All the quantities used in the interpolation can be computed in a preprocessing stage and stored for later use, making the method extremely fast and efficient. Because all the limiting is applied to the edge midpoints via MUSCL reconstruction (as discussed in previous paragraphs), it may appear that extrema could occur at the vertices and that some sort of limiting constraint would be required when applying the above interpolation technique. In [23], it is shown that this is not so and that, in fact, edge midpoint limiting is sufficient to ensure positivity of the cell averaged data without imposing special requirements to the quality of the computational grid.

The computation of the flow gradients at the edge midpoints follows the diamond method of Coirier [24] adapted for triangular unstructured grids. In this approach, the computational molecule associated with each edge is formed by the edge's end points and the centers of the control volumes that share that edge, as shown in Fig. 6.5. The method uses linear interpolation to determine the value of the gradient of the dependent variables over each of the shaded triangles, in a manner that is identical to the technique used in finite element interpolation (i.e., shape functions). Then, the gradient at the edge midpoint is given by the area-averaged value of the same gradients over each of the triangles (shaded areas in Fig. 6.5).

Using the notation of Fig. 6.5, the gradient of the dependent variables at the midpoint of edge *k* is given for interior edges as

$$\begin{aligned}\frac{\partial V_k}{\partial x} &= \frac{1}{2(A^+ + A^-)}\left[(y_4 - y_3)(V_1 - V_2) + (y_1 - y_2)(V_3 - V_4)\right] \\ \frac{\partial V_k}{\partial y} &= \frac{1}{2(A^+ + A^-)}\left[(x_3 - x_4)(V_1 - V_2) + (x_2 - x_1)(V_3 - V_4)\right]\end{aligned} \tag{6.25}$$

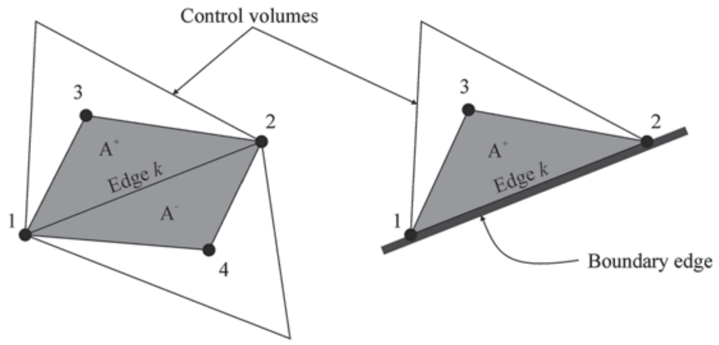

Fig. 6.5 Computational molecule used in the calculation of the viscous fluxes

and for boundary edges as

$$\begin{aligned}\frac{\partial V_k}{\partial x} &= \frac{1}{2A^+}\left[(y_2 - y_3)V_1 + (y_3 - y_1)V_2 + (y_1 - y_2)V_3\right] \\ \frac{\partial V_k}{\partial y} &= \frac{1}{2A^+}\left[(x_3 - x_2)V_1 + (x_1 - x_3)V_2 + (x_2 - x_1)V_3\right]\end{aligned} \tag{6.26}$$

where A^+ and A^- are the areas of the shaded triangles (which are equal to one third of the area of the respective control volumes). Note that points 3 and 4 are the control volume centroids, therefore the dependent variables there are a direct result of the solution of the governing equations. However, the vertex values of the same variables (points 1 and 2) are a result of the least squares second-order interpolation described in the preceding paragraphs. Consequently, the discretization of the viscous terms is also second-order accurate.

Typical environmental flows are subject to multiple forcing factors, such as bottom friction, bed slope, wind forcing, Coriolis forces, and tidal forces. It is, therefore, important to include accurate representations of these forcing terms—the source terms of Eq. (6.12)—in the conceptual and numerical model. In this section, only bed slope and bottom friction are considered, but the other terms are discretized using similar, straightforward, centered difference schemes.

Bed friction is represented by Eq. (6.9). For most circumstances, a simple treatment of these terms is sufficient. Using a standard cell-centered approach that uses only information at the cell centroids results in

$$\int_\Omega ghS_{fx}d\Omega = C_f u_i\sqrt{u_i^2 + v_i^2}\,\Omega_i \quad \text{and} \quad \int_\Omega ghS_{fy}d\Omega = C_f v_i\sqrt{u_i^2 + v_i^2}\,\Omega_i \tag{6.27}$$

where the subscript i indicates quantities evaluated at the centroid of control volume i. The main numerical problem associated with this type of formulation arises for very shallow depths, such as near advancing and receding wet–dry fronts. In very shallow water, a dimensional analysis shows that the relative importance of the convective terms in Eq. (6.11) becomes small and the system is no longer convection dominated, resulting in a stiff system that becomes severely restricted by the very small time step required to maintain the stability of the numerical solution. In this circumstance, explicit discretization of the friction terms can also produce numerical oscillations and even localized velocity inversion, especially when the roughness is high.

To circumvent these problems, a pointwise implicit discretization of the friction terms is used. In this approach, the momentum equations are advanced explicitly in time (from time step n to time step $n+1$) without the friction terms, producing the intermediate unit discharges q'_{xi} and q'_{yi} in the x- and y-directions, respectively (the unit discharges are defined as $q_{xi} = h_i u_i$ and $q_{yi} = h_i v_i$, and the subscript i denotes the control volume number). Then, the final unit discharges are updated using

$$
\begin{aligned}
q_{xi}^{n+1} &= \frac{q'_{xi}}{1 + g\Delta t (S'_{fxi} / u'_i)} \\
q_{yi}^{n+1} &= \frac{q'_{yi}}{1 + g\Delta t (S'_{fxi} / v'_i)}
\end{aligned}
\tag{6.28}
$$

where the superscript $n+1$ denotes the time step. All the primed variables refer to quantities computed from the frictionless updated values of u and v in the time step.

Discretization of the bed slope term within the context of Godunov-type solvers has been a topic of much research in the past decade. This stems from the fact that, when using cell-centered control volumes with a MUSCL reconstruction technique, it is necessary to properly discretize the source term in order to satisfy the C-property, i.e., to ensure the balance between the flux gradient and the slope term in order to guarantee hydrostatic balance in still water conditions—e.g., see [25]. To circumvent this issue, the bed elevation, z_b, is defined at the control volume vertices instead of at its centers, as shown in Fig. 6.3. Using this approach, there is no discontinuity in the bed elevation across control volumes and the Riemann problem retains its self-similarity solution, therefore a simple centered differencing approach can be used, as indicated in [16]. Bed elevation is piecewise linear within the control volumes and its slope can be calculated using the standard interpolation techniques used in finite element methods.

Finally, the system of governing equations must be integrated in time. The choice here is to advance the solution explicitly in time using nonlinear strong stability preserving Runge–Kutta (SSPRK) schemes, also known as total variation diminishing (TVD) Runge–Kutta schemes [26]. This is done by first rewriting the governing equations as a coupled system of ordinary differential equations:

$$
\Omega_i \frac{\partial q_i}{\partial t} = R_i(u, v; t), \quad i = 1, 2, 3 \tag{6.29}
$$

Table 6.3 Values of the SSPRK coefficients for the schemes used. For $m=1$, the method reduces to the traditional forward Euler method

Order (m)	i ($i=1,\dots,m$)	α_{ij} ($j=0,\dots,i-1$)	β_j ($i=1,\dots,m$)
1	1	1	1
2	1	1	1
	2	$\frac{1}{2},\frac{1}{2}$	$\frac{1}{2}$
3	1	1	1
	2	$\frac{3}{4},\frac{1}{4}$	$\frac{1}{4}$
	3	$\frac{1}{3},0,\frac{2}{3}$	$\frac{2}{3}$

SSPRK Strong Stability Preserving Runge–Kutta

where R_i is called the residual. Here, a simplified form of the SSPRK schemes is used, in which an m-stage SSPRK method for Eq. (6.29) is written in the form

$$\begin{cases} u^{(0)} = u^n \\ u^{(i)} = \sum_{j=0}^{i-1} \alpha_{ij} u^{(j)} + \beta_i \dfrac{\Delta t}{\Omega_i} R(u^{(i-1)}), \quad \alpha_{ij} \geq 0, \quad i = 1,\dots,m \\ u^{n+1} = u^{(m)} \end{cases} \tag{6.30}$$

where Δt is the time step size, the superscript n denotes the time level, and the parenthetic superscripts denote the Runge–Kutta level. The coefficients α_{ij} and β_i are chosen to meet desired criteria. Second- ($m=2$) and third-order ($m=3$) optimal SSPRK methods, in the sense of the Courant–Friedrichs–Lewy (CFL) coefficient θ, are used—see [26]. The values of the coefficients are given in Table 6.3. For time-dependent cases, Δt is prescribed or it is computed from

$$\Delta t = \theta \mathrm{Min} \left\{ \frac{l_k}{\lambda_k^*}, \frac{l_k^2}{\nu_k} \right\}_{k=1,\dots,N_{we}} \tag{6.31}$$

where l_k is the length of edge k, λ_k^* is the highest eigenvalue at the edge's midpoint, ν_k is the effective eddy viscosity at the same location, and N_{we} is the number of wet edges over the entire computational domain. In the latter case, θ must be prescribed. For inviscid computations, the terms containing ν_k in Eq. (6.31) are dropped and only the terms containing λ_k^* are considered. Note that the presence of source terms places additional restrictions on the maximum admissible time step that preserves stability. These source terms are problem dependent; therefore, the CFL condition must be considered a general guideline and the maximum time step must be determined through numerical experimentation, by prescribing either θ or Δt.

Although for time-dependent calculations the solution must be advanced in time in a physically consistent and accurate manner, in steady-state calculations it does not, because only the converged solution needs to retain physical meaning and be consistent with Eq. (6.11). This observation can be used to devise techniques to speed convergence. For example, the Runge–Kutta scheme could use coefficients that emphasize convergence rather than accuracy. For example, two convergence acceleration techniques can be employed: local time stepping and implicit residual smoothing. Local time stepping advances the solution at each cell using a time step close to the stability limit for that cell:

$$\Delta t_i = \theta \mathrm{Min} \left\{ \frac{l_k}{\lambda_k^*}, \frac{l_k^2}{\nu_k} \right\}_{k=1,2,3} \tag{6.32}$$

where the subscript i denotes the cell index and the subscript k denotes the edges that belong to that cell. This technique has the objective of speeding up the transport of information within the computational domain and results in an increase in the convergence rate of explicit schemes by a factor close to two—see [27].

The implicit residual smoothing suggested in [28] is another technique used to increase the maximum allowed time step. This is accomplished by enlarging the support of the scheme by averaging the residuals R_i with their neighbors by means of a smoothing operator:

$$\overline{R}_i = R_i + \kappa \sum_j (\overline{R}_j - \overline{R}_i) \tag{6.33}$$

where κ is a constant and the summation index j spans over all the neighboring elements that share an edge with element i. Using a suitable choice for κ (1/2 in this work), the resulting system is strongly diagonally dominant and can be easily solved using Jacobi iteration:

$$\overline{R}_i^{(m)} = \left(R_i + \kappa \sum_j \overline{R}_j^{(m\text{-}1)} \right) \left(1 + \kappa \sum_j 1 \right)^{-1} \tag{6.34}$$

In practice, the system does not need to be solved exactly and two or three iterations are sufficient to produce an adequately accurate approximation of $\overline{R}_i$ at the cell centers. Residual averaging is performed at every step of the Runge–Kutta marching procedure.

3.3 Numerical Implementation Considerations

Wetting and drying occur not only during the propagation of floods but also at the edges of any body of water. Thus, the dry–wet front constitutes not only a propagation problem but also a static boundary condition problem, because it defines the shoreline. It is not easy to include these effects in a straightforward manner in a numerical code and most researchers resort to different degrees of approximation. The difficulties associated with this problem were presented with detail by Horritt in [29], which also included a review of some of the approaches used by different researchers. Another review of several methods can be found in [30]. Here, a compromise between accuracy and computational efficiency is sought. The overall treatment of wet–dry lines impacts directly the quality of the solution: Mass conservation, unphysical solution behavior, and numerical stability are the major aspects of concern.

Advancing (wetting) fronts, such as those encountered during catastrophic flood events, are treated directly by the Riemann solver in Eqs. (6.15, 6.16, 6.17, and 6.18) and the numerical scheme is robust enough to treat the sharp gradients in these areas without difficulty. However, the standard Riemann solution does not apply to the dry bed problem and can result in the wrong prediction of the front velocity. A fix presented in [16] consists in wetting the dry bed of the receiving cell by some very small value δ_w. For $\delta_w > 0$, the propagating speed is not the same as for the true dry bed situation, but in the limit $\delta_w \to 0$ the two speeds coincide. In practice, due to the robustness of the numerical methods used, a value close to the machine zero can effectively be used for δ_w (currently, 10^{-12} is used).

Drying fronts are treated similarly, but they pose the additional problem that, during a drying time step, negative water depths may be reached. Mass conservation requires that the time step should be restricted to the value that corresponds to the time that takes the cell to dry out, i.e., to reach $h_i = 0$. If a control volume reaches negative depth, additional mass is introduced in the system at the receiving control volume(s), and the excess volume is equal to the "negative" volume that corresponds to the negative depth. In practice, it is too complex and time consuming to keep track of the required time step size for each control volume, which also could result in significant computational costs if too many restrictions were to be imposed on the maximum time step allowed. It is desirable to run unsteady phenomena with the largest time step possible, therefore, a less restrictive approach was chosen. After each time step, all the control volumes with negative depths are treated in the following manner: The water depth is set to zero and the amount of fluid corresponding to the "negative" volume is removed from the downstream-most wet control volume adjacent to it. If that control volume does not have enough water, the remaining volume is removed from its own downstream-most adjacent control volume (the adjacent control volume that has the lowest centroid bed elevation), and so on until all excess mass is accounted for. The process continues for every control volume with negative depth until the total mass balance is restored. A list of downstream-most elements is constructed for each control volume as part of a preprocessing stage and its computational cost is negligible.

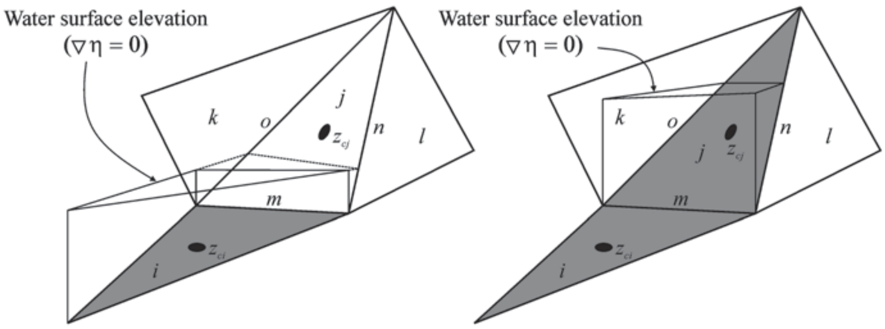

Fig. 6.6. Shoreline definition sketch. *Gray triangles* are wet control volumes. Control volumes are denoted by the letters *i, j, k,* and *l,* edges by *m, n,* and *o*. The *black dots* show the locations of the centroids of triangles *i* and *j*. On the *right*, the water-surface elevation in control volume *i* is not shown to improve clarity

The shoreline treatment is different from the two preceding cases. A shoreline is defined when all the surrounding dry triangles of a partially or fully wet control volume *i* have a mean bed elevation higher than the stage at the centroid of triangle *i*. Under this circumstance, the shoreline is defined at the control volume edges and it is treated identically to a solid wall (described later in this section): The flow velocity is set to zero and no mass can cross it. Additionally, the gradient of the water-surface elevation is set to zero in those control volumes.

Two typically occurring types of shoreline are shown in Fig. 6.6, where z_c denotes the bed elevation at the control volumes' centroid. At left, element *i* is fully wet, but the water-surface elevation is below z_{cj}, therefore, the shoreline is located at edge *m*. The gradient of the water-surface elevation in *i* is set to zero. Element *j* will become wet when the water-surface elevation in triangle *i* becomes higher than z_{cj}. On the other hand, at the right of Fig. 6.6, both elements *i* and *j* are wet. Assuming that control volumes *l* and *k* are dry, the shoreline is now located at edges *n* and *o*. The water surface is flat at control volume *j,* but not at *i* (assuming that all control volumes surrounding *i* are wet).

The decision whether a control volume is wet or dry is given by a threshold value of the water depth, h_{dry}: If $h_i > h_{dry}$ then control volume *i* is wet, otherwise it is dry. Hysteresis is introduced to prevent the triggering of instabilities, such as what might result from control volumes that become alternatively dry and wet over multiple successive time steps. This is done in the following manner: If a control volume *i* is wet, it becomes dry when h_i falls below h_{dry}; if *i* is dry, it becomes wet when h_i rises above h_{wet}. A hysteresis factor is used: $h_{wet} = \varphi h_{dry}$, where φ is set to a positive number greater than one. Note that the momentum equations are not solved for dry triangles, but the continuity equation is, which allows the continual computation of *h* even when $h < h_{dry}$, while simultaneously ensuring mass conservation. The threshold h_{dry} is a user set parameter, which is usually taken in the range 10^{-4}–10^{-8}.

Partially dry cells have further corrections to their area and water depth. When applying the time marching procedure to Eq. (6.13), it is assumed that $\partial\Omega_i / \partial t = 0$

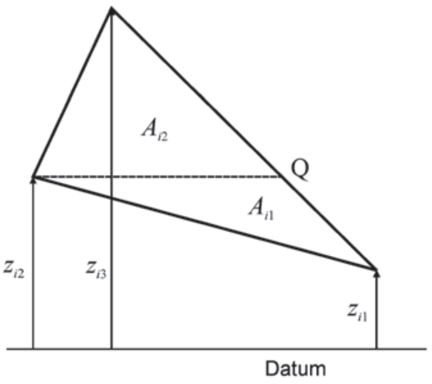

Fig. 6.7 Subdivision of a computational cell into two subtriangles for wetted area computations in partially dry cells. Note that $Q=(x_Q,y_Q,z_Q)=(x_Q,y_Q,z_{i2})$

in Eq. (6.29). In drying or wetting cells this is not true. In order to keep the computational complexity and cost low, instead of considering the time-dependent area term, the total area is simply replaced by the wetted area of the element. This area can be calculated very quickly and efficiently using the following procedure: Sort all the triangle vertices according to their vertical coordinate and renumber them appropriately, such that $z_3 \geq z_2 \geq z_1$ (see Fig. 6.7); divide the computational element in two triangles using a line connecting vertex z_2 with the opposing edge; the coordinates of point z_Q can be easily computed using the parametric equations of a line in three dimensions, namely

$$\begin{aligned} z_Q &= z_2 = z_1 + p(z_3 - z_1) \\ x_Q &= x_1 + p(x_3 - x_1) \\ y_Q &= y_1 + p(y_3 - y_1) \end{aligned}$$

where p is a parameter between 0 and 1, given by $p=(z_2-z_1)/(z_3-z_1)$. Now, compute the areas of the triangles A_{i1} and A_{i2}; the wetted area Ω_i^{w} can then be computed by linear interpolation:

$$\Omega_i^{\mathrm{w}} = \begin{cases} \dfrac{\eta_i - z_{i1}}{z_{i2} - z_{i1}} A_{i1} & \text{if} \quad \eta_i < z_2 \\ \dfrac{\eta_i - z_{i2}}{z_{i3} - z_{i2}} A_{i2} + A_{i1} & \text{if} \quad \eta_i \geq z_2 \end{cases}$$

where η_i is the water-surface elevation at the element. Similarly, in a partially dry cell, the water depth is not represented well by the water depth at its centroid. An effective water depth is used instead, which is computed in the following way:

$$h_i^{eff} = \begin{cases} \dfrac{1}{3} h_{i1} & \text{if}\, \eta_i < z_{i2} \\ \dfrac{1}{3}\left(h_{i1} + h_{i2}\right) & \text{if}\, \eta_i \geq z_{i2} \end{cases}, \tag{6.35}$$

Table 6.4 Inflow and outflow boundary conditions

	Inflow	Outflow
Subcritical flow	$u^- = u_B; \quad v^- = 0;$ $h^- = (\sqrt{h^+} - (u^- - u^+)/2\sqrt{g})^2$	$h^- = h_B; \quad v^- = v^+;$ $u^- = u^+ + 2\sqrt{g}(\sqrt{h^+} - \sqrt{h^-})$
Supercritical flow	$h^- = h_B; \quad u^- = u_B; \quad v^- = 0$	$h^- = h^+; \quad u^- = u^+; \quad v^- = v^+$

u^-, v^-, h^-, velocity and water depth outside the computational domain, where the boundary conditions apply. u is the component of the velocity normal to the edge and v is parallel to the edge
u^+, v^+, h^+, velocity and water depth inside the computational domain, obtained by solving the governing equations
u_B, h_B, prescribed boundary conditions for the (normal) velocity and the water depth

where h_{i1}, h_{i2}, and h_{i3} are the water depths at vertices z_{i1}, z_{i2}, and z_{i3}, respectively, computed by the center-to-vertex interpolation procedure of Eq. (6.23) and (6.24). All the relevant computational parameters (z_{i1}, z_{i2}, and z_{i3}, A_{i1}, and A_{i2}) can be sorted and/or computed in a preprocessing stage and stored for later use. Note also that, using this approach, h_i^{eff} will approach h_i (the water depth when all three vertices are wet) in a smooth and continuously differentiable manner, which is a desirable property to avoid numerical instability and speed convergence. The same can be said about Ω_i^w and Ω_i.

There are several limitations of the approach described above. For example, all the wet and dry computations are done at the cell edges, i.e., at the level of the cell flux computations, following an edge-based data structure described later in this section. This results in the approximation that a wet–dry boundary is constrained to cell edges and, therefore, it is dependent on the computational mesh. Balzano [30] has shown that this is a feature common to many methods and is remediated by using mesh refinement in those areas. Flooding of dry areas can never happen faster than one grid size per time step, which may pose restrictions to the time step size, due to the explicit nature of the time marching schemes used. Finally, on partially wet edges, the computation of mass and momentum fluxes is still only an approximation, albeit one that preserves mass and momentum. However, the approximation error is not severe because, in order to be considered wet, an edge must always be dry for less than half its length.

All boundary conditions are applied at the edges of the model grid, consistently with the edge-oriented data structure used (see the paragraphs below). They use Riemann invariants and are applied in a similar manner as used by [31], i.e., the boundary fluxes are also computed by solving a Riemann problem between the interior states and the "ghost" states outside the computational domain. These "ghost" states are introduced in order to compute the boundary fluxes in a similar and consistent way to the interior fluxes. The inflow and outflow boundary conditions are summarized in Table 6.4. Note that the value of the velocity boundary condition appearing in Table 6.4 is the normal inflow velocity u_B, but in practice the discharge is the desired quantity to enforce. This is accomplished by a fixed-point iteration that computes the values of h^- and u^- successively until the desired value of the discharge is achieved. Usually two or three iterations are sufficient to match the prescribed discharge with less than 1 % of error. Additionally, it was found that convergence to steady state was improved if the stage was also enforced at the center of the control volumes at subcritical outflow boundaries. This was implemented using a relaxation parameter, instead of simply clamping the water-surface elevation at those triangles. Using the values in Table 6.4, all the fluxes can be computed at the inflow and outflow edges as if they were internal edges.

At solid walls and shorelines, a no-slip condition is used. The velocity is set to zero there and the inviscid fluxes are calculated directly, i.e.,

$$\mathbf{E}_{ik}\Delta l_{ik} = \left[0;\ \frac{1}{2}g(h^-)^2 n_{xk};\ \frac{1}{2}g(h^-)^2 n_{yk}\right]^T \Delta l_{ik}$$

where Δl_{ik} is the length of the wall edge, and n_{xk} and n_{yk} are the components of the normal unit vector pointing towards the outside of the control volume. In the simplest formulation the water depth is set as $h^- = h^+$, but the Riemann invariant can also be used: $h^- = \overline{\vec{u}^+ \cdot \hat{n}_k + 2\sqrt{gh^+}}^{\,2} / 4g$. This approach does not require the solution of the Riemann solver at this edge type, therefore, it is expeditious without losing accuracy.

For inviscid flow computations, free-slip wall boundary conditions are employed, which is done by setting a "ghost" state outside the computational cell with $h^- = h^+$, $u^- = 0$, and $v^- = v^+$, and solving the Riemann problem.

The appeal of unstructured grids resides in the flexibility of discretization of physical domains with complex geometry, and in the ease with which local mesh refinement and adaptation can be implemented. However, compared with traditional structured grids, the application of unstructured grids comes at an increased computational cost. This is due to the natural "lack of structure" suggested by the moniker, in which grid nodes are arbitrarily located over the computational domain, lacking distinct grid directions and structure. Grid nodes are combined into polygons (control volumes), but the number of polygons sharing a particular node usually varies from node to node. Connectivity tables must be established to provide the geometric links between computational nodes, edges, and control volumes necessary to carry out the algebraic integrations resulting from the discretization of the governing equations. Unfortunately, traditional computing machines perform much better if the data in their memory banks are accessed in an orderly, sequential manner. Accessing and storing data in a random way, such as what is usually required in unstructured grid solvers, result in an increased penalty that can be significant [32].

In a preprocessing stage, a reverse Cuthill–McKee reordering of the triangulation of the computational domain is performed [33]. The reordering is applied to the vertices of the triangulation. Then, reordering of the triangles' and edges' numbering is carried out following the numbering of the vertices: Starting at vertex #1, all the triangles (and edges) that share it are numbered sequentially; proceed to vertex #2 and continue numbering the triangles (and edges) that remain unnumbered; in ascending order of vertex number, and sequentially, continue until all the triangles (and edges) have been renumbered. This results in a numbering sequence that substantially reduces the overhead due to indirect memory addressing, as compared to the numbering that results from the triangulation software (i.e., from the automatic grid generators). All the data structures and look-up tables containing the linkages among nodes, edges, and control volumes are built using the newly renumbered computational grid.

Additionally, the solution methods were implemented in a computer program using an edge-based data structure. The use of edge-based data structures leads to codes with reduced computer processing unit (CPU) and memory access overhead when compared to codes that use a more traditional element-based structure. Different types of edges are classified and distributed to different, independent computational loops (dry edges, fully wet edges, partially wet edges, solid boundary edges, inflow and outflow edges, advancing front edges, receding front edges, and bank shoreline edges). The main solution cycles over the edges and the residuals are summed by scattering (antisymmetrically) the fluxes to the control volumes sharing the edge. The use of different cycles for different types of edges allows elimination of data dependencies, which results in highly optimized code in vector–parallel computers. It also allows to fully eliminate conditional statements inside those cycles, therefore, paving the way for an efficient implementation in programmable graphics hardware architectures [34].

4 Deploying the Model

Model deployment is a process with the objective to bridge the gap between model development and model application to real physical systems. This is a broad subject where controversy can easily arise, but here the discussion is limited to only a few of the relevant factors in model dissemination. The most important factors are transparency, time, modularity, and quality. Model transparency addresses the issues involved in transmitting to the user all of the relevant information concerning the model's purpose, formulations, and assumptions or, in other words, model documentation. Model documentation must inform the user about range of applicability and allow critical peer evaluation of the model and its applications. Unfortunately, a unified and definitive modeling terminology does not exist, which can make communication between model developer and user ambiguous and even difficult; therefore, particular care must be taken when producing model documentation. Documentation should be complete and include the mathematical formulation of the model and the assumptions on which it is based; the model's parametrizations and suggested parameter values and ranges, as appropriate; and the operating instructions of the model implementation in its computer software package.

The term "time" refers to the time it takes to complete a modeling project, and the purpose is to deliver a complete study within a reasonably short period of time—say, in months rather than in years. Identifying and automating the most repetitive and time consuming tasks is, therefore, of paramount importance. Data preparation has traditionally been one of the most time-consuming tasks in computer modeling, now of increased importance due to the use of digital terrain models (DTMs). In fact, the use of DTMs has become standard in many geomorphological applications, becoming a tool of choice for the extensive data requirements demanded by a complex model, such as the one described in the previous section. Increasingly available is high-resolution topographic data derived from airborne remote sensing, such as interferometric synthetic aperture radar [35], aerial digital photogrammetry [36], LiDAR [37], and terrestrial laser scanning surveys [38]. With resolutions of 1 m and higher available for even very large areas, a survey may easily contain many million points. It is a challenging task to handle such large amounts of data, while at the same time merging sets from different sources (perhaps in different coordinate systems and storage formats) while keeping a tight quality control.

Another time-consuming task is model calibration. Model calibration is necessary because some quantities cannot be directly measured in the field. An example is channel bed roughness: An indirect measurement of bed friction is instead obtained by measuring the slope of the water free-surface elevation for specific known water discharges. In this case, the calibration process has the objective of finding the correct parameters to use for bed roughness, consisting in a trial-and-error cycle in which the model operator guesses successive values for the roughness parameters until the modeled water surface matches the measured slopes. Other examples of quantities that may need calibration are turbulence parameters and diffusion coefficients. As the number of parameters needing calibration increases, the longer and more tedious the calibration exercise becomes. To streamline the process, it is important to provide quick and easy ways to enter model parameters (many of which are spatially varying) and to analyze computation results.

"Modularity" consists in the ability to use and reuse previously developed work and, possibly, to use and link to the work of others. This requires the definition and use of data transmission and storage standards. These standards should be machine independent, provide durability (the standards should not be temporary and fall from use in a short amount of time), and be widely employed, preferably across multiple disciplines (which also contributes to durability). Another advantage of using standards is the potential disambiguation in data interpretation and the facilitation of quality control. Modularity, therefore, also contributes to reducing model development time and cost, as well as providing a means for project archival.

Finally "quality" refers not only to the data and model but also to the fact that there is the need to produce results of proven quality that can be replicated by others. This objective is better met in a framework that can be used by many and, therefore, that can be applied to a wide range of different models and, preferably, in multidisciplinary projects.

The above factors must be addressed through complex integrated modeling frameworks. One such framework developed specifically for environmental flow modeling is given by the iRIC Project (http://i-ric.org/en/), which provides means to integrate diverse models within the same graphical user interface (GUI) using the same data formats and protocols. The iRIC framework provides operational facilities that are model independent, such as data input and output (multiple formats

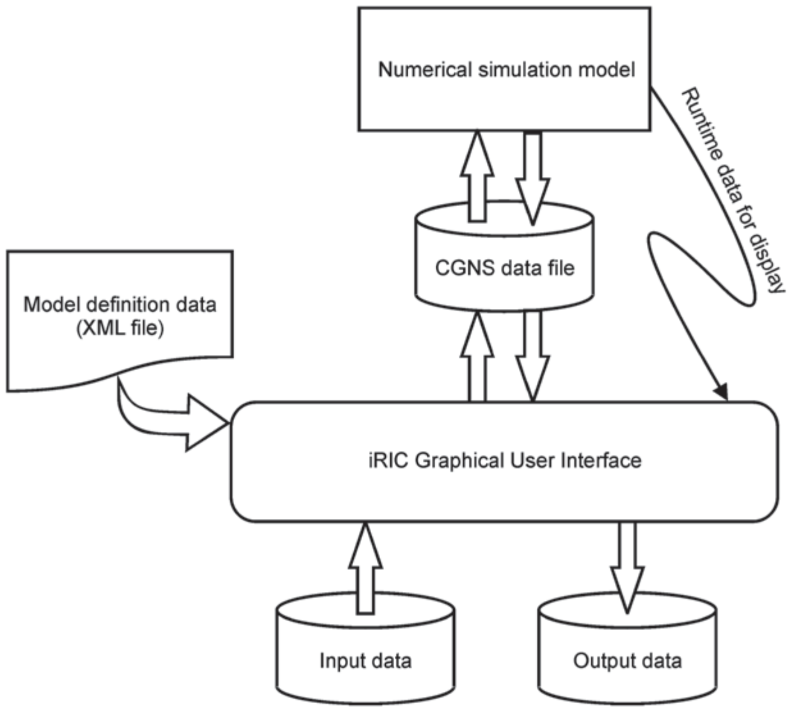
Numerical simulation model
Runtime data for display
CGNS data file
Model definition data (XML file)
iRIC Graphical User Interface
Input data
Output data

Fig. 6.8 Schematic outline of the integration of a numerical model in the iRIC graphical modeling framework

are supported), automatic grid generation, interactive visualization and editing of model input and output, ability to work with ancillary data sets for model calibration, and device-independent plotting. This functionality frees the numerical model from all of these concerns by separating the roles of model developer from those of GUI programmer, with consequential benefits to both.

A schematic view of how the iRIC software works is given in Fig. 6.8. A GUI is used as the user's front-end. The GUI communicates with the model through a device-independent file using a format that has become a standard in many applications of CFD (CGNS, see http://cgns.sourceforge.net/). Runtime information can also be displayed in a console window. The model parameter definitions needed to customize the GUI to the desired model are coded in a flat file in XML format (http://www.w3.org/XML/). The GUI can read data in a multitude of formats commonly used in hydraulics and other DTM applications. Entire GUI setups, including input data, parameter definitions, and model simulation output, can be saved in individual data files for later use, and for transmission and archival. Besides data reutilization, the GUI also provides simpler and shorter training requirements for users desiring to use different models within the same framework.

As an illustrating example, let us consider one of principal modeling preprocessing tasks (i.e., operations needed to be accomplished before the model can be run):

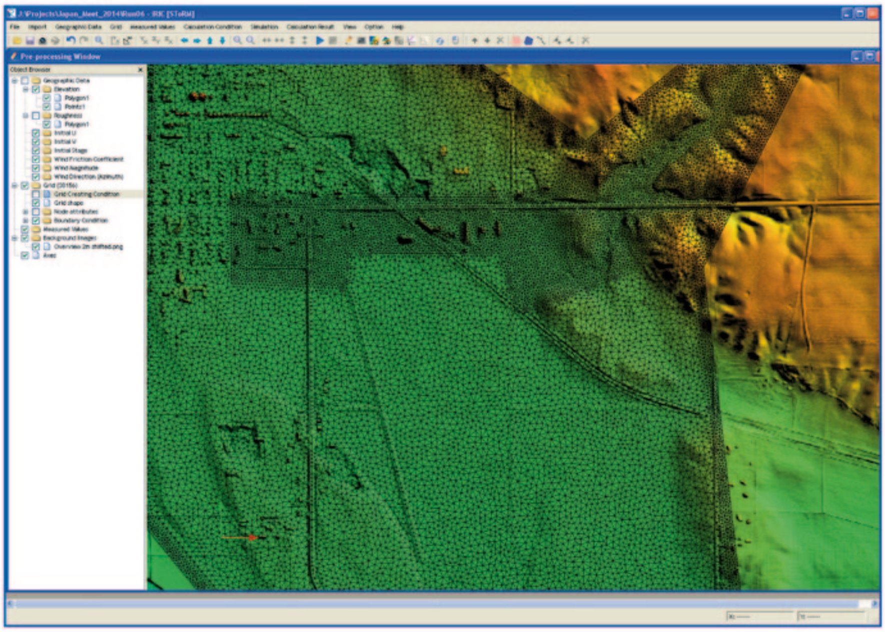

Fig. 6.9 iRIC GUI showing an automatically generated unstructured, triangular computational grid. *GUI* graphical user interface

preparing the computational grid. For the model discussed in the previous sections, this grid is formed by a lattice of nonoverlapping triangles (the control volumes) that must contain the relevant bathymetric information, and the bathymetry is provided by a DTM. The present example concerns a flood inundation study of the LaMoure Rott Municipal Airport in the city of LaMoure, ND, with the purpose of designing flood mitigation measures. The DTM consists of a set of more than 3.7 million points with 2 m of spatial resolution. The iRIC GUI allows automating the process by a few simple and quick clicks of the mouse: Start by importing the data into the GUI; next, define the area of interest where the computational mesh will be created; then enter the desired mean triangle size and click on the automatic mesh generator icon. The result is a quality grid that is automatically mapped to the terrain using the information contained in the DTM, and that is displayed on the computer screen as an overlay to the DTM. The immediate visual feedback permits the user to judge mesh quality and to make adjustments and improvements as desired. Figure 6.9 shows the result of the operation for a grid of 38,156 points (74,565 triangles) zoomed into the airport area. Note the finer grid nested within the coarser main grid, allowing for the more accurate definition of the topography in areas where higher accuracy is needed.

An example of data post-processing (i.e., graphical visualization of model results) is shown in Fig. 6.10, where the water depth contour levels are plotted over topography in the same region of Fig. 6.9. Multiple types of spatial plots (contour

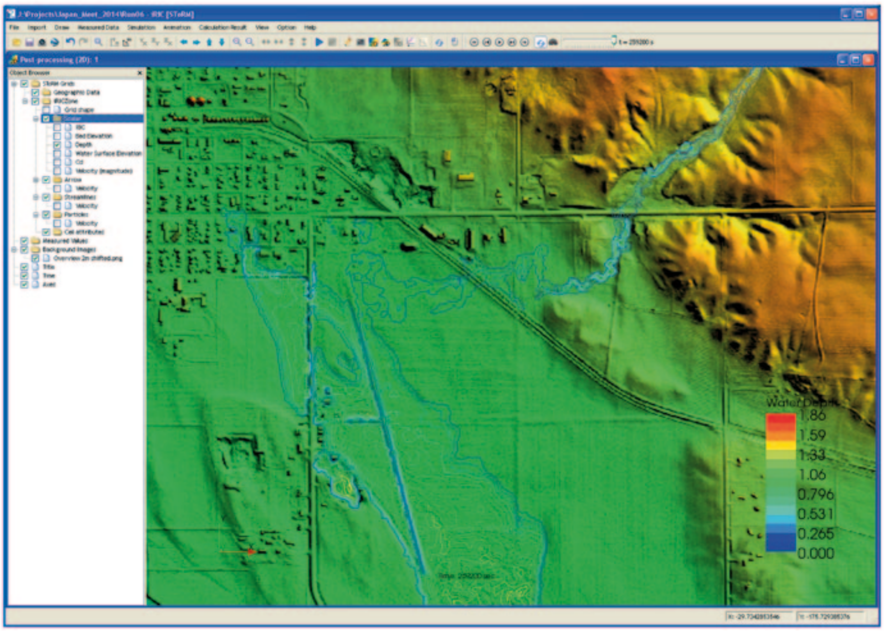

Fig. 6.10 Interactive display of computational modeling simulation results. Shown are the contour levels of water depth, colored using the color coding shown in the legend located in the lower right corner of the display

lines, color maps, vectors, etc.) allow for rapid, but detailed interactive analysis of model results and suggest areas where improvements can be made, reducing simulation production times and improving quality control. They also substantially decrease the production times of project deliverables, such as documentation and visual presentation aids, and provide a useful tool to communicate technical information to those in the management- and policy-making disciplines.

5 Model Validation Applications

Unfortunately, a model quality assurance methodology has not been established by the hydraulic modeling community. Some efforts have been made, from which terms such as model *verification*, model *validation*, and model *confirmation* have surfaced. Although universally accepted terms do not exist, Refsgaard and Henriksen [7], in large part commenting and citing the work of Oreskes and others [39], provide the following definitions:

> Verify is "an assertion or establishment of truth." To verify a model therefore means to demonstrate its truth. According to the authors "verification is only possible in closed systems in which all the components of the system is established independently and are known

> to be correct. In its application to models of natural systems, the term verification is highly misleading. It suggests a demonstration of proof that is simply not accessible." […] mathematical components are subject to verification, because they are part of closed systems, but numerical models in application cannot be verified because of uncertainty of input parameters, scaling problems and uncertainty in observations.
> The term validation is weaker than the term verification. Thus validation does not necessarily denote an establishment of truth, but rather "the establishment of legitimacy, typically given in terms of contracts, arguments and method." […] "the term valid may be useful for assertions about a generic model code but is clearly misleading if used to refer to actual model results in any particular realization."
> The term confirmation is weaker than the terms verification and validation. It is used with regard to a theory, when it is found that the theory is in agreement with empirical observations. […] such agreement does not prove that the theory is true, it only confirms it.

Applying these concepts to an instance of computer software is not trivial, but some ideas may be retained. For example, "model validation" refers to whether the model has the intended accuracy when applied to processes consistent with its designated range of application; "code verification" indicates if the model code is a true representation of the conceptual model that was used for its design; and "model confirmation" concerns the indication whether the scientific theories and hypothesis used by the conceptual model provide an adequate level of agreement to the intended real-world applications. In other words, validation checks if the right equations are being solved, while verification checks if the equations are being solved right.

Validation of a hydraulic model can be accomplished in several ways. One way is to compare the results produced by the model with the results obtained by another model applied to the same problem. This is a valid approach if the latter model has been previously verified and validated for that type of phenomenon and that range of physical parameters. A second approach is to compare model results with those obtained in a laboratory experiment. The physical and design conditions can be carefully controlled and the flow variables accurately measured in a laboratory setting, thus providing reliable data for model validation in which a scaled-down version of the real world is employed. Finally, model results can be compared with those measured *in loco*. This latter approach is the most costly and difficult, but is also the one that may produce the most desirable outcomes, because it provides data collected in the physical ambient for which the model was designed. It is not without problems: Data collection is difficult and expensive, error-free data do not exist, and real-world environments are open systems, exacerbating errors due to parameter calibration and shortcomings in model structure. This section presents examples of all three approaches to model validation.

5.1 Supercritical Flow

Supercritical flows do not often appear in natural channels, but are not uncommon in engineering works, such as spillways and flow-measuring structures. In particular, supercritical flow may occur in channels with nozzle-like constrictions, such as the Parshall flume, where steady oblique shocks are observed to develop under

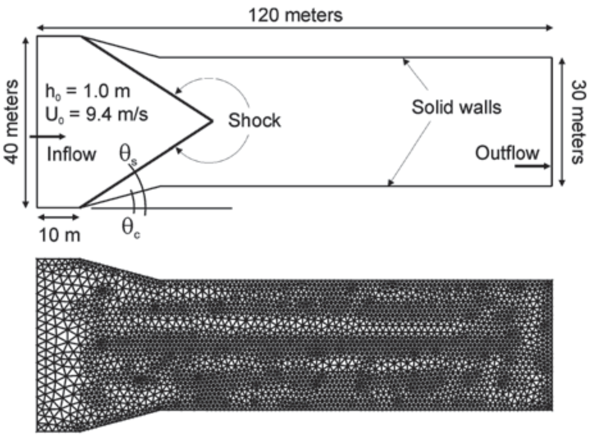

Fig. 6.11 Channel dimensions (*top*) and coarse mesh setup (*bottom*) for the symmetric contracting channel used. Flow is from *left* to *right*

appropriate conditions with supercritical flow. The jump ratio across the shock and its angle can be derived from the shock relations for the shallow-water equations and are given by

$$\frac{h_1}{h_0} = \frac{\tan\theta_s}{\tan(\theta_s - \theta_c)} \quad \text{and} \quad \sin\theta_s = \sqrt{\frac{h_1}{2h_0 Fr^2}\left(1+\frac{h_1}{h_0}\right)}$$

where h_0 and h_1 are the depths before and after the shock, Fr is the Froude number of the approaching flow, θ_c is the flume's contraction angle (see Fig. 6.11) and θ_s is the angle of the shock (measured the same way as θ_c). This type of configuration is common in the literature and the case of [40] was chosen for this application of the model.

The setup used consists of a rectangular, straight 120-m long channel with an approaching width of 40 m and a symmetric constriction with $\theta_c = 15^\circ$, reducing the width of the channel to 30 m, as depicted in Fig. 6.11. Uniform supercritical flow is prescribed at the inflow boundary with $Fr = 3.0$ and $h_0 = 1.0$ m. With these conditions, the theoretical values for the shock are $h_1 = 1.9459$ m and $\theta_s = 34.3560^\circ$.

The computations were carried with two different computational grids: a coarse grid with 2997 nodes (5704 triangles, as depicted at the bottom of Fig. 6.11) and a finer grid with 15,895 node points (31,248 triangles, not shown). At the inflow boundary, the velocity and the water depth are prescribed ($U = U_0 = 9.3960$ m/s, $V = 0$ m/s, and $h_0 = 1.0$ m). At the outlet, no conditions are needed because the flow is supercritical. The initial conditions are the same as the inflow boundary conditions and a free-slip treatment is used at the solid walls. The inviscid flow equations are solved, without bottom friction or slope, until steady state is reached. The computational results are shown in Fig. 6.12.

The solution presents well-defined steady oblique shocks, starting at the wall points where the constriction begins and propagating downstream. The predicted

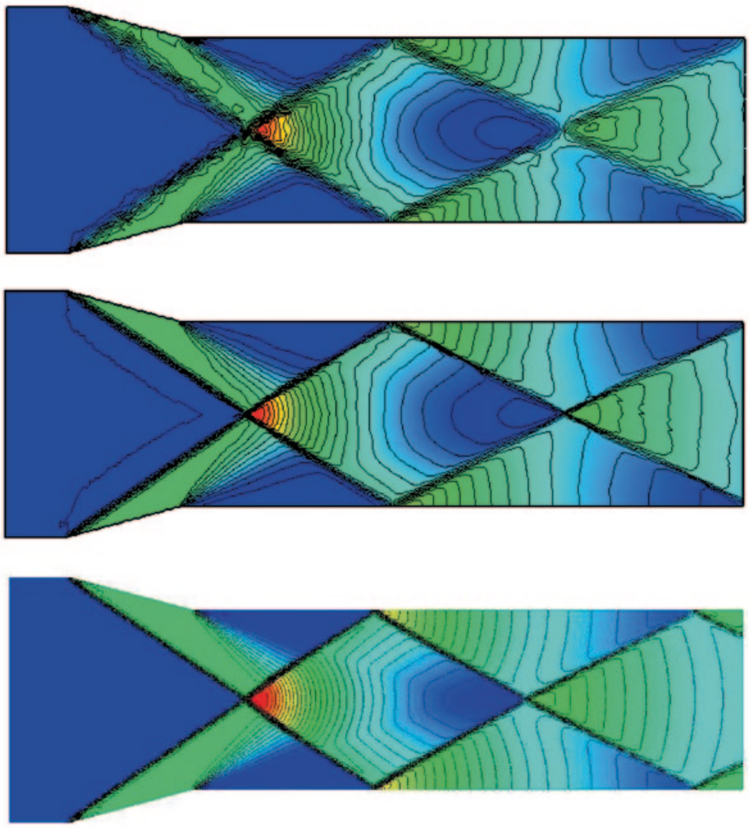

Fig. 6.12 From *top* to *bottom*: computed solution using the *coarse grid*; computed solution using the *fine grid*; reference solution of [40]. Colors represent water depth, *dark blue* is shallow water ($h = 1.0$ m) and *red* is deep water ($h = 3.1$ m).

angle of the shock and the water depth downstream from it are $\theta_s = 34.3487 \pm 0.10°$ and $h_1 = 1.9430 \pm 0.05$ m, which match very well the theoretical values presented above. For comparison, the solution obtained by [40] in a mesh with 23,349 computational cells is also presented.

5.2 Recirculation Past a Spur Dike

Groynes, spur dikes, and other structures are river-training structures used to divert the flow and protect the river banks from erosion. They are also used to improve navigability and to maintain river alignment. Rajaratnam and Nwachukwu [41] presented detailed measurements of flow velocity and bed shear stress in several experimental configurations. Their experiment A1 was chosen as a means to provide a comparison between model predictions and physical data obtained in a controlled

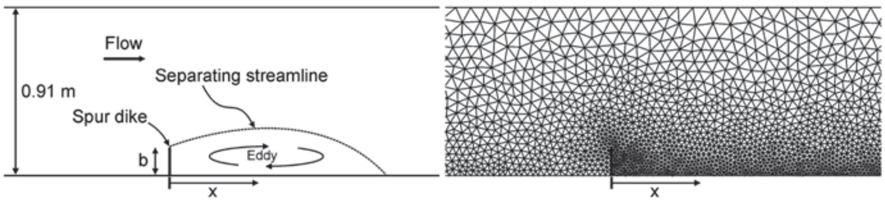

Fig. 6.13 General flow configuration past the spur dike in experiment A1 of [41] and detail of the computational mesh used in the same area

laboratory setting. The same experiment was used by Tingsanchali and Maheswaran [42], who used the TEACH code with an elaborate k–ε turbulence model. In this application, however, a simple constant eddy viscosity turbulence model is used.

The experimental configuration and computational setup are presented in Fig. 6.13. The experimental tilting flume used was 37 m in length and 0.91 m in width. The spur dike consisted of a 3-mm thin plate placed across the flow, perpendicularly, with a length of $b = 0.152$ m. The slope of the channel was set in order to obtain a uniform flow depth of 0.189 m in the regions away from the constriction (both upstream and downstream). This value was used as the downstream boundary condition. The channel had smooth walls and bottom, corresponding to an approximate value of the Manning's roughness n of 0.010.

The inflow boundary condition was set to the experimental discharge $Q = 0.04531$ m^3/s and a no-slip treatment was used for the solid walls and spur dike. The value of the eddy viscosity was set to $\nu_t = 0.001$ m^2/s, but there was no attempt to calibrate this value. Grid-converged solutions were obtained with a mesh of 7693 nodes (14,467 triangles), which was partially refined in the regions around and downstream from the plate (see Fig. 6.13). A detailed view of the computed eddy streamline pattern near the plate is shown in Fig. 6.14, superimposed on the contour lines of the water depth.

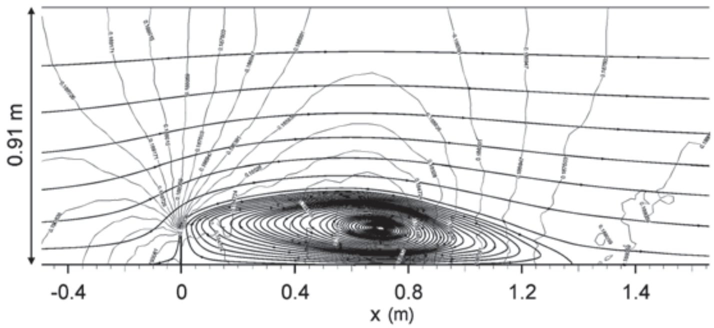

Fig. 6.14 Streamlines of the eddy formed downstream from the spur dike. The contour lines of the water depth are also shown

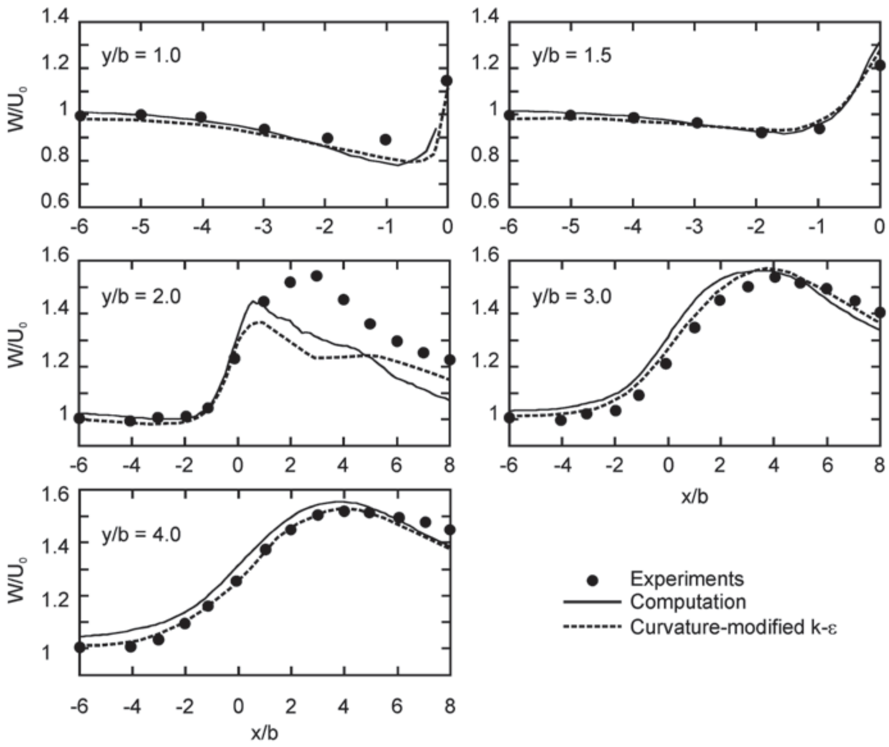

Fig. 6.15 Comparison between the computed velocity profiles (*solid line*), the calculated values of [42] (*dashed line,* only the results of the enhanced turbulence model are shown), and the experimental results (*solid circles*) for experiment A1 of [41]

The reattachment point of the eddy is located at $x/b = 9.175$. The experimental value is of approximately 12.5, resulting in an underprediction of 26.6 %. This result compares well with that of [42], which underpredicts the eddy size by 40 % using the standard k–ε model. Only after a streamline curvature correction factor was applied to the turbulence model did these authors improve their prediction, but their result still underestimated eddy length by 20 %. The maximum width of the eddy is $y/b = 1.824$, which is 9 % less than the experimental value of approximately 2.

Comparisons between the computed and experimental total velocities ($W = \sqrt{u^2 + v^2}$) are shown in Fig. 6.15 for several sections measured along lines of constant y. The results shown are normalized with the approaching velocity, U_0, and the plate length b. Unfortunately, there are no measurements inside the eddy region, but the results show an overall good agreement comparable to the results obtained by [42]. The poorest results are in the region $1 < x/b < 8$ at the transect $y/b = 2.0$. This is a region near the top of the eddy. The poorer predictions there do not seem to affect substantially the computed velocities in the same region of the next transect (at $y/b = 3.0$).

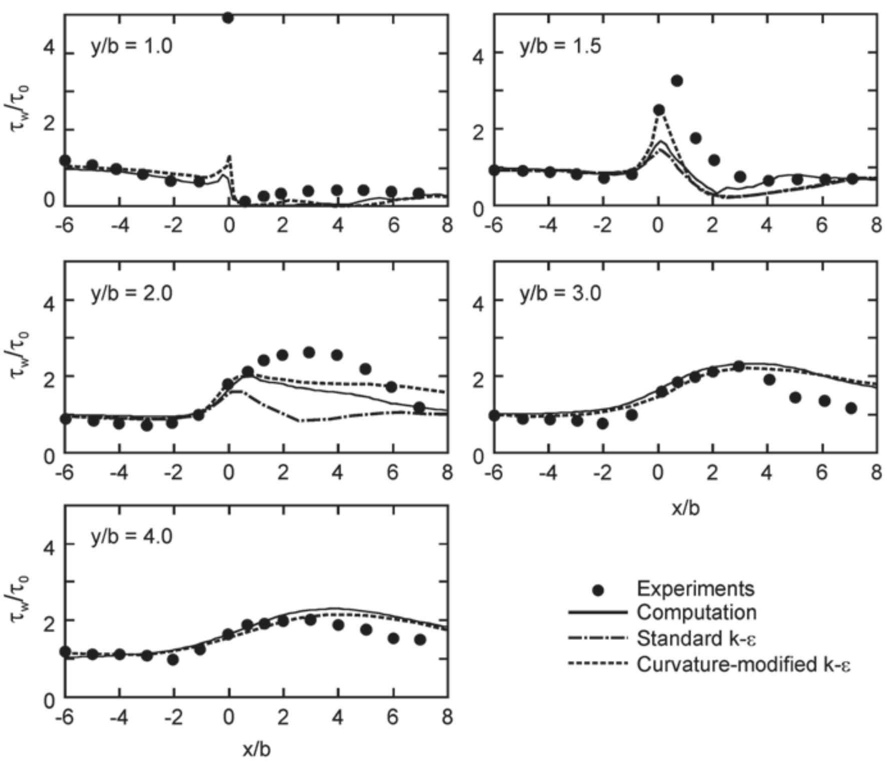

Fig. 6.16 Comparison between the computed bed shear stress profiles (*solid line*), the calculations of [42] (*dashed lines*), and the experimental results (*solid circles*) for experiment A1 of [41]

The presentation of the bed shear stresses is important because it indicates the quality of how flow resistance is modeled. Its correct computation is also important for sediment transport predictions. The computational values of the total bed shear stress, $\tau_w = \rho g n^2 W^2 / h^{1/3}$, are shown in Fig. 6.16 normalized with the approaching bed shear stress τ_0. Once again, the overall computation results agree very well with the measurements, except in the vortex regions downstream from the spur dike, where the 3D effects are significant, such as in the region $1 < x < 6$ for the transect located at $y/b = 2.0$. Nonetheless, these results compare favorably to those in [42], being substantially better than the results obtained by the standard *k*–ε model, and marginally better than the modified *k*–ε model in some regions.

It is interesting to note that the k–ε turbulence model does not seem to produce considerably better results than the algebraic turbulence approach used in this work, in spite of the significant increase in computational cost of the former model, which requires the solution of two additional partial differential equations. It is not clear why this is so. It may be due to a breakdown of the assumptions made in deriving the 2D k–ε model, which may be valid for 3D modeling but may lose generality in shallow 2D flows. Or it just may happen that bed friction (or shear velocity) is all one has to consider in order to obtain an accurate enough evaluation of the effects of turbulence dissipation on the main flow quantities.

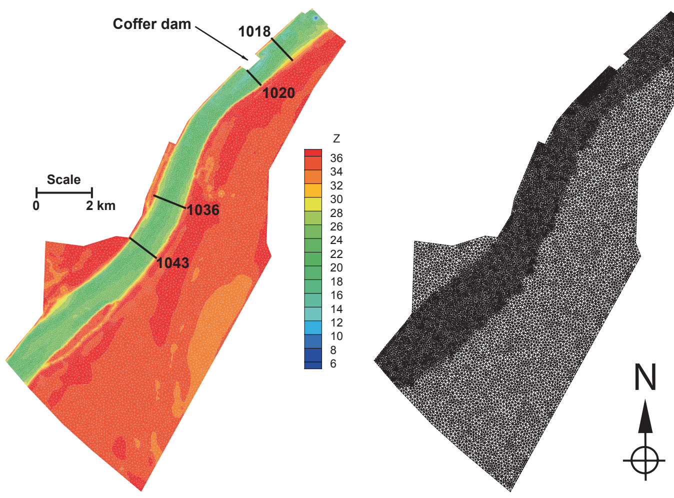

Fig. 6.17 Bathymetry (*left*) and computational mesh (*right*) used in the numerical computations. At left, the measurement transects used for verification are shown. For consistency and easy reference, the number designation of those cross sections was kept identical to the designation assigned in the data collection program. The colorization shows the bed elevation above an arbitrary datum (*Z*, in meters)

5.3 Application to a Large River

The application of hydraulic models to real-world applications is of practical importance and constitutes the final measure of the robustness and usefulness of the techniques used. The challenges posed by the scales and geometries normally encountered in environmental flows provide the best test bed for any numerical model and exercises the full range of the algorithms implemented in the computer code. For this application test case, data collected in a 15-km reach of the Ohio River near the US Army Corps of Engineering Olmsted Locks and Dam project, at river mile 964.4 are used. The data were presented in [43] and were collected using acoustic Doppler current profilers with a differentially corrected global positioning system with the purpose to provide data that could be used to calibrate and validate numerical models. The data consist of detailed bathymetry, water-surface elevations, and velocity profiles collected with known river discharges. For this study, the data corresponding to the discharge of approximately 9900 m^3/s are used.

The overall bathymetry is shown in Fig. 6.17. At the time of the data collection, a coffer dam was in place, located at the upstream end of the reach. The flow is from northeast to southwest. A computational mesh containing 10,288 nodes (20,101 triangles) was developed based on the bathymetry, using larger triangles in the flood plains, with refinements near the upstream and downstream ends of the coffer dam, and shown in the right-hand side of Fig. 6.17. The downstream water-surface elevation supplied in [43] was used for the downstream boundary, and the known discharge was prescribed at the upstream boundary.

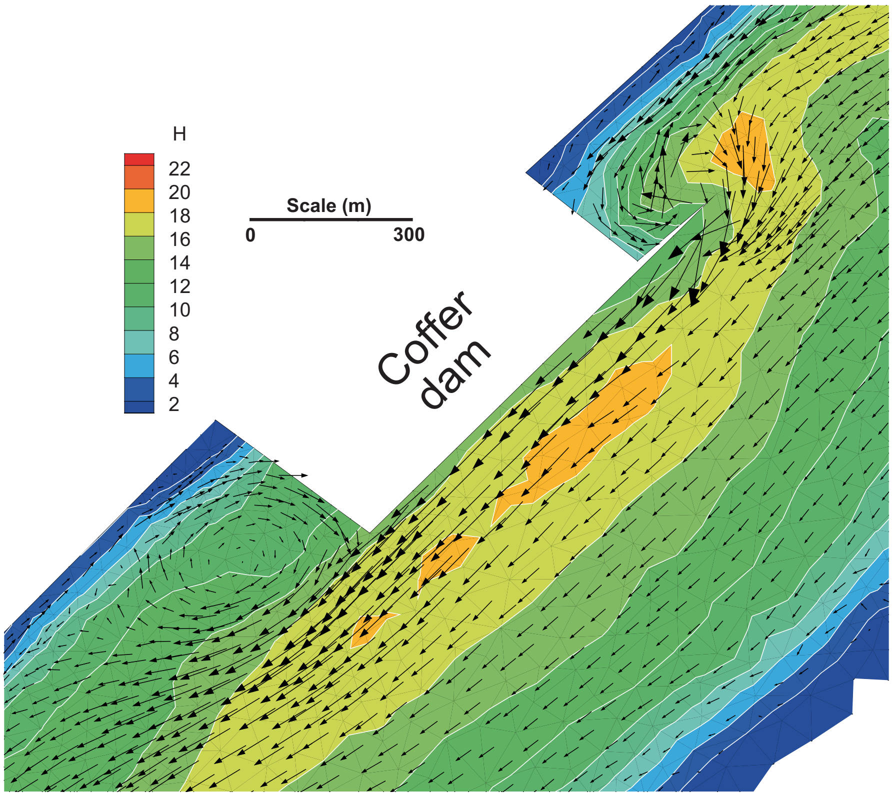

Fig. 6.18 Detailed view of the flow solution near the coffer dam. The colors indicate water depth (*H,* in meters)

The values of the bed roughness were found by calibration. The value of the water-surface elevation at the downstream corner of the coffer dam is known. Several values of the Manning's roughness *n* were tried until a good match was found. The final value chosen was $n=0.020$, which was prescribed uniformly throughout the reach. The value of the turbulent eddy viscosity was prescribed uniformly throughout the computational domain to $\nu_t=0.001\ \mathrm{m}^2/\mathrm{s}$.

The results of the computation are shown in Figs. 6.18 and 6.19. Figure 6.18 shows the known areas of recirculation around the coffer dam. The downstream eddy is smaller than the observed one by nearly 40%. The underprediction of the eddy sizes was expected, as shown in the previous validation exercise, but not by such a large amount. The simple turbulence model used has shown to provide solutions that are more accurate than this; therefore, the magnitude of this discrepancy may be attributed instead to an incorrect localized roughness value, or to the inflow boundary condition being too close to the region of interest, or both. Unfortunately, the lack of field data on the nature of the bed roughness in that region precludes a more elaborate analysis. Nonetheless, it is worthwhile to point out that the simulations carried out in [43] used different values of *n* for the banks and for the main channel.

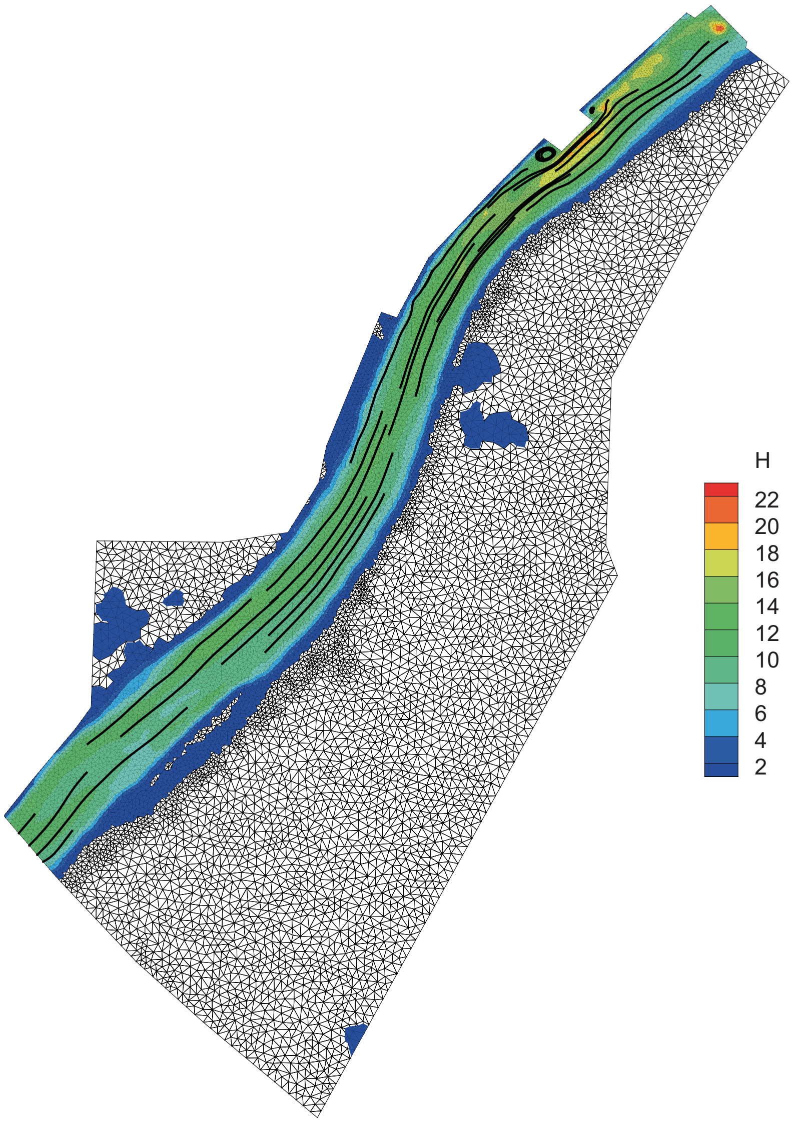

Fig. 6.19 Streamlines of the solution showing a smooth flow without oscillations. The wetted domain is shown colorized by water depth (*H,* in meters). The dry triangles are shown in black and white

The wetted areas resulting from the computation are shown in Fig. 6.19. Some of the wet areas in the flood plains resulted from the initial conditions, which were set with a high initial water-surface elevation. This resulted in some of the adjacent low-lying areas to be initially wet. During the course of the computation, water flows to the river leaving wetted ponds behind. These ponds are disconnected from

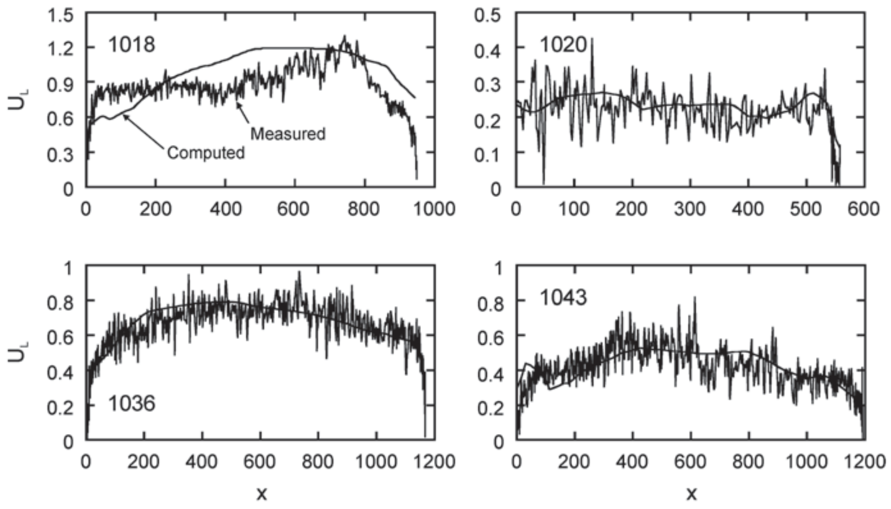

Fig. 6.20 Comparison between computed and measured longitudinal velocity profiles at selected locations

the river's main stem. These effects could have been avoided by using a much lower initial water-surface elevation, but convergence to steady state would have taken a substantially longer time.

The location of the shoreline shows a good agreement with the survey data. The measured velocity profiles taken at the transects shown in Fig. 6.17 were used to provide comparison with the corresponding profiles of the numerical solution. The results are shown in Fig. 6.20, which displays the longitudinal component of the velocities, U_L. Overall, there is good agreement between measurements and computation, with the computed velocities falling well within the band of the measurement variability. The exception resides at transect 1018, where larger differences between measurements and computation are present. This is attributed to the way the upstream boundary conditions are enforced: The total discharge is enforced by generating a velocity profile with uniform unit discharge and whose velocities are normal to the boundary line. The velocity components of the profile thus generated may be (and most likely are) substantially different from the field conditions in that region of the flow, adversely impacting the predicted velocities at transect 1018, which is located less than three channel widths away from the inflow boundary. Effects from boundaries may be seen up to 10 channel widths away from the boundary and it is suggested that, when possible, computational boundaries be placed at a large distance from critical areas. In the present situation, unavailability of bathymetric data and the fact that the modeled area is downstream from a river bend, precluded extending the modeling reach in the upstream direction in a meaningful manner. However, the flow recovers rather quickly and a good agreement between the computed and the measured longitudinal velocities seems to have already been established near the coffer dam (transect 1020).

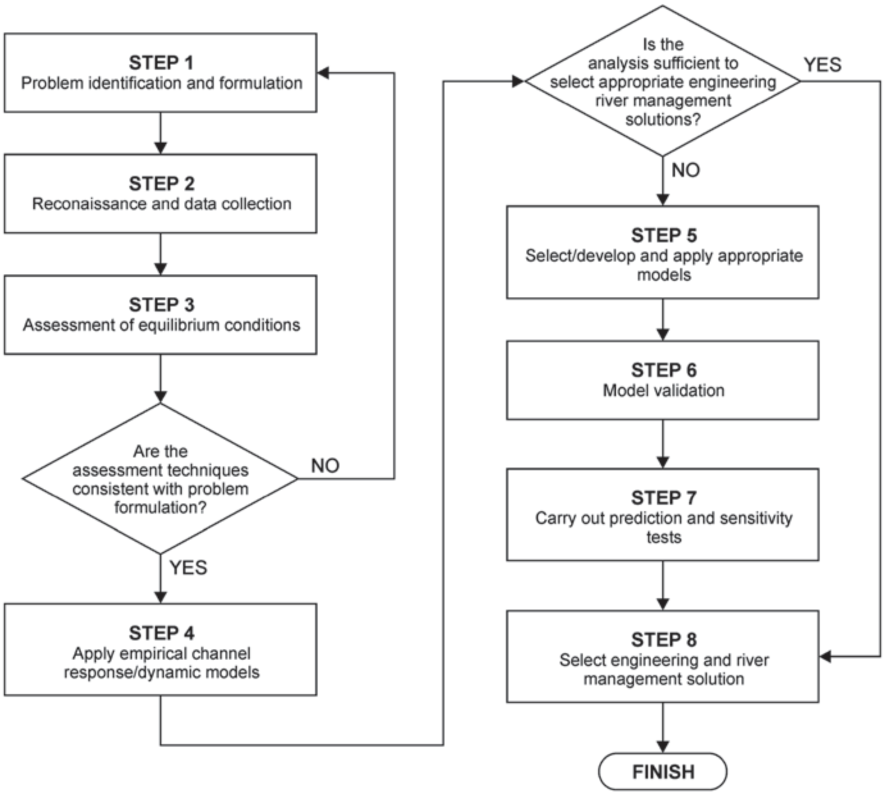
STEP 1
Problem identification and formulation
STEP 2
Reconaissance and data collection
STEP 3
Assessment of equilibrium conditions
Are the assessment techniques consistent with problem formulation?
NO
YES
STEP 4
Apply empirical channel response/dynamic models
Is the analysis sufficient to select appropriate engineering river management solutions?
YES
NO
STEP 5
Select/develop and apply appropriate models
STEP 6
Model validation
STEP 7
Carry out prediction and sensitivity tests
STEP 8
Select engineering and river management solution
FINISH

Fig. 6.21 Procedure for the identification, analysis, and modeling of river width adjustment problems, after [44], with modifications

6 Concluding Remarks

From the expositions presented in the previous sections, it has been made clear that the task of modeling environmental flows in surface hydraulics is not a simple one—certainly not as simple as just running a software program. Like in many other areas of application, the complexity and variety of geomorphic factors involved in natural phenomena require a careful and methodical approach to their modeling, with large support of field observations and prototype data. To deal with these issues, an eight-step approach is proposed in [44]. In spite of being specific to bank evolution modeling, the basic principles of these guidelines are applicable to other areas of interest in hydraulic modeling. A schematic view of the entire process is presented in Fig. 6.21.

Step #1, problem identification and formulation involves determining the factors at play, which may be associated with river engineering factors or social activities, and may relate to existing conditions or future activities. It establishes the level of analysis required and defines the appropriate level of response. It is followed by step #2, field reconnaissance and data collection, in order to identify the nature and extent of the problem. In this stage, channel characteristics are identified, bank conditions and materials, hydrologic conditions, vegetation, any engineering works, and any other parameters that are considered relevant for the case at hand. During this stage, there is an assessment of the existing data available and, if necessary, it is the time to design and mount an adequate data gathering program.

At this stage, a simple assessment of the equilibrium conditions using the techniques of regime theory (step #3) is recommended. The results of this analysis are compared with existing conditions to provide an indication of the present morphological status of the prototype. This allows the determination of the impact of the proposed engineering works.

In step #4, simple empirical channel response models are applied. This exploratory step may help to interpret existing and proposed conditions, and in identifying the dominant processes and trends at play. The information gathered at this stage forms a framework for the more detailed modeling work that follows (if appropriate). In step #5, more advanced models are developed and used, if necessary, to provide the more detailed information. These models should be validated with existing prototype data (step #6), and applied to current conditions and also to assess the impacts of the proposed engineering works (step #7). At this stage, a sensitivity analysis involving all the pertinent parameters is important, with particular emphasis in the parameters that are difficult to determine or that have significant spatial and/or temporal variability. In step #8, all the information gathered in the previous steps is used to formulate and implement the appropriate plan of action.

In practice, environmental real-world problems are complex and often require a multidisciplinary approach covering diverse aspects of the natural sciences. There is no clear-cut methodology that fits all practical cases, but the above analysis offers a rational and systematic approach that can be useful in dealing with large and small modeling projects. In the end, however, practical experience and good engineering judgment remain the fundamental tools for good decision making.

Acknowledgments The author is grateful to Chad Wagner, USGS, Louisville, KY, for kindly providing the bathymetry and experimental measurements used for the Olmsted Lock and Dam case, and for his assistance in data preparation and interpretation. The model developed and presented in this work (System for Transport and River Modeling, SToRM) was supported by the National Research Program of the US Geological Survey. The SToRM model and corresponding documentation are available for free download from the iRIC Project Web pages at http://i-ric.org/en/.

Glossary

CFD Short for computational fluid dynamics, it is a discipline that uses numerical methods and algorithms to solve fluid mechanics problems with computers.

CGNS A standard for the storage and retrieval of digital data produced in CFD applications. It stands for CFD general notation system.

Computational cell Each individual point, or volume, of the lattice (computational grid) transforming the continuous real-world domains into its discrete counterpart, suitable for numerical evaluation and implementation on digital computers.

Digital terrain model Also called a digital elevation model (DEM), it is a 3D digital representation of a terrain's surface.

Eddy viscosity A method to model the transfer of momentum caused by turbulent eddies that is mathematically similar to momentum transfer due to molecular diffusion, and that consists in replacing the fluid viscosity ν by an effective turbulent viscosity, ν_t, also called the eddy viscosity.

Froude number *(Fr)* A dimensionless quantity defined as the ratio of a characteristic velocity to a gravitational wave velocity.

Godunov scheme A conservative finite-volume numerical scheme used in the solution of partial differential equations, which solves exact or approximate Riemann problems at inter-cell boundaries.

Graphical user interface Type of computer-user interface that allows the user to interact with a computer program using pointing hardware devices, graphical icons, and other visual indicators.

Hydrology The study of flow of water over the Earth's surface

k–ε model A turbulence model based on solving two differential transport equations, one for the turbulent kinetic energy k and the other for the rate of turbulent dissipation ε.

Model A idealized representation of a system.

MUSCL Short for modified upwind scheme for conservation laws, it is a method to describe (reconstruct) the states of the variables in a computational cell based on the cell-averaged states (and their gradients) obtained in the previous time step.

Navier–Stokes equations Partial differential equations arising from Newton's second law (conservation of momentum) that describe the motion of fluids.

Runge–Kutta methods A family of implicit and explicit iterative methods used in temporal discretization for the approximation of solutions of ordinary differential equations.

Supercritical flow A flow whose velocity is larger than the wave velocity, therefore with $Fr > 1$.

TVD Total variation diminishing (TVD) is a property of certain discretization schemes used to solve hyperbolic partial differential equations that do not increase the total variation of the solution from one time step to the next.

References

1. Kirby, M. J. (1987). Models in physical geography. In M. Clark et al. (Eds.), *Horizons in physical geography.* (pp. 47–61). Totowa: Barnes and Noble.
2. Bird, R., Stewart, W., & Lightfoot, E. (1960). *Transport phenomena*. New York: Wiley.

3. Formentin, S., & Zanuttigh, B. (2013). *Prediction of wave transmission through a new artificial neural network developed for wave reflection*. Proceedings of the 7th International Conference on Coastal Dynamics, Arcachon, France, 24–28 June 2013, pp. 627–638.
4. Azamathulla, H., & Zahiri, A. (2012). Flow discharge prediction in compound channels using linear genetic programming. *Journal of Hydrology, 454–455*, 203–207.
5. Biswas, G. (2013). *Computational fluid dynamics*. Pangbourne: Alpha Science.
6. Rosenquist, T., & Story, S. (2012). Using the Intel Math Kernel Library (Intel MKL) and Intel compilers to obtain run-to-run numerical reproducible results. *The Parallel Universe*, no. 11, September 2012, pp. 26–28 Intel Corporation.
7. Refsgaard, J., & Henriksen, H. (2004). Modeling guidelines—terminology and guiding principles. *Advances in Water Resources, 27,* 71–82.
8. EPA. (2009). *Guidance on the development, evaluation, and application of environmental models*. U.S. Environmental Protection Agency, Council for Regulatory Environmental Modeling, EPA/100/K-09/003, March 2009.
9. Tennekes, H., & Lumley, J. (1972). *A first course in turbulence*. Cambridge: MIT Press.
10. Rodi, W. (1993). *Turbulence models and their application in hydraulics*. IAHR monograph. Roterdam: Balkema.
11. Pritchard, D. W. (1971). *Hydrodynamic models*. Estuarine models: An assessment (pp. 33). Tracor, Inc, for the Water Quality Office of the Environmental Protection Agency.
12. Shimizu, Y., Yamaguchi, H., & Itakura, T. (1991). Three-dimensional computation of flow and bed deformation. *Journal of Hydraulic Engineering, 116*(9), 1090–1108.
13. Vreugdenhil, C. (1994). *Numerical methods for shallow-water flow*. Boston: Kluwer.
14. Fischer, H. B., List, E. J., Koh, R. C. Y., Imberger, J., & Brooks, N. H. (1979). *Mixing in inland and coastal waters*. San Diego: Academic.
15. Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics, 43,* 357–372.
16. Toro, E. F. (2001). *Shock-capturing methods for free-surface shallow flows*. Chichester: Wiley.
17. Alcrudo, F., & Garcia-Navarro, P. (1993). A high-resolution Godunov-type scheme in finite volumes in 2D shallow water equations. *International Journal for Numerical Methods in Fluids, 16,* 489–505.
18. Harten, A., & Hyman, J. M. (1983). Self adjusting grid methods for one-dimensional hyperbolic conservation laws. *Journal of Computational Physics, 50,* 235–269.
19. 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.
20. Barth, T. J., & Jespersen, D. C. (1989). *The design and application of upwind schemes on unstructured meshes*. AIAA paper AIAA-89-0366.
21. Venkatakrishnan, V. (1995). Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics, 118,* 120–130.
22. Rausch, R., Batina, J., & Yang, H. (1991). *Spatial adaptation procedures on unstructured meshes for accurate unsteady aerodynamic flow computation*. AIAA Paper 91-1106.
23. Batten, P., Lambert, C., & Causon, D. M. (1996). Positively conservative high-resolution convective schemes for unstructured elements. *International Journal for Numerical Methods in Engineering, 39,* 1821–1838.
24. Coirier, W. J. (1994). *An adaptively-refined, Cartesian, cell-based scheme for the Euler and Navier-Stokes equations*, Ph.D. dissertation, Dept. of Aerospace Engineering, Univeristy of Michigan, MI.
25. Bradford, S. F., & Sanders, B. F. (2002). Finite-volume method for shallow water flooding of arbitrary topography. *Journal of Hydraulic Engineering. ASCE, 128*(3), 289–298.
26. Gottlieb, S., Shu, C.-W., & Tadmor, E. (2001). Strong stability-preserving high-order time discretization methods. *SIAM Review, 43*(1), 89–112.
27. Swanson, R. C., & Turkel, E. (1997). *Multistage schemes with multigrid for Euler and Navier-Stokes equations, components and analysis*. NASA technical paper 3631, Langley Research Center, Hampton, VA.

28. Jameson, A., & Mavriplis, D. (1986). Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. *AIAA Journal, 24*(4), 611–618.
29. Horritt, M. S. (2002). Evaluating wetting and drying algorithms for finite element models of shallow water flow. *International Journal for Numerical Methods in Engineering, 55,* 835–851.
30. Balzano, A. (1998). Evaluation of methods for numerical simulation of wetting and drying of shallow water flow models. *Coastal Engineering, 34,* 83–107.
31. Anastasiou, K., & Chan, C. T. (1997). Solution of the 2D shallow water equations using the finite volume method on unstructured triangular meshes. *International Journal for Numerical Methods in Fluids, 24,* 1225–1245.
32. Löhner, R., & Galle, M. (2002). Minimization of indirect addressing for edge-based field solvers. *Communications in Numerical Methods in Engineering, 18,* 335–343.
33. George, A., & Liu, J. (1981). *Computer solution of large sparse positive definite systems*. Computational mathematics. Englewood Cliffs: Prentice-Hall.
34. Dokken, T., Hagen, T. R., & Hjelmervik, J. M. (2007). An introduction to general purpose computing on programmable graphics hardware. In G. Hasle, et al. (Eds.), *Geometric modeling, numerical simulation, and optimization: Industrial mathematics at SINTEF* (pp. 123–161). Springer.
35. Tarpanelli, A., Brocca, L., Melone, F., & Moramarco, T. (2013). Hydraulic modelling calibration in small rivers by using coarse resolution synthetic aperture radar imagery. *Hydrological Processes, 27,* 1321–1330.
36. Gallegos, H. A., Schubert, J. E., & Sanders, B. F. (2009). Two-dimensional, high-resolution modeling of urban dam-break flooding: A case study of Baldwin Hills, California. *Advances in Water Resources, 32,* 1323–1335.
37. Hunter, N. M., Bates, P. D., Neelz, S., Pender, G., Villanueva, I., Wright, N. G., Liang, D., Falconer, R. A., Lin, B., Waller, S., Crossley, A. J., & Mason, D. C. (2008). Benchmarking 2D hydraulic models for urban flooding. *Water Management, 161,* 13–30.
38. Williams, R. D., Brasington, J., Vericat, D., & Hicks, D. M. (2014). Hyperscale terrain modelling of braided rivers: Fusing mobile terrestrial laser scanning and optical bathymetric mapping. *Earth Surface Processes and Landforms, 39,* 167–183.
39. Oreskes, N., Shrader-Frechette, K., & Belitz, K. (1994). Verification, validation and confirmation of numerical models in the earth sciences. *Science, 263*(5147), 641–646.
40. Cueto-Felgueroso, L., Colominas, I., Fe, J., Navarrina, F., & Casteleiro, M. (2006). High order finite volume schemes on unstructured grids using moving least squares reconstruction. Application to shallow water dynamics. *International Journal for Numerical Methods in Engineering, 65*(3), 295–331.
41. Rajaratnman, N., & Nwachukwu, B. A. (1983). Flow near groin-like structures. *Journal of Hydraulic Engineering. ASCE, 109*(3), 463–480.
42. Tingsanchali, T., & Maheswaran, S. (1990). 2-D depth-averaged flow near groyne. *Journal of Hydraulic Engineering. ASCE, 166*(1), 71–86.
43. Wagner, C. R., & Muller, D. S. (2002). Use of velocity data to calibrate and validate two-dimensional hydrodynamic models. Proceedings of the Second Federal Interagency Hydrologic Modeling Conference, Las Vegas, NV, July 29–August 1, 2002.
44. ASCE (1998). River width adjustment. II: Modeling. *Journal of Hydraulic Engineering. ASCE, 124*(9), 903–917.