Automatic sensitivity analysis of a finite volume model for two-dimensionnal shallow water flows

O. Souhar, J.B. Faure, André Paquier

Author-produced version of the article published in Environnemental Fluid Mechanics, 2007, 7, 303-315.
The original publication is available at http://www.springerlink.com/
doi : 10.1007/s10652-007-9028-5

Automatic sensitivity analysis of a finite volume model for two-dimensional shallow water flows

Souhar, O.[1], Faure, J-B.[2] and Paquier, A.[3]

June 21, 2007

Cemagref de Lyon
Hydrology-Hydraulic research unit
3 bis, quai Chauveau-CP 220,
69336 Lyon Cedex 09, FRANCE
Tel. +33 4 72 20 87 87 - Fax. +33 4 78 47 78 75

[1] Corresponding author, E-mail: drsouhar@hotmail.com
[2] faure@lyon.cemagref.fr
[3] paquier@lyon.cemagref.fr

Abstract

Given a numerical model for solving two-dimensional Shallow Water Equations, we are interested in the robustness of the simulation by identifying the rate of change of the water depths and discharges with respect to a change in the bottom friction coefficients. Such a sensitivity analysis can be carried out by computing the corresponding derivatives. Automatic Differentiation (AD) is an efficient numerical method, free of approximation errors, to evaluate derivatives of the objective function specified by the computer program, *Rubar20* for example. In this paper AD software tool *Tapenade* is used to compute forward derivatives. Numerical tests were done to show the robustness of the model and to demonstrate the efficiency of these AD-derivatives.

Key words : Automatic differentiation, Hydraulic model, Sensitivity analysis, Uncertainty propagation, Steady flow.

1 Introduction

The two-dimensional Shallow Water Equations are a relevant basis for modelling free surface flows, but their solution requires detailed information about the river configuration as the topographical data and the friction coefficient according to Manning law. It is also necessary to provide information bound to a particular event: these are the initial conditions of the flow as well as the upstream and downstream boundary conditions of the river.
The model input data are submitted to uncertainties difficult to evaluate and which may lead to meaningful uncertainties on the model output. The output results may be a map showing a flooded urbanized area; consequently, the induced damages may be very high. In these situations, the river can go far away from their mean water channel, submerging large areas, eventually with high velocity. In order to evaluate the uncertainties of the water level and river flow width according to the input data of a one-dimensional hydraulic model (Giraud *et al.* [11]), Dhervillez ([5]) and Faure *et al.* ([6]) tested several noising methods. Their study showed that if the bottom level is gaussian then consequently the water level is gaussian too in steady flow. On the other hand, they showed in the same paper that if the bottom level is gaussian then the river flow width is not gaussian.
In such a case, sensitivity analysis can be lead by computing partial derivatives of the output functions with respect to the corresponding parameters. The most familiar schemes to approximate derivatives of functions specified by computer programs are the central differences and the so-called forward finite differences. So for a smooth function $F : I\!R^n \longrightarrow I\!R^m$ and for some direction $d \in I\!R^n$, the central approximation, for $\bar{\varepsilon} << 1$, is defined as

$$\nabla F(x).d \approx \frac{F(x + \bar{\varepsilon}.d) - F(x - \bar{\varepsilon}.d)}{2.\bar{\varepsilon}}, \tag{1.1}$$

which have a truncation error of $\mathcal{O}((\bar{\varepsilon}.d)^2)$. Thus, using this central operator, the gradient requires at least $(2n)$ evaluations of the function F. However, the forward operator defined below requires only $(n + 1)$ evaluations of F, but the truncation error is only of $\mathcal{O}(\bar{\varepsilon}.d)$:

$$\nabla F(x).d \approx \frac{F(x + \bar{\varepsilon}.d) - F(x)}{\bar{\varepsilon}}. \tag{1.2}$$

From a theoretical point of view, a good accuracy of the approximate (1.1) or (1.2) is obtained for very small values of $\bar{\varepsilon}$. Nevertheless, $\bar{\varepsilon}$ may not be arbitrarily small, since for very small values, the difference between $F(x + \bar{\varepsilon}.d)$

and $F(x)$ is of the discretization error order, thus losing its significance.
Nowadays, methods and softwares for sensitivity analysis of several engineering problems (fluid dynamics, chemistry, meteorology or environment studies etc) have demonstrated that forward sensitivities can be computed reliably and efficiently via Automatic Differentiation (AD). AD is superior to divided differences because AD-generated derivative values are free of approximation errors, see Section 4 for more details. In this paper the *Tapenade* software tool, developed by Tropics team INRIA Sophia-Antipolis, is used to compute the desired derivatives of the objective functions specified by the finite volume two-dimensional shallow water model *Rubar20* with respect to the roughness coefficients. The output results or the objective functions are the water depth and the discharge computed by *Rubar20* (Paquier [21]).
The outline of this paper is as follows. The governing equations are first presented. Details of the numerical model *Rubar20* are discussed. An overview of the AD strategy is given. The numerical results are presented and the computed results are verified by comparing them with other results obtained from another hydraulic model. Finally, the sensitivity of the model according to the roughness coefficient is studied.

2 Shallow Water Equations

The two-dimensional shallow water equations describe the unsteady free surface flows, which simply express the conservation of mass and momentum. Usually these equations are described in terms of water depth and discharge under the following assumptions (Lai [16]): hydrostatic pressure distribution, very small bottom slopes, uniform velocity in the vertical direction and incompressible fluid. The resulting equations are

$$\frac{\partial h}{\partial t} + \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} = 0 \tag{2.1}$$

$$\begin{aligned} \frac{\partial q_x}{\partial t} + \frac{\partial (\frac{q_x^2}{h} + g\frac{h^2}{2})}{\partial x} + \frac{\partial (\frac{q_x q_y}{h})}{\partial y} &= -gh\frac{\partial Z}{\partial x} - gq_x\frac{\sqrt{q_x^2+q_y^2}}{C^2h^2} + \\ & D(\frac{\partial}{\partial x}(h\frac{\partial (\frac{q_x}{h})}{\partial x}) + \frac{\partial}{\partial y}(h\frac{\partial (\frac{q_x}{h})}{\partial y})) \end{aligned} \tag{2.2}$$

$$\begin{aligned} \frac{\partial q_y}{\partial t} + \frac{\partial (\frac{q_x q_y}{h})}{\partial x} + \frac{\partial (\frac{q_y^2}{h} + g\frac{h^2}{2})}{\partial y} &= -gh\frac{\partial Z}{\partial y} - gq_y\frac{\sqrt{q_x^2+q_y^2}}{C^2h^2} + \\ & D(\frac{\partial}{\partial x}(h\frac{\partial (\frac{q_y}{h})}{\partial x}) + \frac{\partial}{\partial y}(h\frac{\partial (\frac{q_y}{h})}{\partial y})) \end{aligned} \tag{2.3}$$

where t = time; h = water depth; Z = bed elevation; q_x and q_y = 'discharge' (water depth multiplied by velocity) along respectively the x-axis and the y-axis; C = Chezy friction coefficient which is considered as a constant or equalled to $STK \times h^{\frac{1}{6}}$ where STK is the Strickler's friction coefficient; g = acceleration due to gravity; D = diffusion coefficient which is estimated as a constant.

3 Numerical model

Based on a finite volume method, the model *Rubar20* is designed to solve the two-dimensional shallow water equations on unstructured triangular and rectangular grids. The model, written in Fortran 77 has been developed at Cemagref since 1988 (Paquier [21]). It uses a Van Leer explicit projection-correction scheme with second-order spatial and temporal accuracy in order to avoid spurious oscillations. The two main features of the numerical scheme are described here below. In case of hydraulic structures, the classical relations for weirs, gates, ... are used instead of the shallow water equations. In fact, an hydraulic structure can be defined as a set of a few cells where fluxes through one edge are computed either from the relations for the hydraulic structure, either from an addition of a flow computed from shallow water equations and a flow coming through an hydraulic structure. This last possibility may be useful, for instance, in the case of an embankment which is submerged although some discharge crosses it through a culvert. The model has been applied to simulate different situations of flows (Mignot [17], Mignot *et al.* [18], Mignot *et al.* [19], Paquier [22], [23], Paquier and Farissier [24], Paquier and Mignot [25], Paquier *et al.* [26], Proust *et al.* [27]). The scheme is stable for Courant number less than 1. The two main steps of the numerical scheme are described here below.

3.1 Riemann problems

In a finite volume approach, the updated values of the main variables (h,q_x,q_y) are calculated using the fluxes crossing the edges. To estimate these fluxes, the 2-D problem constituted by the first member of the equations (2.1) + (2.2) + (2.3) is simplified in a 1-D problem in which the left and right values (on each side of the edge) are constant. If x means the direction perpendicular

to the edge, the simplified 1-D problem writes:

$$\frac{\partial h}{\partial t} + \frac{\partial q_x}{\partial x} = 0 \tag{3.1}$$

$$\frac{\partial q_x}{\partial t} + \frac{\partial (\frac{q_x^2}{h} + g\frac{h^2}{2})}{\partial x} = 0 \tag{3.2}$$

$$\frac{\partial q_y}{\partial t} + \frac{\partial (\frac{q_x q_y}{h})}{\partial x} = 0 \tag{3.3}$$

The equations (3.1) + (3.2) + (3.3) constitute a nonlinear hyperbolic system for the 3-D variable $W = (h, q_x, q_y)$. Eigenvalues are $u - c$, u and $u + c$ where u is the velocity perpendicular to the edge and $c = \sqrt{gh}$ is the wave celerity. The second field is linearly degenerated whereas the first and the third ones are truly nonlinear. The Riemann invariants can be very simply computed as they respectively equal $u + 2c$ and v, $u + 2c$ and $u - 2c$, $u - 2c$ and v where v is the velocity parallel to the edge.
To solve this Riemann problem, it is possible to use a Roe type linearization (Roe [29]) which directly gives an estimate of the flux but alternative methods keeping the properties of the fields (and thus rarefactions and shocks) exist. Generally, these latter methods have the advantage to provide also an estimate of the values of the variables on the edge, which can be used when computing second member of the system (2.1) + (2.2) + (2.3) (Paquier [21]).
In order to obtain a second-order scheme in space, the values considered in the Riemann problem are computed by using slopes in the x and y directions (for each of the 3 variables h, q_x and q_y independently). For every cell, these slopes are computed by the method of the least squares (on each of the 3 variables independently) from the basis of the values at the centre of the cells which have one common edge with the given cell (Naaim [20]). The slope for any variable is later limited if the value of the given variable A (which may be $h + Z$, q_x or q_y) at the middle of one edge is not located between the value at the centre of the cell A_c and $\alpha A_n + (1 - \alpha)A_c$ where A_n is the value at the centre of the corresponding adjacent cell. The Van Leer correction factor α should be selected between 0.5 and 1.
In order to obtain second-order accuracy in time, the values of fluxes should be assessed at time $t_{n+1/2} = t_n + 0.5\Delta t$ instead of time t_n (Paquier [21]).

3.2 Treatment of second member

Using the fluxes across edges, it is possible to obtain provisional values of the variables at time $t_{n+1} = t_n + \Delta t$. Then we add the contribution of the second member which includes, at least, 2 terms:

i. Gravity or slope terms ($-gh\frac{\partial Z}{\partial x}$ or $-gh\frac{\partial Z}{\partial y}$).

ii. Bed friction by the use of a Chezy's coefficient (small water depths) or a Strickler's coefficient.

Gravity terms are treated as fluxes in such a way as an horizontal water surface remains strictly horizontal (Paquier [21]). For instance, the integral on one cell $\int\int -gh\frac{\partial Z}{\partial x}dxdy$ becomes

$$0.5 \sum_{edges} \alpha_i \frac{L_i}{S}(h_c + h_i)(Z_i - Z_c),$$

in which α_i is the x-co-ordinate of the normal to the edge i, L_i is the length of the edge, h_c and Z_c are the water depth and the bed elevation at the centre of the cell respectively, h_i and Z_i are the water depth and the bed elevation at the middle of the edge i respectively, S is the area of the cell.
Bed friction terms are more simply assessed at the centre of the cell. Their computation uses an implicitation in time in order to avoid numerical instabilities when a rapid change in water depth or velocity occurs.

4 Automatic differentiation

4.1 Generality

Derivatives play a fundamental role in various areas of computational science including optimization, data assimilation, assessment of uncertainties, sensitivity analysis, etc (Buecker *et al.* [1], Castaings *et al.* [3], Corliss *et al.* [4], Gilbert and Lemaréchal [10]). A classical way to obtain these derivatives (from computer models) is the *divided differences method*. The rounding errors and cancelation are important drawbacks of this approximation method; furthermore, these errors and complexity increase when higher derivatives are calculated. To overcome these problems we use AD tool (Souhar [30], Souhar and Faure [31], [32]).

The AD, just like the divided differences, requires only the original program. In mathematics, it's a method to numerically compute the partial derivatives of a function F specified by a computer program (Buecker *et al.* [1], Carle and Fagan [2], Giering [8], Griewank [13], Hascoet and Pascual [14]). An AD tool (Carle and Fagan [2] or Hascoet and Pascual [14] for example) builds a new source program that computes the derivatives of F. The tool behaves like a compiler, which first understands the meaning of the original program, represents it in an internal form, performs analyses to get deepen understanding, and generates an object code. AD exploits the fact that any computer program that implements a vector function $Y = F(X)$ can be decomposed into a sequence of elementary assignments, any one of which may be trivially differentiated by a simple table lookup. These elementary partial derivatives are combined to form some derivative information of F. This process leads to exact numerical derivatives, because derivatives are computed *analytically.*
In AD there are two methods to compute derivatives, the so-called *tangent linear model* that computes directional derivatives and is easy to implement on computer. The second method is the *reverse model* that computes the gradient of a function at one execution of the *adjoint program.* This latter program is relatively expensive and should therefore be optimized whenever possible.
Let $\mathcal{M}$ be a general nonlinear model, a mapping from $\mathbb{R}^n$ of the input variables $X = (x_1, ..., x_n)^T$ (initial conditions, topographic data, roughness coefficient, boundary conditions ... in hydraulic model cases) to $\mathbb{R}^m$ of the output variables $Y = (y_1, ..., y_m)^T$ (model state, objective function ...):

$$\begin{array}{rccc} F & : & \mathbb{R}^n & \longrightarrow & \mathbb{R}^m \\ & & X & \longmapsto & Y = F(X) \end{array}$$

The vectors $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ may be represented with respect to the ordinary unit basis vectors $\{\vec{e}_i; i = 1, ..., n\}$ and $\{\vec{f}_k; k = 1, ..., m\}$ as

$$X = \sum_{i=1}^{n} x_i \vec{e}_i, \quad Y = \sum_{k=1}^{m} y_k \vec{f}_k.$$

The entries x_i and y_k forming the vectors X and Y are called input *independent* and output *dependent* variables respectively. And all the other variables used in the program to evaluate $F(X)$ are called *intermediate.*
Consider a perturbation dX of the input variables. Its effect on the outputs may be obtained via the linear approximation of the model $\mathcal{M}$ in terms of

its Jacobian matrix, evaluated in the point X according to

$$dY = \nabla F(X).dX \tag{4.1}$$

Equation (4.1) is called the *tangent linear model* (Hascoet and Pascual [14], Inria [15]).

$$\delta y_k = \sum_{i=1}^{n} \frac{\partial F_k(X)}{\partial x_i} \delta x_i, \tag{4.2}$$

where, $dX = \sum_{i=1}^{n} \delta x_i \vec{e}_i$ and $dY = \sum_{k=1}^{m} \delta y_k \vec{f}_k$.

The cost (algorithmic complexity) for computing gradients of scalar-valued functions with either *divided differences method* or *tangent linear model* of AD grows linearly with the number of variables, whereas the so-called *reverse model* of AD, an extremenlly efficient way, can compute such gradients at constant cost (Griewank [12]). Thus, if there are many input variables ($10^6 - 10^8$, for large-scale shallow water applications), it becomes prohibitive to proceed directly as in (4.2), and for these large applications the *reverse model* is used, see (Castaings *et al.* [3], Gilbert [9], Griewank [12], Hascoet and Pascual [14], Inria [15], Rall [28]) for more details.
In this work we use *Tapenade*, an AD software tool, to compute automatic derivatives of a two-dimensional hydraulic model. This tool is developed at INRIA Sophia-Antipolis by Tropics team (Hascoet and Pascual [14], Inria [15]).

4.2 Building and validation of AD models

Derivation of models is not so easy, indeed a lot of difficulties can occur if *direct models* (the source models which must be differentiated) have not been written in the goal to be differentiated.
The first step in using *Tapenade*, in particular to derive *Rubar20* model, is to load the input source program and to identify which routine implements the mathematical function that we want to differentiate. We should then share the model in two parts corresponding respectively to the reading of data and initialization of the model and to the calculation a long the time. Therefore only the part of the model that achieves the computations is going to be differentiated.
A pre-treatment step for AD strategy must be done. *Tapenade* may issue a

number of error and warning messages when differentiating a file is done. For example, in the source of the direct model there exists some subroutines with subroutine as arguments. *Tapenade* doesn't accept this type of instructions and hence obliges the user to modify manually the source model.
A post-treatment step is also required, it consists to insert read data and write results instructions in the differentiated subroutines and the necessary calls of subroutines like those of initializations. After this, a part of debugging compilation errors must be done (Souhar [30]).
Tapenade can be utilzed as a server, which runs at INRIA Sophia-Antipolis at *http://tapenade.inria.fr.8080/tapenade/indes.jsp*. It can also be installed locally, in that case it is run by a simple command line, which can be included into a *Makefile*. Figure 1 shows a typical *Makefile* built to make easy the derivation of the code after each modification.

```
FILES =  module.f calcvl.f cflmax.f critd.f critg.f dhchoc.f \
         dicho2.f flvla1.f flvla2.f flxga.f grdlp.f intapp.f \
         nwt2.f nwt.f resrmn.f rmn2.f rmn3.f rmnd.f rmng.f \
         schem1.f secmb1.f secmb2.f  secmb3.f secmb5.f secmb6.f \
         topo.f lefr.f cedce.f lecl.f caldeb.f c.f gamad.f dgamad.f\
         gamag.f dgamag.f sgn.f

b:
     ~/usr/bin/tapenade -b -head module \
          -inputlanguage fortran77 \
          -outputlanguage fortran77 \
          -outvars "HE QUE QVE" \
          -vars "FREE" \
          -r8 \
          -html \
          -O backward $(FILES)
d:
     ~/usr/bin/tapenade -d -head module \
          -inputlanguage fortran77 \
          -outputlanguage fortran77 \
          -outvars "HE QUE QVE" \
          -vars "FREE" \
          -r8 \
          -html \
          -O forward  $(FILES)
```

Figure 1: *A typical Makefile of "Rubar20"*

So the options are defined as follows.

- *head* allows to specify the top differentiation routine that implements the mathematical function.

- *inputlanguage* allows to select the source code language.
- *outputlanguage* allows to select the output differentiated code language.
- *outvars* allows to define the output variables and whose derivatives are required (here the outputs are the water depth and discharge).
- *vars* allows to choose the input variables with respect to which differentiated must be made (here the inputs are the friction coefficients).
- *O* point out the directory of the differentiated code.

The command *make d* (resp. *make b*) permit then to obtain the *tangent linear model* (resp. *the reverse model*) of the source program defined by its arborescence given in *FILES* option of Fig. 1.
Tapenade will probably not produce a perfect differentiated program at first try. After AD, a validation step is necessary to check the computed derivatives. It should be noted that a common technique to check validation of these derivatives is to compare them to finite differences approximations, for example:

$$\frac{\partial F(x)}{\partial x_i} = lim_{\varepsilon \to 0} \frac{1}{\varepsilon}[F(x_1, ..., x_i + \varepsilon, ..., x_n) - F(x_1, ..., x_i, ..., x_n)]$$

for $i = 1, ..., n$. In (Hascoet and Pascual [14], Inria [15]), the developers of *Tapenade* proposed this technique to validate *tangent linear models* and it is called "gradient test". The so-called "dot product test" is proposed in the same user's guide to validate *reverse models*.
Figure 2 below shows the validation of derivatives of *Rubar20* obtained by *Tapenade*. This figure shows well that the error function $r(\varepsilon) \longrightarrow 0$ when $\varepsilon \longrightarrow 0$, where r is defined as follows:

$$r(\varepsilon) = \| \frac{F(x + \varepsilon.d) - F(x)}{\varepsilon} - \nabla F(x).d \|_2, \tag{4.3}$$

where $d = (1, ..., 1)^T$ is the unit vector of $\mathbb{R}^n$ and $\nabla F(x).d$ is computed by *Tapenade*.

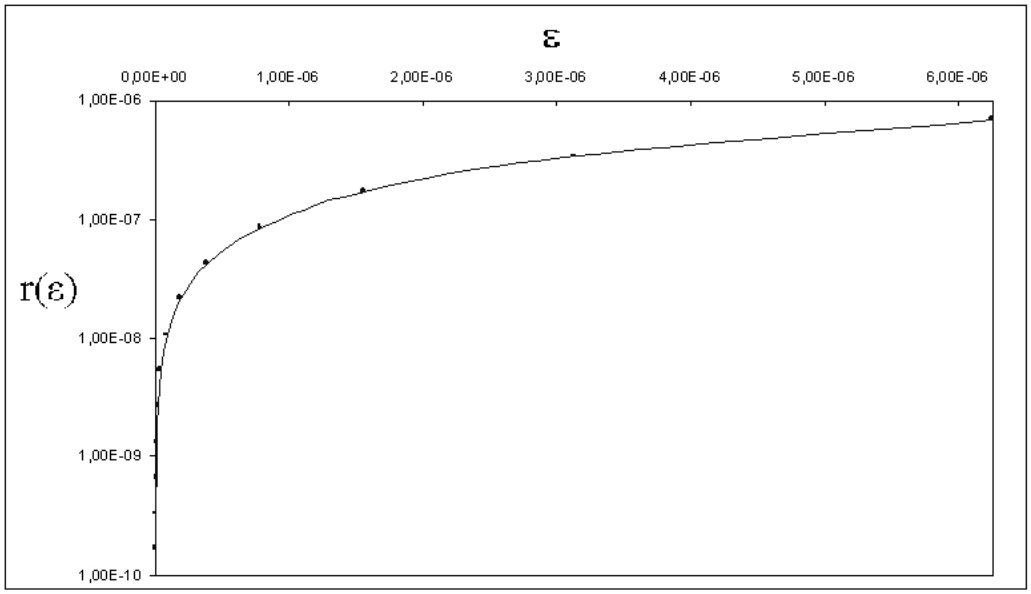

Figure 2: *Gradient test for a 2-D model: "Rubar20"*

5 Numerical experiment

In this Section, first we propose one example to show the robustness and validity of our numerical model *Rubar20*; secondly we demonstrate the efficiency of the forward sensitivity method, which uses AD-generated derivatives computed by *Tapenade*.
In all of our tests, the precision for water depths is $10^{-4}\,m$, the correction Van Leer's coefficient is 0.6, the Courant number is 0.5 and the duration of the simulation is $80\,s$.

5.1 Validity of *Rubar20*

We are interested here in a complex problem. The flow, a portion of a river of length $28.5\,m$ and width $20\,m$, is perturbed by some obstacles. The portion is crossed by a bridge with two triangles representing the bridge abutments and two cylindric bridge piers schematized by two octagons (see Fig. 3).
The cross sections of the main channel are trapezoidal of slope 2/1, a discharge of $62\,m^3/s$ constitutes the upstream boundary condition, the downstream condition is imposed such that $Z + h = 0$. The friction coefficient

according to Manning law is homogeneous and uses a Strickler's coefficient equal to 40.

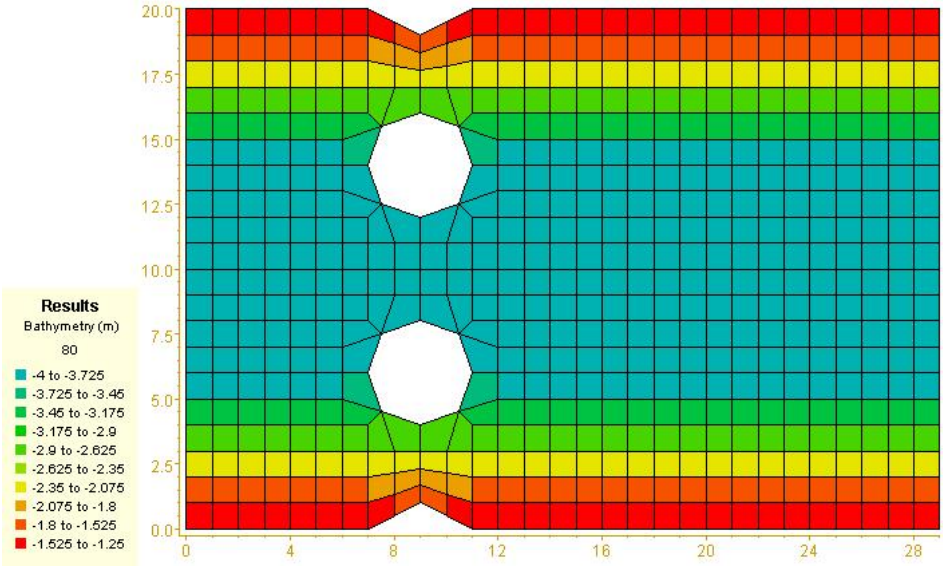

Figure 3: *Grid for computation of the study*

Fig. 4 and Fig. 5 provide water depths (m) and discharge fields (m^2/s) computed by *Rubar20* for two diffusion coefficient values. According to these figures we remark that the model conserve the symmetry of the results because of the symmetry of the posed problem. We study now the recirculation behind the bridge piers and the bridge abutments for two different diffusion coefficients D. Fig. 4 and Fig. 5 give the velocity fields which permits to assess the recirculation's length. For $D = 5.10^{-2}$ (Fig. 4), the results are near to those computed by *Télémac* (*Télémac* is a numerical model developed by Electricité de France) (Galland *et al.* [7]). For $D = 5.10^{-3}$, *Télémac* finds eddies behind the bridge piers. *Rubar20* gives recirculations behind the bridge abutments (Fig. 5). This difference results mainly from a small difference of grids used for computations in the two codes.

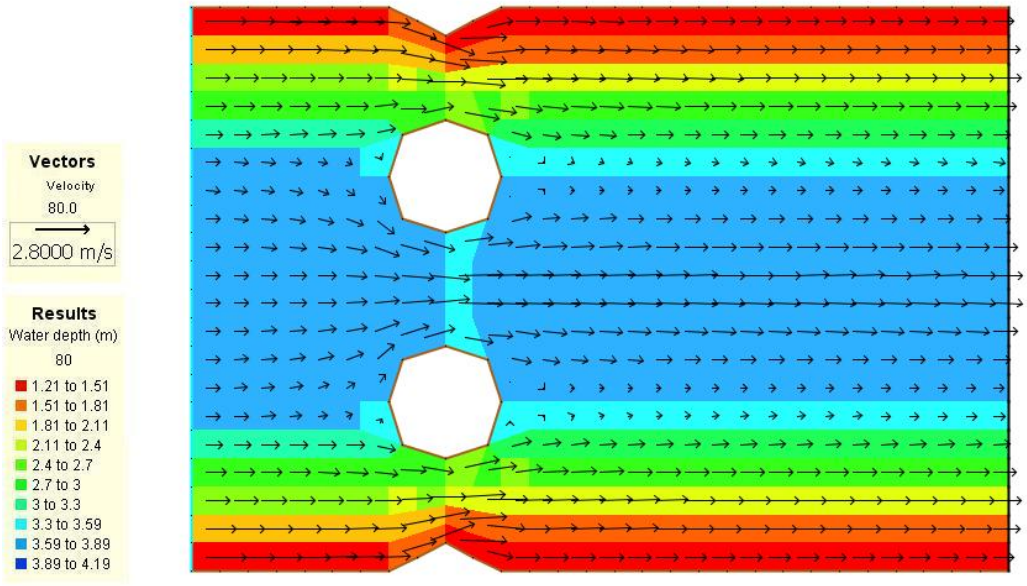

Figure 4: *Velocity field in steady flow using Rubar20:* $D = 5.10^{-2}$

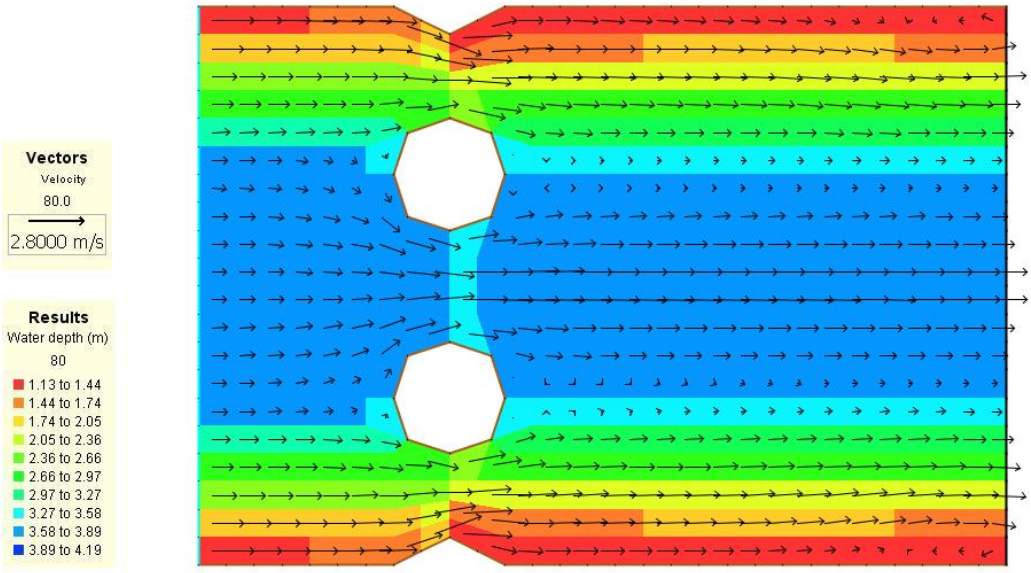

Figure 5: *Velocity field in steady flow using Rubar20:* $D = 5.10^{-3}$

5.2 Sensitivity analysis

Assessment and sensitivity analysis play an important role in decision making on flood control, so minimization of the risk should be the general objective. In the following we are interested in the sensitivities of water depth and discharge with respect to the roughness coefficient. We compute via AD the corresponding derivatives.
The study has a homogeneous roughness, then it is important to define the subspace $\mathbb{K} = span\{\varphi\}$ of $\mathbb{R}^n$, where $n = 548$ is the number of computational points and the basis vector is $\varphi = (1, ..., 1)^T \in \mathbb{R}^n$, in this case the function F of Subsection 4.1 becomes:

$$\begin{array}{rccl} F: & \mathbb{K} & \longrightarrow & \mathbb{R}^m \\ & (STK) & \longmapsto & F(STK) = (F_1(STK), ..., F_m(STK))^T, \end{array} \tag{5.1}$$

where $F_j(STK)$ represents the water depth or the discharge at section j and $m = 548$, and STK is the friction coefficient.
According to Taylor's formula, which illustrates the physical definition of the gradient, the values of the function F at the points STK and $STK + \delta\varphi$ are related by:

$$F(STK + \delta\varphi) = F(STK) + \delta\nabla F(STK).\varphi + \mathcal{O}(\delta^2\varphi), \tag{5.2}$$

with

$$\nabla F(STK) = \begin{pmatrix} \frac{dF_1(STK)}{dx} \\ . \\ . \\ . \\ \frac{dF_m(STK)}{dx} \end{pmatrix} \tag{5.3}$$

Fig. 6 and Fig. 7 show gradients of water depths (m) and discharges (m^2/s) related to the friction coefficient (defined by (5.3)) computed using the AD tool *Tapenade*. So the sensitivities of the water depths and discharges are also symmetric for both diffusion coefficient values. We remark that these sensitivities vary in the opposite of the diffusion coefficient; i.e., sensitivities are less high for $D = 5.10^{-2}$ than for $D = 5.10^{-3}$. We remark also that discharge's sensitivities are high behind the bridge piers and the bridge abutments for both diffusion coefficient values.

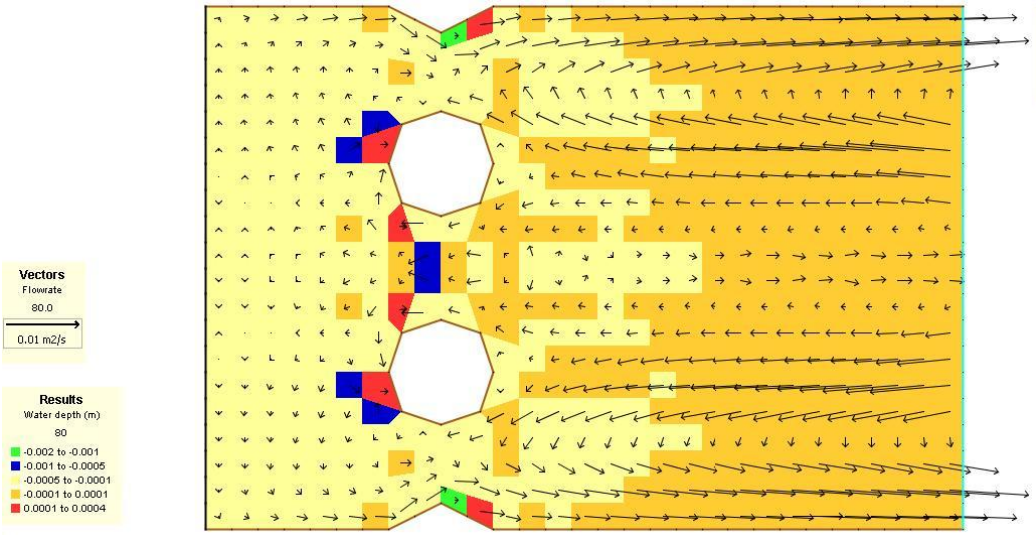

Figure 6: *Sensitivities of water depths and discharges to bottom friction:* $D = 5.10^{-2}$

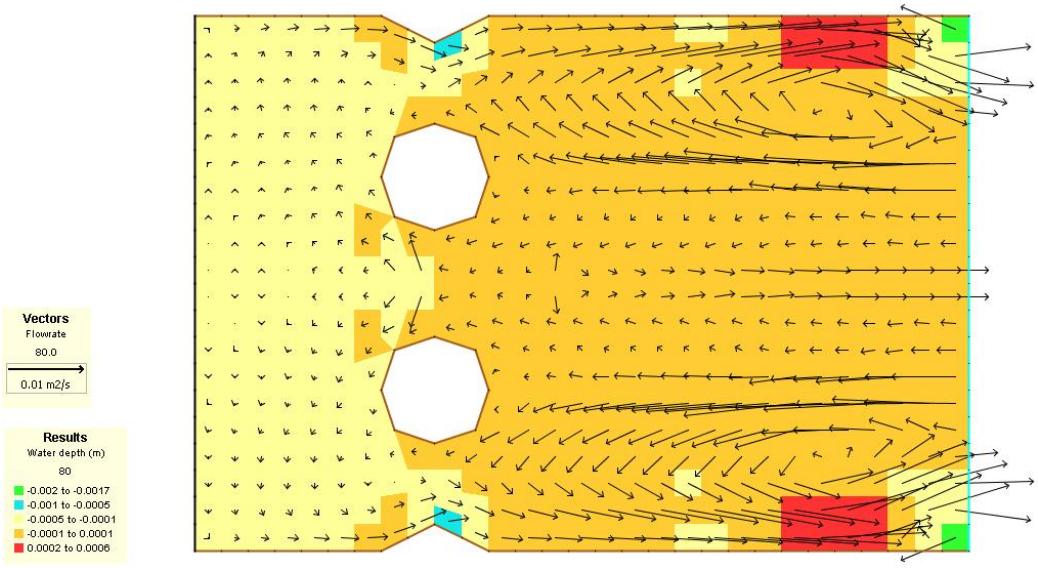

Figure 7: *Sensitivities of water depths and discharges to bottom friction:* $D = 5.10^{-3}$

According to Fig. 4 and Fig. 6 ($D = 5.10^{-2}$), an increase of the friction coefficient may lead to find recirculations behind the bridge piers, whereas a decrease of the friction coefficient may lead to find recirculations behind the bridge abutments. From Fig. 5 and Fig. 7 ($D = 5.10^{-3}$), we remark that an increase of the friction coefficient may lead to find more important recirculations behind the bridge piers and to increase recirculations zones behind the bridge abutments.

Conclusion and remarks

The two-dimensional hydraulic model *Rubar20* provides satisfactory results faced to a complex problem. The model can provide detailed velocity field and water levels. But a computation with fine grids should be used to detail and understand the recirculation problem more accurately. The computational technique can be easily extended to channels with arbitrary cross section, for computing the propagation of a flood surge on a dry bed, also for the computation of the flood wave resulting from the break of a dam, etc.
We have shown how AD tool can aid in the sensitivity analysis of a two dimensional finite volume model of shallow water flows. So the numerical sensitivities of water depths and discharges related to the friction coefficient are also symmetric as the results provided by the model and because of the symmetry of the channel. Furthermore, these sensitivities become more important for a small diffusion coefficient. Such sensitivities employed the forward derivatives computed using the AD software *Tapenade*. But it is still necessary to manually modify the source codes before automatic derivative step because AD tools still remain unable to take into account all programming instructions.

References

[1] Buecker, H-M., Corliss, G., Hovland, P., Naumann, U., Noriss, B. 2005 *Automatic Differentiation: Applications, theory and tools*, Springer, New York (NY), Lecture Notes in Computational Science and Engineering, Chapter 34, 293 - 298.

[2] Carle, A. and Fagan, M. 2000 *ADIFOR 3.0 Overview* , Rice University Report CAAM-TR-00-02.

[3] Castaings, W., Dartus, D., Honnorat, M., Le Dimet, F.-X., Loukili, Y. and Monnier, J. 2004 *Automatic differentiation: a tool for variational data assimilation and adjoint sensitivity analysis for flood modelling*, AD 2004 - Fourth International Workshop on Automatic Differentiation, July 19–23, 2004, Argonne National Laboratory, Gleatcher Center,Chicago, IL, USA.

[4] Corliss, G., Faure, C.,Griewank, A., Hascoet, L., Naumann, U. 2001 *Automatic Differentiation of Algorithms From Simulation to optimization*, Springer, Computer and Information Science.

[5] Dhervillez, L. 2001 *Incertitudes sur l'extension d'une zone inondable*, DESS Thesis, Joseph Fourier University, Grenoble, France, 110 pages.

[6] Faure, J-B., Dhervillez, L. and Vidal, J-P. 2002 *Automatic evaluation of the uncertainty on a flooded area*, Hydroinformatics'2002, Cardiff, UK.

[7] Galland, J.C., Hervouet, J.M., Rouge, D. and Thellier, P. 1991 *Code TELEMAC, Version 1.0, Note de validation*, Electricité de France/Direction des études et Recherches HE-42/90.06.

[8] Giering, R. 1997 *Tangent linear and Adjoint model compiler*, Users manual, http://www.autodiff.com/tamc , FastOpt Gmbh.

[9] Gilbert, J. C. 1992 *Automatic differentiation and iterative processes*, Optimization methods and software, 1, 13-21.

[10] Gilbert, J. C., and Lemaréchal, C. 1989 *Some numerical experiments with variable storage quasi-Newton algorithms*, Mathematical Programming 45: 407 - 435.

[11] Giraud, F. M., Faure, J-B., Zimmer, D., Lefeuvre, J.C. and Skaggs, R.W. 1997 *Hydrologic modeling of a complex wetland*, Journal of Irrigation and Drainage Engineering, 123: 5, 344-353.

[12] Griewank, A. 1993 *Some bounds on the complexity of gradients, Jacobians, and Hessians*, In Panos M. Pardalos, editor, Complexity in Nonlinear Optimization, 128-161, World Scientific Publishers.

[13] Griewank, A. 2000 *Evaluating Derivatives: Principales and Techniques of Algorithmic Differentiation*, Number 19 in Frontiers in Applied Mathematics, SIAM, Philadelphia.

[14] Hascoet, L. and Pascual, V. 2004 *Tapenade 2.1 user's guide*, Rapport INRIA n 300, Septembre 2004, 78 pp. http://www.inria.fr/rrrt/rt-0300.html

[15] Inria, *Projet TROPICS*, TAPENADE tutorial. http://www-sop.inria.fr/tropics/

[16] Lai, C. 1986 *Numerical modelling of unsteady open-channel flow*, Adv.in Hydroscience,14,162-323.

[17] Mignot, E. 2005 *Etude expérimentale et numérique de l'inondation d'une zone urbanisée : cas des écoulements dans les carrefours en croix*, Thèse en Mécanique, Ecole Centrale de Lyon.

[18] Mignot, E., Paquier, A. and Haider, S. 2006 *Modeling floods in a dense urban area using 2D shallow water equations*, Journal of Hydrology, vol. 327, n 1-2, 186-199.

[19] Mignot, E., Paquier, A. and Ishigaki, T. 2006 *Comparison of numerical and experimental simulations of a flood in a dense urban area*, Water science and technology, vol. 54, n 6-7, p. 65-73.

[20] Naaim, M. 1991 *Numerical modelling of hydrodynamic effects caused by a solid sliding in a reservoir*, Thèse de l'université Joseph Fourier Grenoble 1.

[21] Paquier, A. 1995 *Modélisation et simulation de la propagation de l'onde de rupture de barrage*, Thèse de l'université de Saint-Etienne.

[22] Paquier, A. 1998 *1-D and 2-D models for simulating dam-break waves and natural floods*, Union Européenne CADAM meeting, Wallingford, GBR, Mars 1998.

[23] Paquier, A. 2003 *Surface flows during high floods in towns*, Houille blanche, vol. 6, p. 89-93.

[24] Paquier, A. and Farissier, P. 1996 *Use of a 2-D model fo simulating the flooding of a plain*, Hydroinformatics'96, Muller, 129-136.

[25] Paquier, A. and Mignot, E. 2003 *Use of 2D models to calculate flood water levels: calibration and sensitivity analysis*, XXX IAHR congress, Thessaloniki, GRC, 24-29 August 2003.

[26] Paquier, A., Tanguy, J.M., Haider, S. and Zhang, B. 2003 *Estimation of the flood levels for a flash flood in urban area: comparison of two hydrodynamic models on the 1988's Nîmes flood*, Revue des sciences de l'eau, vol. 16, n 1, p. 79-102.

[27] Proust, S., Rivière, N., Bousmar, D., Paquier, A., Zech, Y. and Morel, R. 2006 *Flow in compound channel with abrupt floodplain contraction*, Journal of hydraulic engineering, vol. 132, n 9, p. 958-970.

[28] Rall, L. B. 1981 *Automatic Differentiation Techniques and Applications*, Lecture Notes in Computer Science 120 Springer-Verlag, Berlin.

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

[30] Souhar, O. 2006 *Evaluation of uncertainties for 1-D fluvial model. Automatic differentiation of 2-D Shallow water flows by Tapenade*, Tech. Rep., Cemagref de Lyon, CHHLY.20108 (LY), 31 pp.

[31] Souhar, O. and Faure, J-B. 2006 *Two gradient techniques to study sensitivities applied to 1-D and 2-D hydraulic models*, Submitted.

[32] Souhar, O. and Faure, J-B. 2007 *Uncertainty propagation and sensitivity derivatives of a hydraulic model via automatic differentiation*, Submitted.