Telemac version 6.0, release notes. Telemac-2D and Telemac-3D

Jean-Michel Hervouet

18th February 2010

Contents

Chapter 1

Telemac-2D and 3D: positive depths on dry zones

1.1 Principle

The problem of negative depths in Telemac-2D and 3D has always been the price to pay to have fast and implicit schemes, whereas explicit techniques such as the finite volume option with kinetic schemes were able to ensure a positive depth, but at a considerably higher computer time, due to much smaller time steps. To cope with negative depths, a specific smoothing algorithm had been designed, with sometimes a well-known drawback effect: water slowly creeping above dykes when they were discretized with too few points. We now present another solution which consists of limiting the fluxes between points. It is actually a post-treatment which ensures both mass-conservation and positivity of depth. The continuity equation in the sense of finite volumes (e.g. as transmitted to Delwaq) is still ensured, the continuity equation in the sense of finite elements is spoiled because the original velocities are not changed accordingly to the new depths.

The main idea is summed up here and consists of 3 steps:

- The fluxes between points are computed. We use here the ideas of LeoPostma, already implemented in the interface to Delwaq.
- Starting from depths at time n, water corresponding to the fluxes are transferred between points, provided that the depth remains positive, otherwise the fluxes are locally limited (fluxes which are not used are kept for a further iteration). This is done in a loop over triangle edges, which can be repeated until there is no more possible water to transfer.
- The remaining fluxes are left over.

We shall now get into the details of the technique. We start from the 2D continuity equation:

$$\frac{h^{n+1}-h^n}{\Delta t} + div(h^{prop}\left[\theta_u \overrightarrow{u}^{n+1} + (1-\theta_u)\overrightarrow{u}^n\right]) = Sce$$

where Sce stands for the discharge sources at points (culverts and so on). This equation is discretized in the following matrix form:

$$\frac{M}{\Delta t}\,(H^{n+1}-H^n)+BM1\;U+BM2\;V=RHS \tag{1.1}$$

where M is the mass-matrix and Δt the time-step. RHS is a right-hand side accounting for the boundary fluxes (stemming from an integration by part of the divergence term) and the source terms stemming from Sce. If we use mass-lumping (it will be mandatory here), this will give for every degree of freedom i:

$$\frac{S_i}{\Delta t}(h_i^{n+1}-h_i^n)+(BM1\;U+BM2\;V-RHS)_i=0 \tag{1.2}$$

where $S_i=\displaystyle\int_\Omega \Psi_i\;d\Omega$ is the area of the finite volume around point i, and also what we call the volume of basis i. Options like the wave equation and the specific treatment of the free surface gradient (keyword "free surface gradient compatibility") also fit within this framework, at the cost of changing U and V into modified velocities which may be partially treated as piece-wise constant. These modified velocities are denoted $UDEL$ and $VDEL$ in Telemac (because they are used by the interface to Delwaq).

The quantity:

$$(BM1\;U+BM2\;V-RHS)_i$$

can be interpreted as the flux leaving point i. It includes source terms and fluxes at the boundaries, which are in RHS. The terms:

$$BM1\;U+BM2\;V$$

at element level are:

$$\Phi_i^{el}=-\int_\Omega h\overrightarrow{u}.\overrightarrow{grad}(\Psi_i)\;d\Omega \tag{1.3}$$

and we have explained in references [2] and [3] how to transform them into fluxes between points, so that the complete continuity equation becomes:

$$\frac{S_i}{\Delta t}(h_i^{n+1}-h_i^n)+\sum_j \Phi_{ij}+b_i=Sce_i \tag{1.4}$$

where b_i are the fluxes at the boundaries (denoted $FLBOR$ in Fortran sources). Note also that term Sce_i/S_i is SMH in Telemac-2D Fortran sources. We have now:

$$h_i^{n+1}=h_i^n-\frac{\Delta t}{S_i}\left(-Sce_i+\sum_j \Phi_{ij}+b_i\right) \tag{1.5}$$

We have $\Phi_{ij}+\Phi_{ji}=0$ and in what follows, as we shall look at fluxes segment after segment, only the positive Φ_{ij} will be considered, otherwise fluxes would be treated twice.

1.2 Limiting internal fluxes

Let us first deal with fluxes between points, regardless of other boundary and source terms. Starting from h^n we want to construct a new depth at time $n+1$, and the depth "in construction" is denoted here $\widetilde{h}$. In a loop over all segments, we get every time a specific i and j (apices of the segment), and we would like to apply the formula:

$$\widetilde{h}_i \text{ replaced by } \widetilde{h}_i - \frac{\Delta t}{S_i}\Phi_{ij} \tag{1.6}$$

$$\widetilde{h}_j \text{ replaced by } \widetilde{h}_j + \frac{\Delta t}{S_j}\Phi_{ij} \tag{1.7}$$

but there is a risk of negative $\widetilde{h}_i$. If there is a risk, i.e. if $\Phi_{ij} > \frac{S_i \widetilde{h}_i}{\Delta t}$, then the flux is limited by a factor:

$$\theta = \frac{S_i \widetilde{h}_i}{\Phi_{ij}\Delta t}$$

We then do:

$$\widetilde{h}_i \text{ replaced by } \widetilde{h}_i - \theta\frac{\Delta t}{S_i}\Phi_{ij} \tag{1.8}$$

$$\widetilde{h}_j \text{ replaced by } \widetilde{h}_j + \theta\frac{\Delta t}{S_j}\Phi_{ij} \tag{1.9}$$

which ensures the conservation of water, and:

$$\Phi_{ij} \text{ replaced by } (1-\theta)\Phi_{ij}$$

which stores in Φ_{ij} the flux that has not yet been taken into account (it is likely to be used in the next loop over all segments).

Then this loop over all segments is repeated with the remaining Φ_{ij}. This is the key point!

After a number of iterations, the situation remains unchanged, i.e. a criterion like $\sum abs(\Phi_{ij})$ is no longer decreasing. The remaining Φ_{ij} are then left over as non physical because they would lead to negative depths. The parts of fluxes which have been duly transferred may be transmitted to the Delwaq so that they are taken into account to form a perfect continuity equation with depths and fluxes in accordance.

1.3 Boundary and source terms

Boundary and source terms are not likely to be interpreted in terms of fluxes between points. Moreover a sink term may lead to negative depths, which brings in fact a specific CFL number for the time step. We have applied the following algorithm:

- Step 1: taking into account the source and boundary terms bringing water (the depths are increased).
- Step 2: applying the limitation of internal fluxes described above, this leads to positive depths.

- Step 3: taking into account the source and boundary terms removing water (the depths are decreased).

Step 3 may raise problems, thus a limiting factor of the source terms or boundary terms may be applied also at this level.

1.4 Extension to 3D

Horizontal fluxes

The extension to 3D raised a priori little difficulty. When the wave equation option is used (i.e. the only option left in version 6.0) the shallow water continuity equation is solved first and we can work on it as it has been described above. The only difference is that the compatible 2D depth averaged velocity field is not known, as we work on 3D velocity fields and then do the integration on depth at the discrete level. The fluxes are thus first computed in 3D, then assembled on the vertical. Instead of computing fluxes with equation 1.3, we compute in 3D:

$$\Phi_i^{el} = -\int_\Omega \overrightarrow{u}.\overrightarrow{grad}(\Psi_i)\ d\Omega$$

without assembling the element by element fluxes, and add them on the vertical, to get non assembled fluxes on triangles. For every layer of 3D elements, the contributions of the 6 points of a prism are added on the 3 points of a triangle in the following way:

- 3D points 1 and 4 on 2D point 1
- 3D points 2 and 5 on 2D point 2
- 3D points 3 and 6 on 2D point 3

  then we can compute the fluxes between 2D points as already explained.

Vertical velocities

- A posteriori, a new difficulty appeared. In 3D, the horizontal fluxes (computed in the transformed mesh) are used to compute the vertical velocities in the transformed mesh. The horizontal fluxes must then be strictly compatible to the depth. As we limit only the point to point fluxes and not the original advecting field, there is an error here that could translate into wrong vertical velocities. The solution consisted of replacing the element by element horizontal fluxes by edge by edge horizontal fluxes to compute the internal fluxes (array called FLUINT in Telemac-3D). The edge by edge fluxes can be limited, provided that the 2D limiting coefficient has been kept. This 2D limiting coefficient (called FLULIM) is given per 2D segment and is applied for all planes on the vertical. This is done in subroutine FLU3DLIM.

1.5 Domain decomposition in parallelism

The algorithm raises an important problem in parallelism with domain decomposition. During the loop on all segments which changes the depths under construction, these changes, if done on an interface point, should be immediately transmitted to the relevant neighbouring sub-domains. This has been considered a too heavy way to proceed. So far we resorted to the following procedure: the depths of interface points are merely shared between processors (structure MESH%FAC%R, the inverse of the number of sub-domains a point belongs to, used for parallel dot products, is available for this). This means that if a point belongs to 2 sub-domains, the loop on segments will start locally with half the real depth. All sub-domain will ensure the positivity of their part of depth. The modified depths will then be summed after the loop (this also ensures the digit-to-digit equality of depths). Tests show that in this way we have also a convergence of the algorithm, but slower. It is important here that the fluxes used are not assembled (other sub-domains are ignored when they are built). The parallel assembly is in fact done when the depths are summed on interface points.

It could be that the algorithm is hindered by the fact that fluxes are not assembled. For example, on either side of an interface segment fluxes could be opposite and sum to 0, which is easier to avoid negative depths. In fact any combination of fluxes that have the same sum when assembled should work, but maybe with a different efficiency. We have eventually chosen to take the average on either side. This seems consistant with the fact that we share the depth, and it will become mandatory with tracers, because the sign of the flux will give the way to do the upwinding. A specific subroutine "mult_interface_segments" has been designed for this. Tests show that there is no overcost, which, given that it costs an extra parallel communication, is a hint that it speeds up the rest of the process.

1.6 Tests and applications

1.6.1 The Malpasset dam break

With Telemac-2D:

Figure 7.1 compares the previous smoothing algorithm and the one presented here, in scalar mode and in parallel mode with 16 processors. The computation is done on a Unix HP C3700 with a frequency of 750 MHz and on a Dell Linux machine with 2 processors. For the sake of simplicity the 16 processors have only been simulated as 16 different runs on a single HP workstation.

	smoothing	**flux correction**	**kinetic scheme**
HP C3700 750 MHz	133 s	168 s	$>$ 6 hours
Dell (Calibre 5) 1 processor	60 s	68 s	2 h 32' 18"
Dell (Calibre 5) 2 processors	44 s	50 s	no parallelism

Table 1.1: computer times with the Malpasset dam break test-case

Table 1.1 gives the computer times of different options and machines with 1 or 2

processors. Kinetic schemes have also been tested, as an alternative to get positive depths, but the times are much higher, due to the fact that it is an explicit scheme with a CFL limitation which leads to time steps sometimes 100 times smaller than the 4 s used on finite element side. On Figure 7.1 a threshold of 1 cm has been chosen for displaying depths. This allows a fair comparison of inundation extents. What is then not seen in the figure at the top (smoothing) is the fact that depths of -0.11 m are observed in the results. The -1 mm line extends far from the normally flooded area, due to the number of smoothings applied. Another important point is that due to truncation errors, the smoothing algorithm starts working also for the water at rest in the sea. At the end of the computation, a boundary point offshore which should have a depth of 20 m has only 19.9994 m. This is negligible but may lead to noticeable differences in long term computations. The new algorithm suppresses all these drawbacks. The mass conservation is excellent in both cases. The water volume lost with smoothing is -1858.743 m^3, and it is 0.298 10^{-7} m^3 with flux correction. If we use an exact solver for linear systems, the error becomes respectively 0.596 10^{-7} m^3 (computer time 217 s) and -0.11 10^{-6} m^3 (computer time 276 s).

With Telemac-3D

The behaviour is quite comparable to what is obtained in 2D, the computer times obtained with the 2-plane case are given in Table 1.2.

	smoothing	**flux correction**
HP C3700 750 MHz	769 s	768 s
Dell (Calibre 5) 1 processor	341 s	361 s
Dell (Calibre 5) 2 processors	238 s	252 s

Table 1.2: computer times with the Malpasset dam break test-case

The difference of computer time is not significant.

Again with the smoothing option, the total mass lost at the end of the computation is -143.8726 m^3, whereas with positive depths it is -0.59 10^{-7} m^3, the solver relative accuracy being 10^{-6}. If we use an exact solver for linear systems, it becomes respectively -0.29 10^{-7} m^3 (computer time 822 s) and -0.298 10^{-7} m^3 (computer time 850 s). We have thus the same behaviour than in 2D and the explanation is the following: the flux correction algorithm recomputes the depth with the continuity equation. In the original continuity equation, which is the wave equation, we have to solve a linear system with a matrix that contains a Laplacian, due to the implicit treatment of the free surface gradient stemming from the momentum equation. Then this free surface gradient is incorporated in the advection field and is thus explicit, the matrix becomes a diagonal (mass-lumping is mandatory). The resulting continuity equation is thus much simpler to solve in an exact way. The conclusion is that when this flux correction is applied, the continuity equation is exactly solved, whatever the accuracy asked for the initial linear system. Consequently, even if there is no tidal flat, the algorithm will have an effect on depth to get an exact continuity equation.

1.6.2 The Wesel river

With Telemac-2D

This river hydraulics case, a steady state flow, was provided by the BAW Karlsruhe. It is numerically speaking very tough, with very high Courant numbers (time steps of 2 minutes) and a locally highly refined mesh (17340 elements) including many groynes. There are 360 steps in the computation. Table 1.3 summarizes the minimum depth and computer times for 3 techniques, including masking of dry elements. These masked elements may contain hidden negative depths which reappear in the post-treatment, as is the case here. Note that the smoothing technique manages to limit the negative depths at -1 cm. Figure 7.2 shows the water depth in a small part of the domain, the smoothing technique and the flux correction give very similar results. The difficulty of the problem shows in the computer time of the flux correction technique. 12 iterations are necessary here. The minimum depth to be corrected is -0.22 m. The initial sum of absolute values of all fluxes is 337119.13 m^3/s. At the end of the process 6.67 m^3/s only are discarded as generating negative depths.

	minimum depth	**computer time (HP C3700)**
smoothing	- 0.01 m	42 s
flux correction	0 m	65 s
masking	- 0.085 m	43 s

Table 1.3: minimum depth and computer time on the Wesel test-case

With Telemac-3D

A remaining problem is that this test-case does not work properly in 3D, unless the time step is reduced to 50 s instead of 100 s, and with a diffusion of 2 m^2/s, whereas the smoothing depth option works without diffusion. Large horizontal velocities appear on groynes, i.e. in highly refined zones, and we notice a sensitivity to the constant chosen to change finite element fluxes into finite volume fluxes. This is exemplified in Figure 7.3 which shows the velocity field near a groyne in red, with the underlying free surface elevation in colored surfaces. The conclusion is different from the case with tracers in 2D, where Leo Postma's constant was the best.

1.7 Discussion

It is worth noticing that this algorithm may be applied also when there is water everywhere. In this case we may have also a temporary limitation of fluxes during the iterations (especially with large Courant numbers) but eventually all the fluxes should be accepted. A test $\sum abs(\Phi_{ij}) = 0$ has thus been added to the cases that stop the iterations.

Another point is that it is very amazing to see that there is no specific CFL limitation of the process, which is on the contrary a strong limitation of explicit schemes. The price to pay is in the iterations, but it happens that it is much less

restrictive than real explicit schemes when they want to ensure positive depths. This fact will occur again in the next chapter.

It seems easy at first sigh to extend this technique to 3D, as the continuity equation is broadly the same. As a matter of fact the smoothing of negative depths was also used in 3D without extra problems.

Last but not least: the extension to 3D is straightforward as it works on the same continuity equation.

Chapter 2

Sisyphe: the positive depths algorithm applied to non erodable beds

The Exner equation in Sisyphe, for bed-load transport, is formally similar to the Saint-Venant continuity equation as it reads:

$$\frac{\partial Z_f}{\partial t} + \mathrm{div}(\overrightarrow{Q}_s) = 0$$

where Z_f is the bottom and $\overrightarrow{Q}_s$ the solid discharge. When there are non erodable beds, a new constraint is that Z_f must not be lower than Z_r, the elevation of non erodable bed. So far non erodable beds were dealt with by an a priori treatment of $\overrightarrow{Q}_s$ ensuring the required property. A limitation factor g for $\mathrm{div}(\overrightarrow{Q}_s)$ was first computed, then a second limiting factor f was deduced such that $\mathrm{div}(f\ \overrightarrow{Q}_s) < g\,\mathrm{div}(\overrightarrow{Q}_s)$, and the form $\mathrm{div}(f\ \overrightarrow{Q}_s)$ was eventually used in Exner equation, thus ensuring mass conservation (see reference [6]). This was however a rather tedious procedure. We now consider that $Z_f - Z_r$ is an available layer of sediment, that must not become negative, the new equivalent equation is:

$$\frac{\partial (Z_f - Z_r)}{\partial t} + \mathrm{div}(\overrightarrow{Q}_s) = 0$$

with the same constraint than the water depth for $Z_f - Z_r$. In the variational formulation the term $\mathrm{div}(\overrightarrow{Q}_s)$ is integrated by parts and thus split into boundary fluxes and internal fluxes. These internal fluxes can be tranformed into point to point fluxes like in previous chapter, then the problem is exactly like moving water between points, the sediment replaces water. For graded sediment, each class has its own layer (which in Sisyphe is a product AVAIL * ELAY) and it leads to a series of problems where every layer must remain positive. With a saturated bed load equation, discarding some fluxes is even more natural than with the Saint-Venant continuity equation. As a matter of fact, the solid discharge is only a potential discharge that assumes that sediment is available for movement. When there is no sediment, it is

normal to ignore that flux. Iterations are necessary, just in case sediment is brought within the time step to a point that was previously bare. Figure 7.4 compares the old and the new technique on a Sisyphe schematical test-case. Sediment in a square hole is washed away by the flow. No limitation is prescribed on the bed slope here so the heap of sediment is not very physical, but shows the similar behaviour of both techniques which, given their very different approach, is a good cross-validation. In this example the new technique is much faster.

Chapter 3

Telemac-3D: a finite volume advection solver

The main idea is to design an advection solver close to what is done with the distributive scheme, i.e. with a splitting of time to ensure the CFL condition, with the only modification that the fluxes taken into account are not the finite element fluxes, but rather the finite volume point to point fluxes which are currently built for the Delwaq interface and for the treatment of negative depths in 3D. In 3D and with distributive schemes (full explanations on distributive schemes are given in reference [1] from page 183 on) we just recall here that we end up in a discretization of advection equation in the form:

$$S_i(\frac{C_i^{n+1} - C_i^n}{\Delta t}) = -\beta_i \Phi_T \tag{3.1}$$

Where S_i is the integral of the test function of point i, β_i is a distribution coefficient, Δt the time step, $C_i^{n+1}$the value of function C at point i after advection, C_i^n the value of function C at point i before advection. Φ_T actually represents $S_i(\overrightarrow{u}.\overrightarrow{grad}(C))$ where $\overrightarrow{u}$ is the advecting field. The only thing to know here is that $\beta_i \Phi_T$ is eventually put in the form:

$$\beta_i \Phi_T = \sum_j \lambda_{ij}(C_i - C_j) \tag{3.2}$$

where the coefficients λ_{ii} are zero and all coefficients λ_{ij} are positive or zero. The sum is done on all the neighbouring points of point i, i.e. all the points that belong to an element containing i.

The final value of function C at point i is thus:

$$C_i^{n+1} = C_i^n \left(1 - \sum_j \lambda_{ij} \frac{\Delta t}{S_i}\right) + \sum_j \frac{\Delta t}{S_i} \lambda_{ij} C_j^n \tag{3.3}$$

Classically, it is said then that the monotonicity criterion consists of ensuring that all the coefficients of f_i^n and f_j^n be positive, which yields, given the fact that all λ_{ij} are positive or zero:

$$\Delta t \leq \frac{S_i}{\sum_j \lambda_{ij}} \tag{3.4}$$

In finite volumes with fluxes between points, the corresponding equation would be:

$$C_i^{n+1} = \left(1 - \frac{\Delta t}{S_i} \sum_j \max(\Phi_{ij}, 0)\right) C_i^n + \sum_j \frac{\Delta t}{S_i} \max(\Phi_{ij}, 0) C_j^n$$

assuming that we start from element by element fluxes in the form:

$$\Phi_i^{el} = \int_\Omega \overrightarrow{u} . \overrightarrow{grad}(\Psi_i) \; d\Omega$$

this explains the difference of signs with what was done in 2D where we start from fluxes in the form:

$$\Phi_i^{el} = -\int_{\Omega 2d} h \overrightarrow{u} . \overrightarrow{grad}(\Psi_i) \; d(\Omega 2d)$$

which are fluxes leaving points. This leads us to the following CFL condition:

$$\Delta t \leq \frac{S_i}{\sum_j \max(\Phi_{ij}, 0)} \tag{3.5}$$

Implementation should be the same provided that we change everywhere λ_{ij} by $\max(\Phi_{ij}, 0)$.

In fact, the distributive scheme implementation in subroutine murd3d.f in Telemac only uses $\sum_j \lambda_{ij}(C_j^n - C_i^n)$ which is array $TRA02$ and $-\sum_j \lambda_{ij}$ which is put in array DB. A finite volume upwind scheme can then be easily implemented if we take:

$$DB = -\sum_j \max(\Phi_{ij}, 0)$$

and

$$TRA02 = \sum_j \max(\Phi_{ij}, 0)(C_j^n - C_i^n)$$

Starting from the coefficients Φ_i^{el} (six per element), the point to point coefficients Φ_{ij} are obtained as is done in the interface to Delwaq, i.e.:

1) assembling element by element fluxes on points: horizontal fluxes will be a sum of two contributions (from the upper and lower layer, except on the bottom and the free surface).

2) changing the horizontal fluxes of points 1, 2, and 3 in the prism into point to point fluxes.

Points 1 and 2 are done in a subroutine called fluxvfleo_3d, which calls the 2D subroutine fluxvfleo.f

3) cancelling all crossed fluxes (between points 1 and 5, 1 and 6, 2 and 4, 2 and 6, 3 and 4, 3 and 5)

4) finding the vertical fluxes that solve the continuity equation. Comparing what is done in subroutine tridw2.f and in subroutine tel4del.f, we find that the vertical fluxes are such that:

$$\Phi_{ij} = -\Delta z \; W^* \int_{\Omega 2d} \Psi_k d(\Omega 2d)$$

where Φ_{ij} is the vertical flux between to points i and j located in the same prism, on the vertical of the 2D point k. On a total of 15 possible fluxes in the prism, we thus retain 9 non-zero terms, which is exactly what is done also with the N-scheme, with the same vertical fluxes. Both schemes are actually very close, though different, and it was very easy to add the new finite volume scheme as a variant of distributive scheme, in subroutine murd3d.f. Figure 7.5 compares the 3 available options on the test-case of the lock-exchange.

Chapter 4

Telemac-2D and 3D: new advection schemes designed for tidal flats

4.1 Principle

The upwind explicit finite volume scheme was the only one in Telemac-2D to ensures mass conservation of tracers in the sense of depth-averaged concentrations or temperatures. This scheme could not be used so far with tidal flats because there is a division by the depth in the derivation, and the CFL number tends to infinity on dry zones. We had then to resort to a masking of dry elements, along with a clipping of depth, which was not mass-conservative. The flux correcting technique presented in the first chapter leads in fact straightforwardly to a new advection scheme which is not sensitive to dry zones.

In the reference [3] we have presented the upwind explicit finite volume advection scheme. With the same notations as in Chapter 1, the new concentrations at a point i were given by the formula:

$$C_i^{n+1} = (1 + \frac{\Delta t}{h_i^{n+1} S_i} \sum_{\text{negative } \Phi_{ij}} \Phi_{ij})\, C_i^n - \frac{\Delta t}{h_i^{n+1} S_i} \sum_{\text{negative } \Phi_{ij}} C_j^n \Phi_{ij} \tag{4.1}$$

The monotonicity condition was that:

$$\Delta t < \frac{h_i^{n+1} S_i}{\sum_{\text{negative } \Phi_{ij}} |\Phi_{ij}|} \tag{4.2}$$

which may turn into 0 on dry zones. The reason is that the inputs and outputs on one point may result in a negative or zero depth, while the remaining quantity of tracer may not be zero. The fundamental reason is that all fluxes to and from a point are considered at the same time. Imagine now that we do it edge by edge, thus considering only two points at a time. Let us number these points 1 and 2 and let us assume that

the flux Φ_{12} is positive, i.e. the water goes from point 1 to point 2. The conservative tracer equations of both points read simply (we omit boundary and source terms for simplicity):

$$\frac{S_1}{\Delta t}(h_1^{n+1}C_1^{n+1} - h_1^n C_1^n) + \Phi_{12}C_1^n = 0 \tag{4.3}$$

$$\frac{S_2}{\Delta t}(h_2^{n+1}C_2^{n+1} - h_2^n C_2^n) - \Phi_{12}C_1^n = 0 \tag{4.4}$$

C_1^n appears in both equations because the flux goes from 1 to 2 (upwind scheme). If we also treat the continuity equation only for this edge (what we do in the flux limiting algorithm), we have also:

$$h_1^{n+1} = h_1^n - \frac{\Delta t}{S_1}\Phi_{12}$$

$$h_2^{n+1} = h_2^n + \frac{\Delta t}{S_2}\Phi_{12}$$

It then turns out that equations 4.3 and 4.4 become simply:

$$C_1^{n+1} = C_1^n \tag{4.5}$$

$$C_2^{n+1} = \frac{h_2^n}{h_2^{n+1}}C_2^n + (1 - \frac{h_2^n}{h_2^{n+1}})C_1^n \tag{4.6}$$

In this context there is no risk of division by 0 because we started from a positive depth h_2^n which was increased by the positive quantity $\frac{\Delta t}{S_2}\Phi_{12}$. h_2^n/h_2^{n+1} is then in the range $[0, 1]$. The positivity and monotonicity of tracers is ensured even on dry zones (in case of zero depth the concentrations remain unchanged). This edge by edge treatment (only a few lines of Fortran) may be inserted within the previous tidal flats algorithm.

4.2 Domain decomposition in parallelism

We have mentioned in Chapter 1 that the internal fluxes were not assembled in parallel, but could be averaged on interfaces, so that the upwinding information is the same on either sides. This trick is used here for the tracers. However it is not the only thing to do. Let us assume that a point is shared between 3 processors, a, b and c. It thus exists in 3 locations in memory, with concentrations C_a, C_b, C_c and depths h_a, h_b, h_c. At the end of the parallel communication the 3 depths are added so that every processor gets $h_a + h_b + h_c$. The relevant conservative merging of concentrations gives the common value of $\frac{C_a h_a + C_b h_b + C_c h_c}{h_a + h_b + h_c}$. This is obtained by creating an array containing the product Ch, and running the parallel communication that adds contribution of interface points (PARCOM with option 2). The same is done with h and we then do the division. This is a case where we have a division by a depth, which is done only if the depth is strictly positive.

4.3 Extension to Sisyphe

In Sisyphe, the velocity field for advection may be corrected by a factor F in the range $[0, 1]$ to account for the fact that the suspended sediment is concentrated near the bottom, where the velocity is smaller than the depth-averaged velocity. A special care is then necessary: the fluxes for computing the positive depths must be done with the original velocity field, which is consistent with the continuity equation. Then the fluxes to consider for the tracer advection must have the coefficient F. We have then to come back to the original tracer equations 4.3 and 4.4, which now read:

$$\frac{S_1}{\Delta t}(h_1^{n+1}C_1^{n+1} - h_1^n C_1^n) + F\ \Phi_{12}C_1^n = 0 \tag{4.7}$$

$$\frac{S_2}{\Delta t}(h_2^{n+1}C_2^{n+1} - h_2^n C_2^n) - F\ \Phi_{12}C_1^n = 0 \tag{4.8}$$

if we assume that Φ_{12} is positive and apply upwinding also on the F coefficient. It gives now:

$$C_1^{n+1} = \left[\frac{h_1^n}{h_1^{n+1}} + F_1(1 - \frac{h_1^n}{h_1^{n+1}})\right] C_1^n \tag{4.9}$$

$$C_2^{n+1} = \frac{h_2^n}{h_2^{n+1}}C_2^n + F_1(1 - \frac{h_2^n}{h_2^{n+1}})C_1^n \tag{4.10}$$

This solution does not ensure monotonicity of point 1, which is a known drawback of using a corrected velocity field. It does not seem practicable here as the interpolation coefficients may tend to infinity. A solution could consist of dealing with erosion and deposition at the same time, in order to get a more physical behaviour. Advection would be done on the depth-integrated quantity of sediment hC. Then, depending on the final depth, it would be decided if this quantity would remain in the water or would deposit.

4.4 Extension to 3D

Once fluxes between points are known (this is done in the next chapter for a finite volume scheme) there is absolutely no difference between 2D and 3D. In 2D, quantities like $S_1 h_1^{n+1}$ and $S_2 h_2^{n+1}$ are volumes of water carried by points 1 and 2. In 3D the volumes carried by points will be simply the integral of test functions (which have been purposedly called VOLUN and VOLU in Telemac-3D). Note that the sum of all these volumes is the integral of 1 over the whole domain, hence the total amount of water. At the beginning of a computation these integrals are VOLUN, when all the fluxes between points have been transferred they are equal to VOLU. The algorithm in 3D is otherwise exactly the same as in 2D. This new advection solver may be used on the velocities, either in 2D or 3D. In this case the continuity equation is not yet done and the new depth will only be compatible with the advection velocity at the beginning of the time step, hence it will be only a predictor of the final depth.

4.5 Extension to distributive schemes

In fact the idea may be applied to any advection scheme able to provide point to point fluxes. This is actually the case of distributive schemes and the procedure could be extended to the N-scheme without difficulty and with similar results. However the PSI scheme raises an extra difficulty. With this latter technique the tracer point to point fluxes are modified in a way that does not affect the overall contributions to points. An extreme case occurs when the tracer is constant in space, then no flux at all is considered. The compatibility with water fluxes is then lost, so that passing tracers along with water is no longer possible. Building an edge by edge PSI scheme thus remains an open problem.

4.6 Tests

4.6.1 Tracer in the bridge piers test-case

Our first test has no tidal flats at all. It is just a comparison of advection schemes to assess their numerical diffusion. We use the bridge piers test-case and enter a tracer with a value of 1 at 3 points in the entrance. The time step is 0.8 s and there are 100 steps in the computation. There is no diffusion. Six different results are plotted on Figure 7.6. On the right are displayed the results given by the upwind explicit finite volume scheme, the method of characteristics, and the Positive Streamwise Invariant distributive scheme. This latter scheme is reputed mass-conservative, but not here in the context of shallow water equations, because it is applied to the concentration, which is not the conservative variable. On the left 3 variants of the present scheme, depending on the choice of the constant to get the element fluxes (see discussion in reference [3] page 33 and in reference [2]). It happens that the choice of this constant has dramatic effects. Only with the original solution introduced by Leo Postma do we get a correct advection scheme, comparable to other schemes. The conservation of mass (of water and tracer) is ensured at the accuracy of the machine (provided that direct solvers are chosen). It can be noted that the method of characteristics and the PSI scheme are less diffusive.

4.6.2 Thermal plume in tidal conditions

We study here a thermal plume in 2 dimensions, i.e. with depth-averaged temperature, on the small domain. The boundary conditions, subjected to tidal conditions, are given by a larger model. For checking the conservation of heat, the diffusion is removed. As a matter of fact this step is actually applied on the temperature and is conservative for this variable, which is not the integral of temperature on the vertical (which is the real conservative variable). The hot water is released by 16 source points of 10.5 m^3/s each with a velocity of 1.51 m/s towards West, and an increment of temperature of $10.6°C$. There is no exchange with atmosphere. The computation consists of 20000 steps of 50 s each, i.e. 11 days 13 hours 46 mn and 40 s. The total heat (temperature multiplied by volume) entered in the domain is 0.17808 10^{10}. When the flow is entering the domain an increment of temperature of 0 is prescribed. At low tide there are dry zones everywhere in the domain, including on the boundaries. Figure 7.7 shows the thermal

plume at the end of the computation, obtained with the method of characteristics, the PSI scheme, and the present edge by edge flux correcting scheme. Characteristics and flux correction are in broad agreement, though the latter appears to be more diffusive. The PSI schemes overestimates temperatures. The figures of Table 4.1 give more explanations on the comparison. CPU time is given on a HP C3700 workstation. Both the PSI scheme and characteristics create an excess of about 20% of heat, but this excess gets out of the domain in the case of characteristics. It is worth noting that, to avoid loss or gain of heat, the Dirichlet boundary conditions of temperature are discarded. Only the correct fluxes are considered. The values observed at boundaries may thus appear to be slightly different from prescribed values (this point is also very important in the Berre lake study).

	CPU time	min. depth	% heat lost	final heat	exited heat
characteristics	5116 s	-0.0114 m	-19.3	$0.467\ 10^9$	$0.166\ 10^{10}$
PSI scheme	5623 s	-0.0114 m	-17.7	$0.836\ 10^9$	$0.126\ 10^{10}$
flux correction	7202 s	0 m	$-7\ 10^{-11}$	$0.646\ 10^9$	$0.113\ 10^{10}$

Table 4.1: comparing advection schemes on a thermal plume test case

On a 2-core Dell Linux workstation, the CPU time in parallel, for the 3 advection schemes, is summarized in Table 4.2. As the method of characteristics is already used for the advection of depth in the momentum equation, it is obviously the most efficient technique here in terms of computer time, as the advection of tracer is only an extra interpolation. It is also worth noticing that the flux correction procedure is done twice, for simplicity of implementation: once for getting positive depths, once again for the tracer. It is clear by comparing Table 4.1 and Table 4.2 that the efficiency of algorithms highly depends on the machine (which is a combination of architecture + compiler). On Dell in parallel the PSI scheme and the new scheme have the same cost.

	CPU time with 2 processors on Dell Linux
characteristics	1286 s
PSI scheme	1495 s
flux correction	1494 s

Table 4.2: comparing advection schemes in parallel

4.6.3 Dam breaks

The 1D dam break exact solution will enable us to show the interest of the new scheme when applied to the advection of velocities. Figure 7.8 compares the free surface elevation obtained with the method of characteristics (in red) with the new scheme (in green). The exact solution is in blue. It appears clearly that the method of characteristics is hindered by tidal flats. In this ideal case, the wave front progresses only with advection. There is no propagation velocity on the front because the depth is zero.

We then tested the Malpasset dam break in 3D, with 2 planes on the vertical. As in the previous case, the new finite volume advection solver is applied to velocities. Figure 7.9 compares the depths after 40 mn, with the method of characteristics or with the new scheme (which is scheme option 13 for key-words SCHEME FOR ADVECTION OF VELOCITIES or SCHEME FOR ADVECTION OF TRACERS). The surprise is the difference which does not appear so large in 2D, but this difference is in favour of the new scheme, since it is known that the wave reached the sea before 40 mn. As in the 1D example this may be explained by the fact that the method of backward characteristics is not very good on tidal flats, since no characteristic on a tidal flat (thus with no velocity) will go backward to get information on the coming wave. The difference with 2D results was investigated, and an explanation could be found and is explained here: in 2D the friction terms in the non conservative momentum equation are multiplied by a factor $1/h$ where h is the depth. This depth was so far taken as the nodal value. In 3D there is no obvious $1/h$ factor but it is implicitly hidden in a factor:

$$\frac{\int_{\Omega 2D} \Psi_i \; d(\Omega 2D)}{\int_{\Omega 3D} \Psi_i \; d(\Omega 3D)}$$

which stems from the variational formulation of the boundary terms. The difference is that when h is zero in 2D friction becomes infinite, velocity is cancelled and it reduces the wave celerity. When h is zero in 3D, the denominator may not be zero if one neighbour of the given point has a depth, because the volume associated with the corresponding test function will not be zero. In this case the friction will not be infinite and this will ease the wave propagation. This is exemplified in Figure 7.10, where the effect of the new advection scheme is also clear.

4.7 Discussion

The new scheme is perfectly mass-conservative in 2D and 3D, it ensures monotonicity and is stable on dry zones. Further tests are necessary to assess its numerical diffusion. A possible drawback is a sensitivity to the mesh, probably by construction because we use the edges as a way of transit of tracer. An obvious odd property is that the result certainly depends on the numbering of edges. A necessary development before other quantitative tests would be adding the diffusion step in the depth-averaged conservative concept. It would simply consist of evaluating diffusion terms as an extra advection.

Chapter 5

FE and FV fluxes: in 2D Leo Postma scheme is the N-scheme

The way to get point to point fluxes from finite element fluxes has already been discussed in reference [3] (page 29). The problem is not well posed and a constant must be introduced. In our tests the constant chosen by Leo Postma seemed to give the best results on the rotating cone test. More recently it became obvious that the distributive schemes were another way to get the point to point fluxes and the N-scheme was tested. Though the formulas are at first sight very different, it happens that the Leo Postma scheme is exactly the N-scheme. We show it hereafter. We keep the notations of reference [3], recalled in the sketch below (a negative Φ_1 means that point 1 looses water).

The original Leo Postma formula can be implemented for every element in the following Fortran-like form (the assembly on segments has been removed to allow a better understanding):

IF(ABS(Φ_1).GE.ABS(Φ_2).AND.ABS(Φ_1).GE.ABS(Φ_3)) THEN
    Φ_{12} = - Φ_2
    Φ_{23} = 0
    Φ_{31} = Φ_3
ELSEIF(ABS(Φ_2).GE.ABS(Φ_1).AND.ABS(Φ_2).GE.ABS(Φ_3)) THEN
    Φ_{12} = Φ_1
    Φ_{23} = - Φ_3
    Φ_{31} = 0
ELSE
    Φ_{12} = 0
    Φ_{23} = Φ_2
    Φ_{31} = - Φ_1
ENDIF

The N-scheme implementation is now the following:

Φ_{12} = MAX(MIN($-\Phi_1$,Φ_2),0.D0) - MAX(MIN($-\Phi_2$,Φ_1),0.D0)

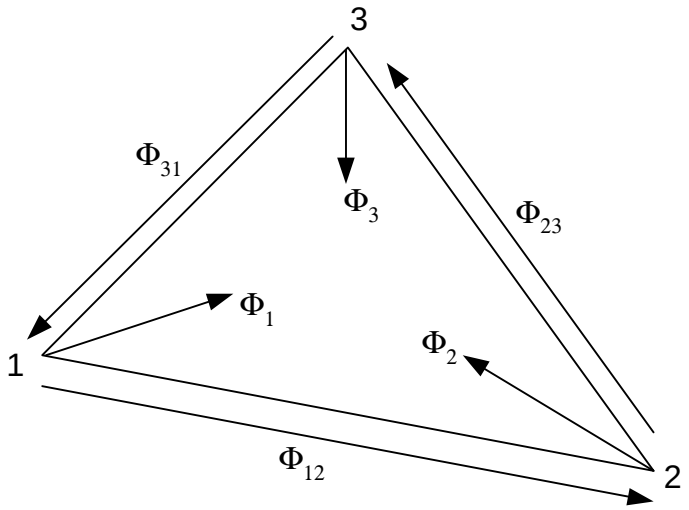

Figure 5.1: fluxes from and between points

Φ_{23} = MAX(MIN($-\Phi_2$,Φ_3),0.D0) - MAX(MIN($-\Phi_3$,Φ_2),0.D0)
Φ_{31} = MAX(MIN($-\Phi_3$,Φ_1),0.D0) - MAX(MIN($-\Phi_1$,Φ_3),0.D0)

To prove the equivalence we first remark that we can always consider that point 1 has a positive Φ_1(we could otherwise change the local numbering of points without changing the result). Then we are left with 3 cases (because the sum of all fluxes is 0):

case 1: $\Phi_1 > 0$ and $\Phi_2 > 0$ and $\Phi_3 < 0$
case 2: $\Phi_1 > 0$ and $\Phi_2 < 0$ and $\Phi_3 < 0$
case 3: $\Phi_1 > 0$ and $\Phi_2 < 0$ and $\Phi_3 > 0$

It is easy (but tedious, headache guaranteed!) to check that both methods will give:

case 1: $\Phi_{12} = 0$ and $\Phi_{23} = \Phi_2$ and $\Phi_{31} = -\Phi_1$
case 2: $\Phi_{12} = -\Phi_2$ and $\Phi_{23} = 0$ and $\Phi_{31} = \Phi_3$
case 3: $\Phi_{12} = \Phi_1$ and $\Phi_{23} - \Phi_3$ and $\Phi_{31} = 0$

Note that the N-scheme was designed to get positive fluxes, which is the case here.

The N-scheme uses 6 MAX functions and 6 MIN functions.

The Leo Postma method can be reduced to 3 ABS functions and 2 or 4 comparisons .GE., it is cheaper, but the proof given here could be translated into:

```
IF(Φ1.GE.0) THEN
    IF(Φ2.GE.0) THEN
        This is case 1
    ELSE
        IF(Φ3.GE.0) THEN
            This is case 3
```

```
            ELSE
                This is case 2
            ENDIF
        ENDIF
    ELSE
        IF(Φ2.GE.0) THEN
            IF(Φ3.GE.0) THEN
                This is case 4
            ELSE
                This is case 5
            ENDIF
        ELSE
            This is case 6
        ENDIF
    ENDIF
```

This way of writing the tests limits to 3 comparisons .GE. in the worst case (probably not worth the effort).

Chapter 6

Telemac-3D: smashed elements now possible

Up to version 5.9, only smashed elements on tidal flats or dry zones were treated. The fact that such elements had no volume precluded the use of finite-volume like numerical schemes, hence only the method of characteristics could be applied for advection. It appeared that most of the treatments done in the case of tidal flats could be applied also in the case of smashed elements with water above. There is indeed a risk of such a situation when the generalised sigma transformation is used, and a constant elevation of planes is prescribed. When the bottom goes above the prescribed elevation of a plane, a number of elements is smashed and this caused a crash of computations. To avoid this the parameter DISMIN in subroutine CALCOT guaranteed a minimum height in the elements. From version 6.0 on, it is now possible to have DISMIN=0. To be more precise there is now a DISMIN_BOT and a DISMIN_SUR, respectively for bottom and free surface, and DISMIN_BOT may be 0. The key modification for achieving is an array of integers IPBOT, of size NPOIN2 (number of points in the 2D mesh), giving the rank of the last layer with no height. if NPLAN is the number of planes on the vertical, we have for a 2D point I:

IPBOT(I)=0 if all layers are normal (height greater than 1 mm)

IPBOT(I)=3 if plane 3 and 4 are closer than 1 mm and distance between plane 4 and plane 5 is greater than 1 mm.

IPBOT(I)=NPLAN-1 if all the planes are smashed (case of tidal flats). NPLAN is the number of planes.

This new array allowed a number of specific treatments listed below:

- Friction is applied at the level IPBOT(I)+1 and all points below.
- In diffusion all points below the real bottom are treated as Dirichlet points, with the previous value as prescribed Dirichlet value. After solving the linear system the points below the real bottom are given the value of the real bottom.
- The Poisson equation for the non-hydrostatic pressure is treated in the same way as diffusion.

A general principle is that points with the same elevation on a vertical must eventually have the same physical value, so that no artificial infinite gradient is created. 1 mm is an arbitrary but reasonable choice and once it is done all the tests are on IPBOT. Figure 7.11 is an example of flow over a bump with an intermediate plane (number 4) imposed at the elevation -0.2 (it appears at elevcation -1 on the picture due to a distortion of 5 on the vertical). On the bump planes 1, 2, 3 and 4 are superimposed. Because the velocities of superimposed points are the same, only a single vector appears on the figure. For this case the new advection scheme for tidal flats is used and can deal with elements without volume. Mass conservation of salt is verified and monotonicity (here salinity between 30 and 40 g/l) is strictly ensured.

Chapter 7

Telemac-3D: note on the sigma transformation for characteristics

The theory behind the vertical displacements in the method of characteristics in 3D is very tricky and we give hereafter some explanations. The first thing to understand is that the advection equation is solved in a transformed mesh with horizontal planes. The sigma transformation is used to take into account the movement of the mesh. As a matter of fact the physical values at a point in the mesh will change due to advection terms (velocity of flow) but also due to the movement of the mesh. It is shown in reference [1] page 21, equation 2.90 that this can be simply taken into account by an advection in the transformed mesh. One would thus expect that the vertical displacements when computing the characteristics pathlines are of the form:

$$\Delta Z^* = W^* \Delta t$$

using the transformed coordinates and the transformed velocities, but we find in the implementation the following form:

$$\Delta z^* = W \, \frac{ZSTAR(IET+1) - ZSTAR(IET)}{\Delta z} \, \Delta t$$

were Δz is the local real width of the given layer between two planes and $ZSTAR$ is the transformed elevation of planes (here $IET+1$ and IET) in the transformed mesh.

The second thing to understand is then that the velocity W provided by telemac-3D to the subroutine CHARAC is in fact $W^* \Delta z$. This quantity is provided by the subroutine TRIDW2 and has the dimension of a velocity, it is linked to the real vertical velocity by the formula:

$$W = \frac{\partial z}{\partial t} + U \frac{\partial z}{\partial x} + V \frac{\partial z}{\partial y} + W^* \frac{\partial z}{\partial z^*}$$

where $\frac{\partial z}{\partial z^*}$ is Δz when we apply a sigma transformation that changes a layer of width Δz into a layer of width 1. Our vertical displacements are thus in reality:

$$\Delta z^* = W^* \frac{\partial z}{\partial z^*} \frac{ZSTAR(IET+1) - ZSTAR(IET)}{\Delta z} \Delta t$$

and $\frac{ZSTAR(IET+1)-ZSTAR(IET)}{\Delta z}$ is here to give the value $\frac{\partial z^*}{\partial z}$ that retrieves the wanted formula $\Delta Z^* = W^* \Delta t$. At this level we understand that the values of ZSTAR must correspond to a sigma transformation that changes a layer Δz into a layer of width 1. A solution would be for example that we have ZSTAR(IET)=IET. This would be simple but this is not exactly what is done.

The third thing to understand:

Actually we are free to apply any correction factor in the vertical displacements if they are taken into account also in the values of ZSTAR. In other words we can multiply the coordinates in a layer by any constant without changing the result. The interest is then that the correction factor $\frac{ZSTAR(IET+1)-ZSTAR(IET)}{\Delta z}$ could be a constant independent of the layer. Then the displacements would keep their meaning all over the mesh. Otherwise we would have to stop at every plane, and recompute a new displacement compatible with the new metrics (this is done with the generalised sigma transformation). The denominator being the real width, our correction factor will be constant if the values of ZSTAR are proportional to the real z coordinates, for example if they are the values of the array ZSTAR used for building the mesh, i.e. a sigma transformation giving the whole 3D mesh a width of 1. For a classical sigma transformation, we have then:

$$ZSTAR(IET) = \frac{IET - 1}{NPLAN - 1}$$

where NPLAN is the number of planes. The final formula thus mixes two different sigma transformations, one for computing W^*, one for the local metrics while navigating in the transformed mesh, and this is not a mistake.

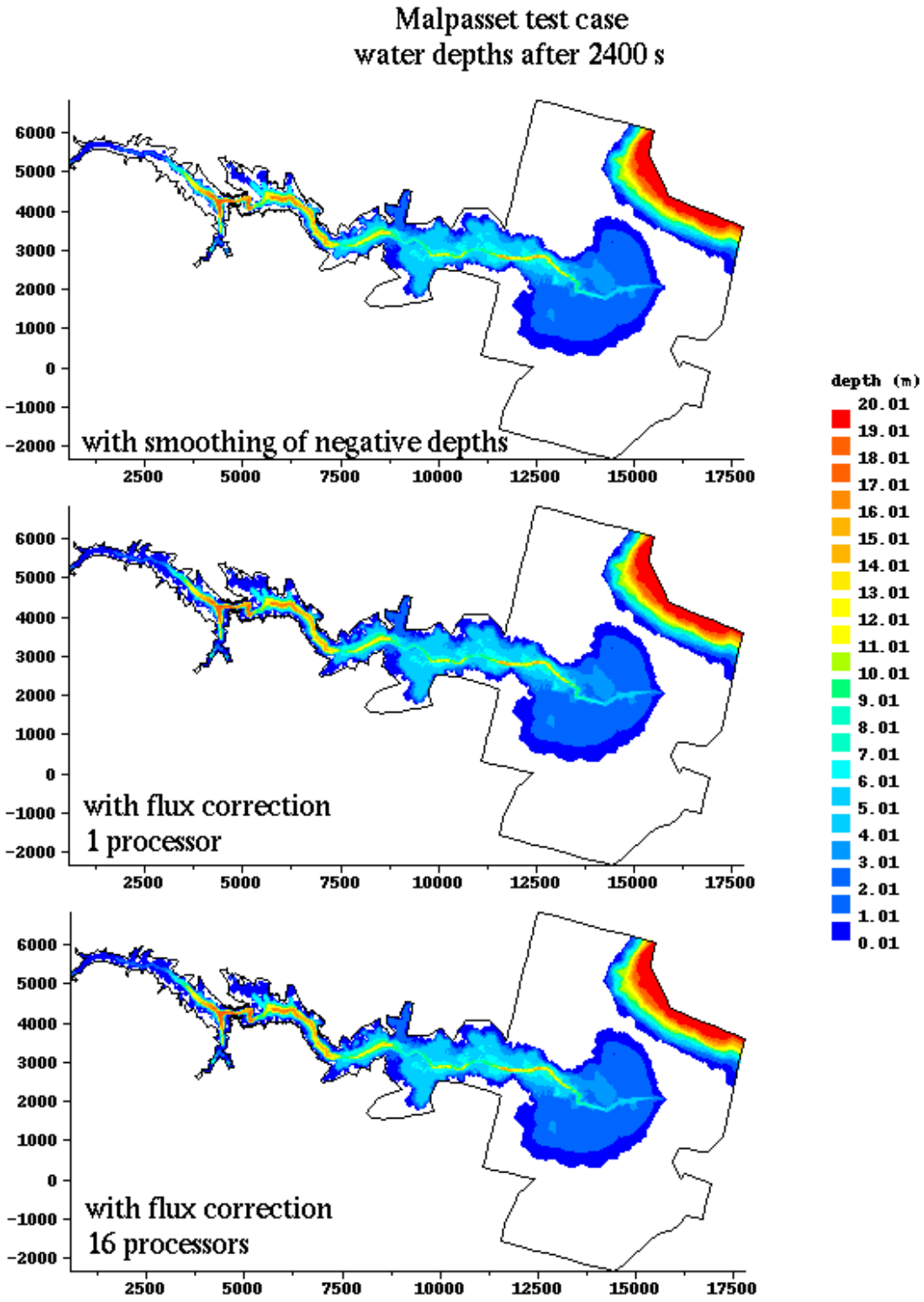

Figure 7.1: flux correction against smoothing of negative depths

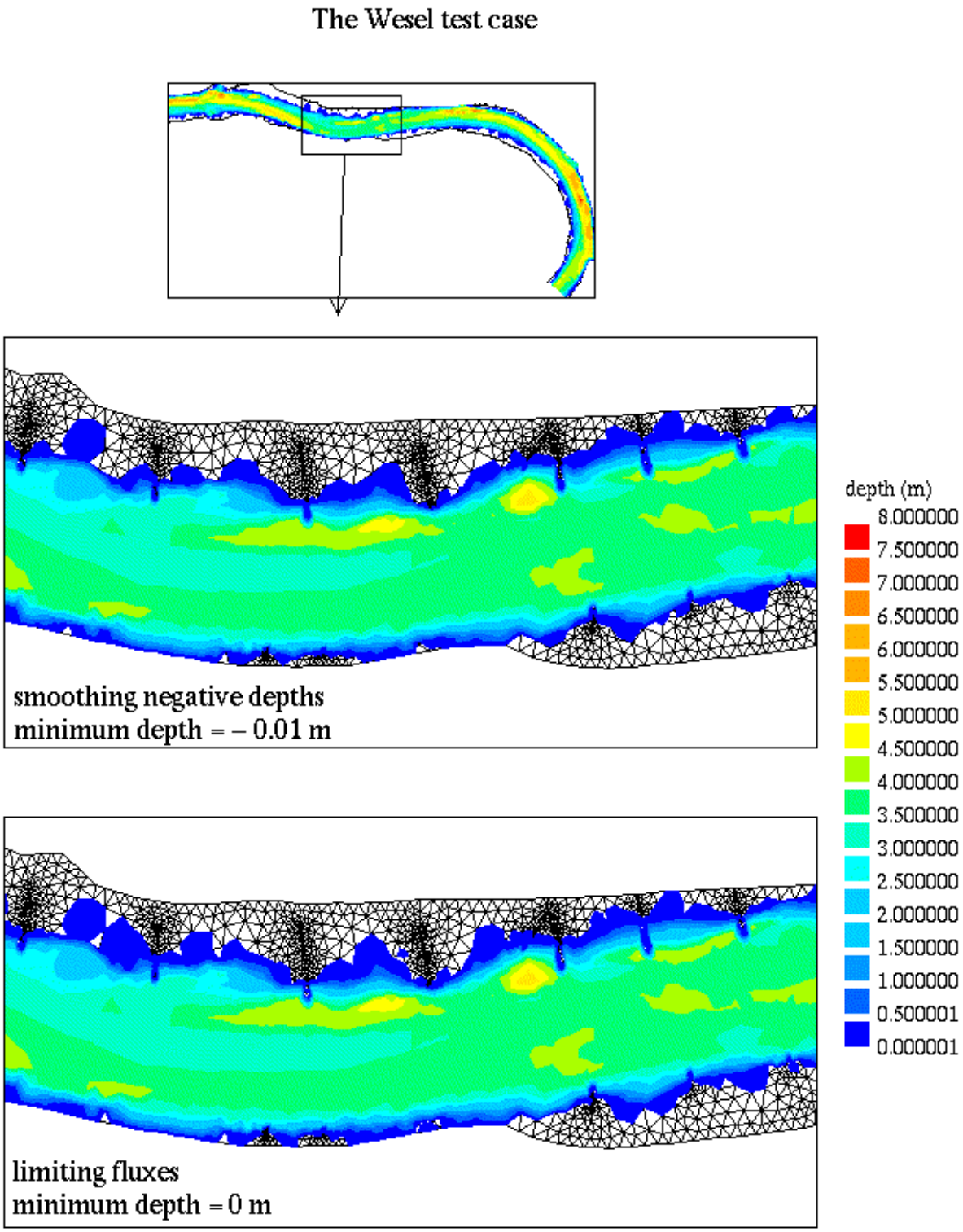

Figure 7.2: a close up view of depths comparing two techniques

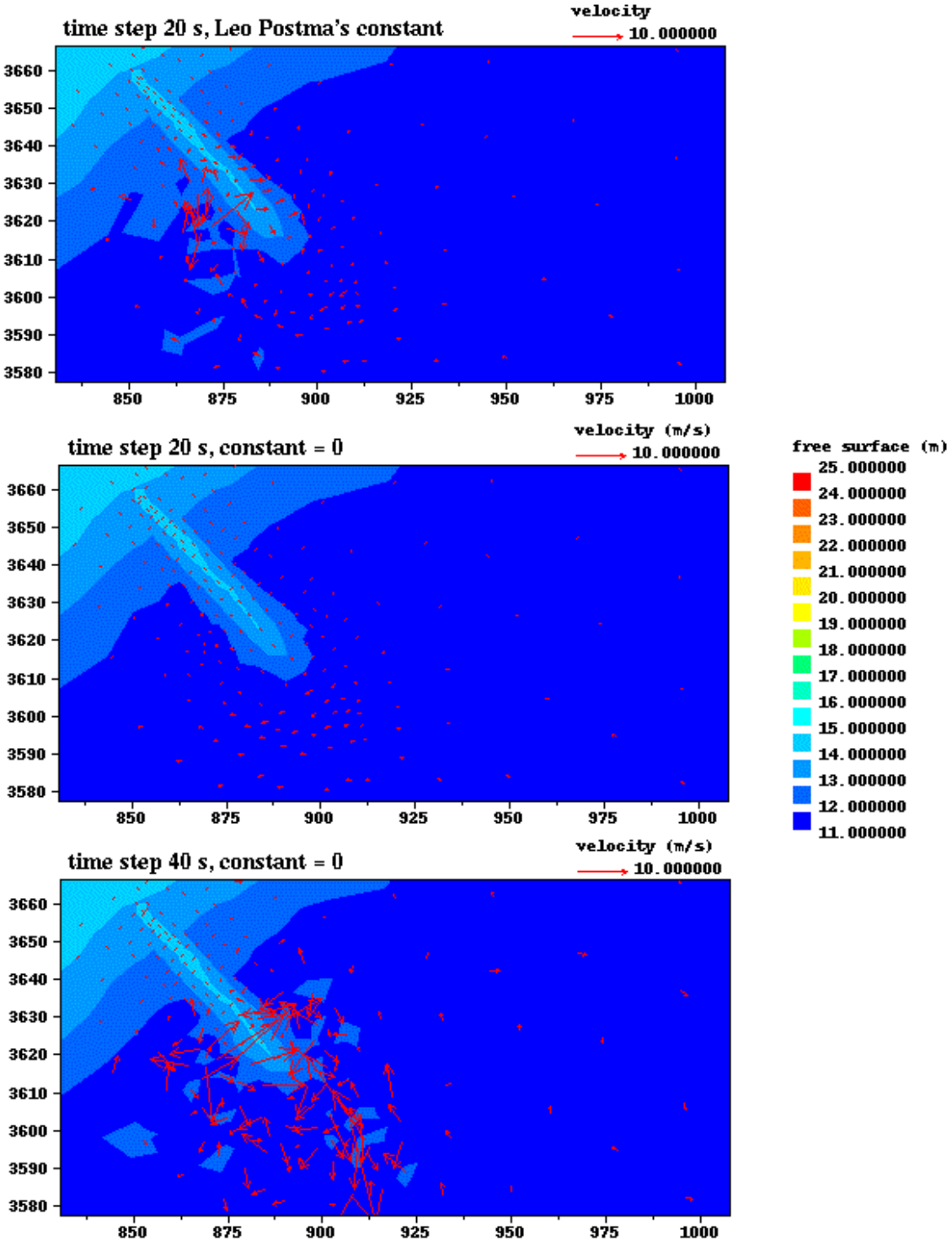

Figure 7.3: influence of time step and constant choice in the Wesel test-case

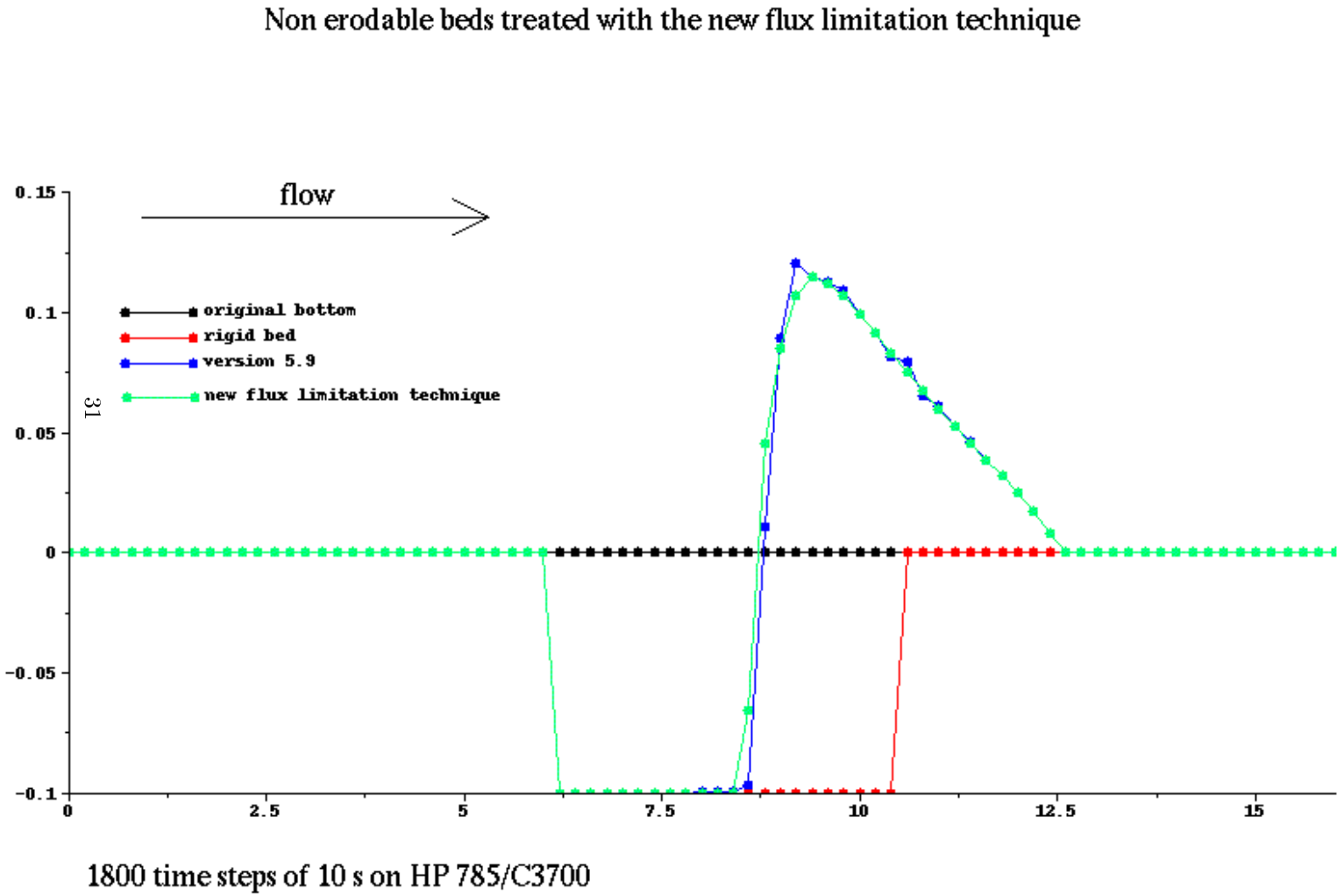

Figure 7.4: comparing the old and the new technique for non erodable beds

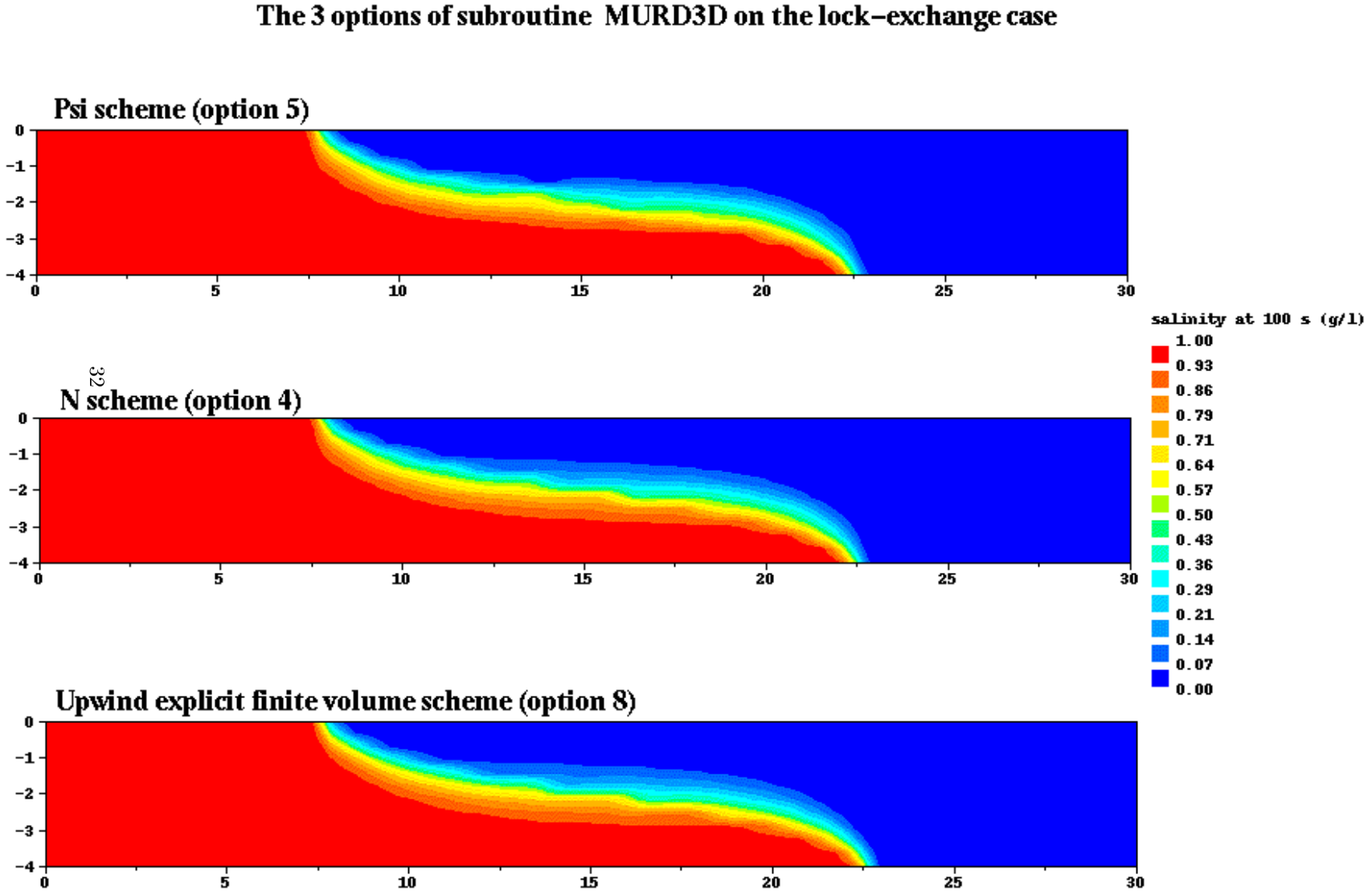

Figure 7.5: comparison of distributive and finite volume schemes

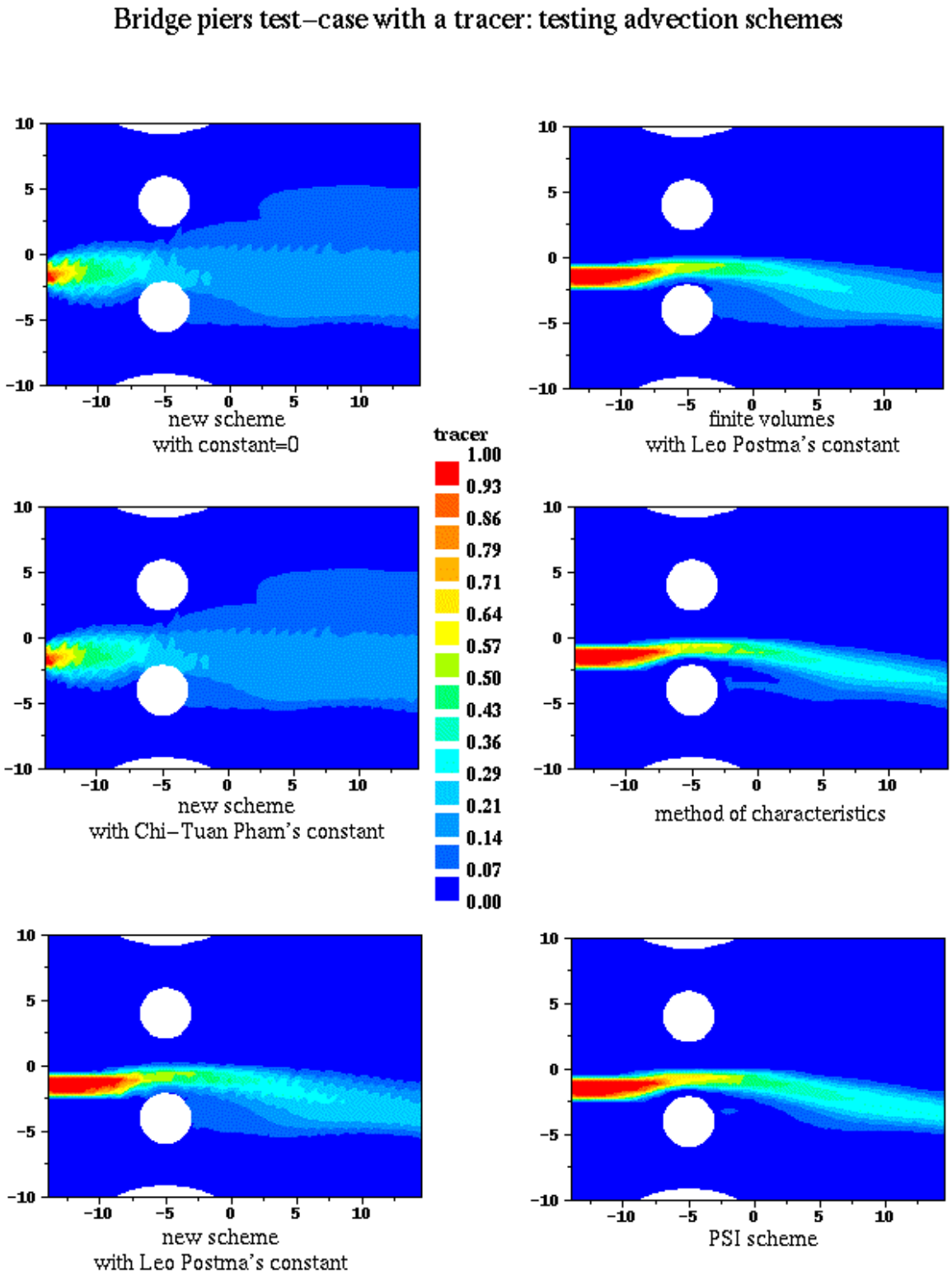

Figure 7.6: comparison of advection schemes on the bridge piers test case

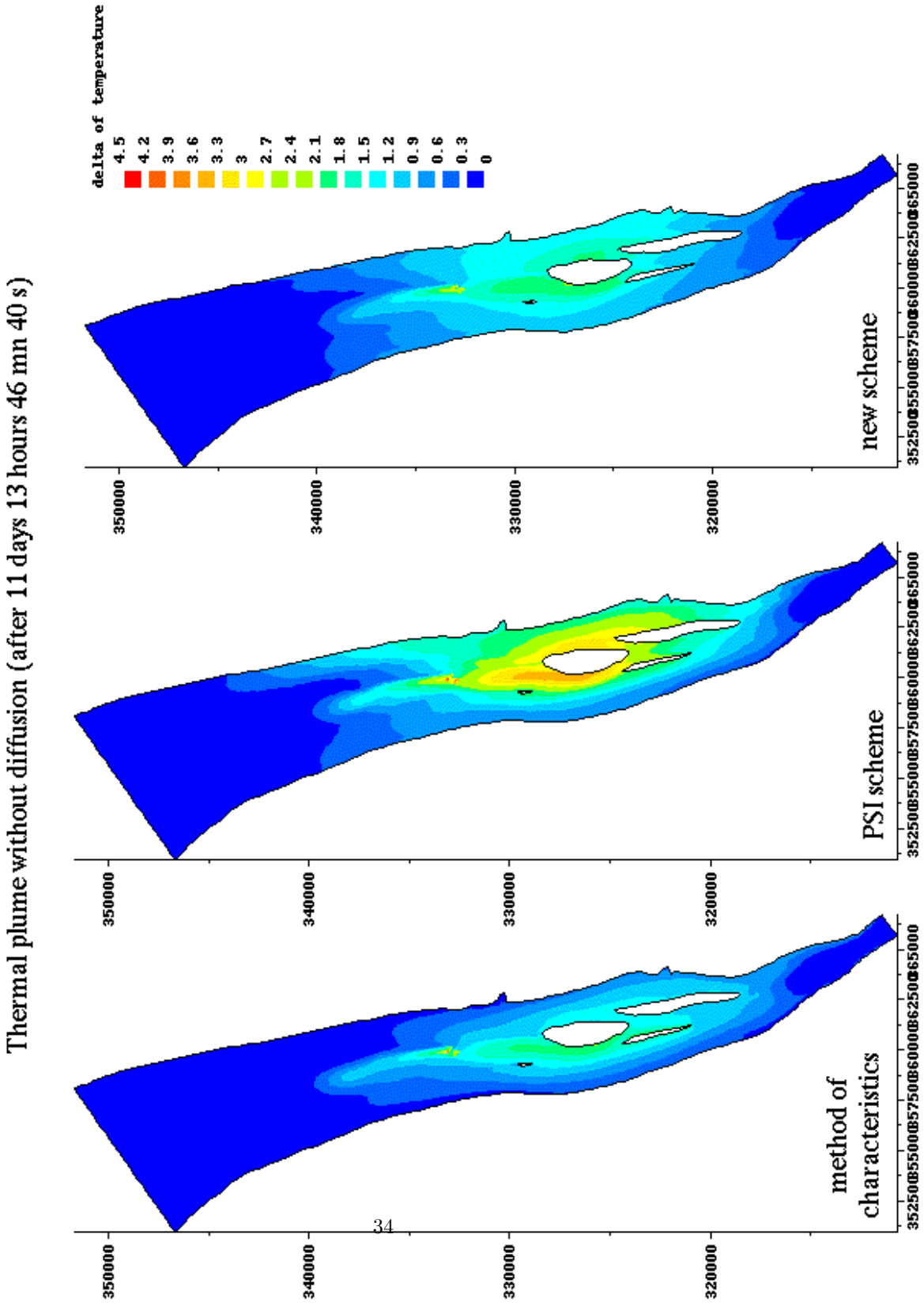

Figure 7.7: a thermal plume case, comparison of 3 advection schemes, without diffusion

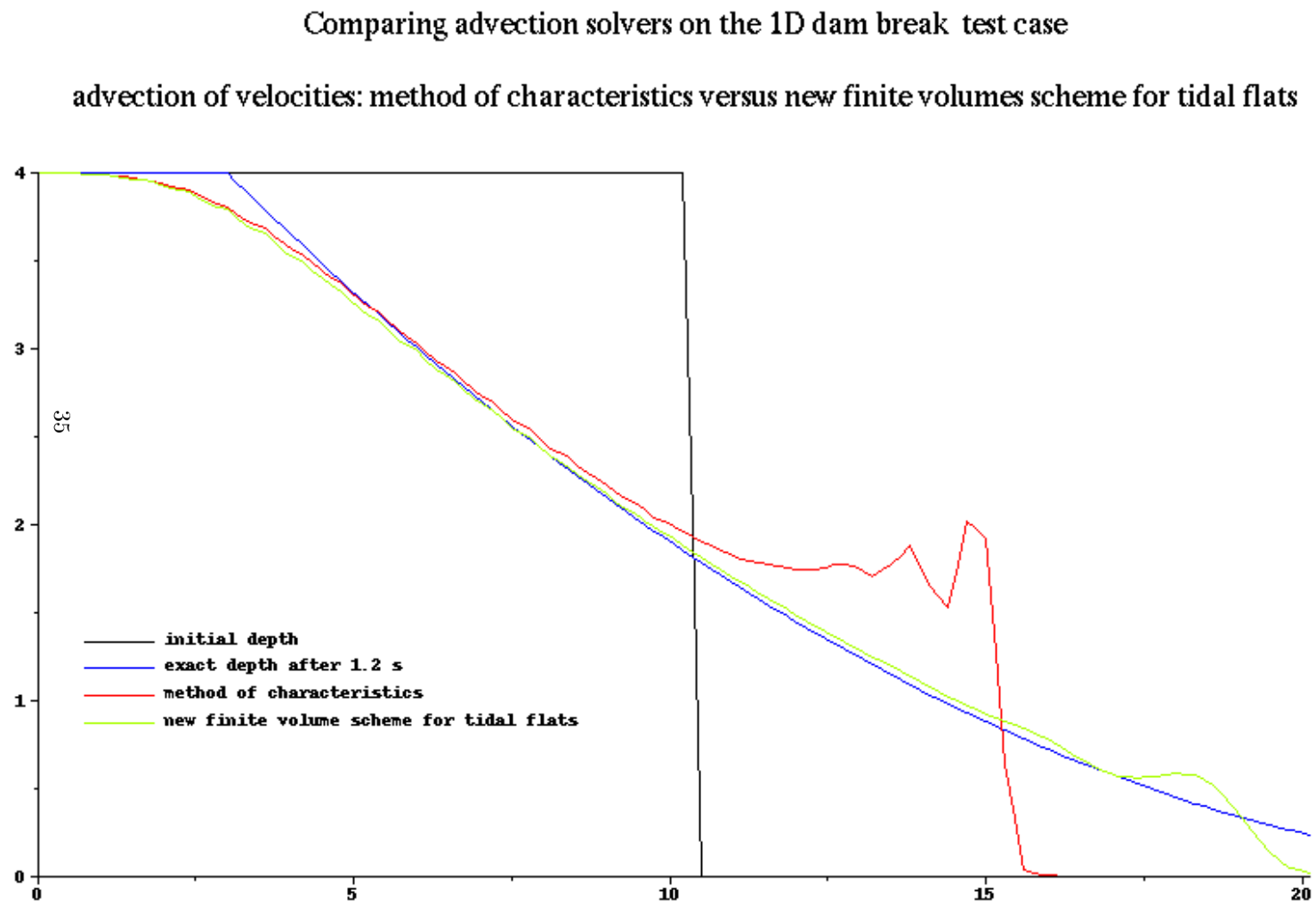

Figure 7.8: testing advection solvers applied to velocities

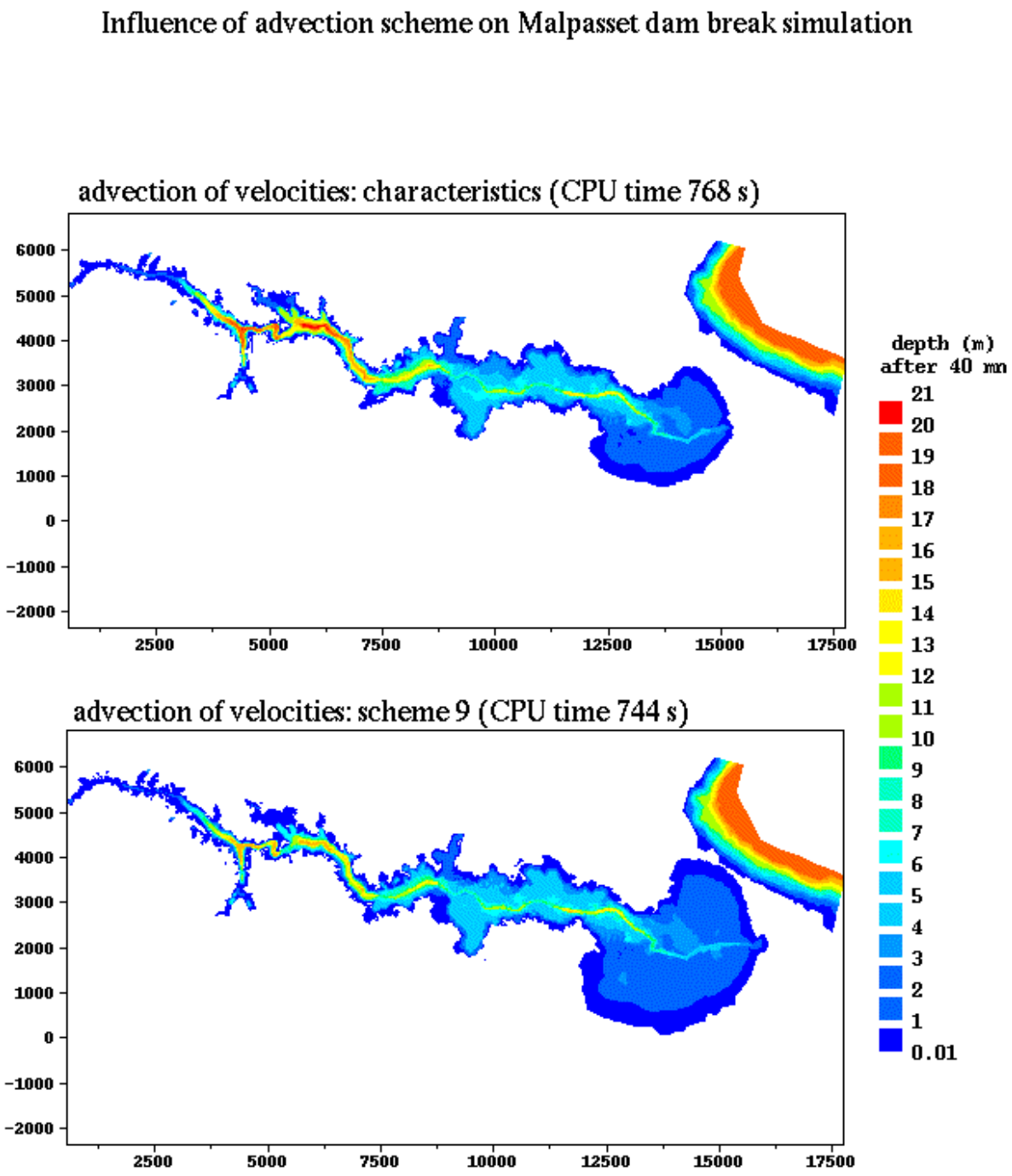

Figure 7.9: Telemac-3D: comparing advection schemes for velocities in the Malpasset test-case

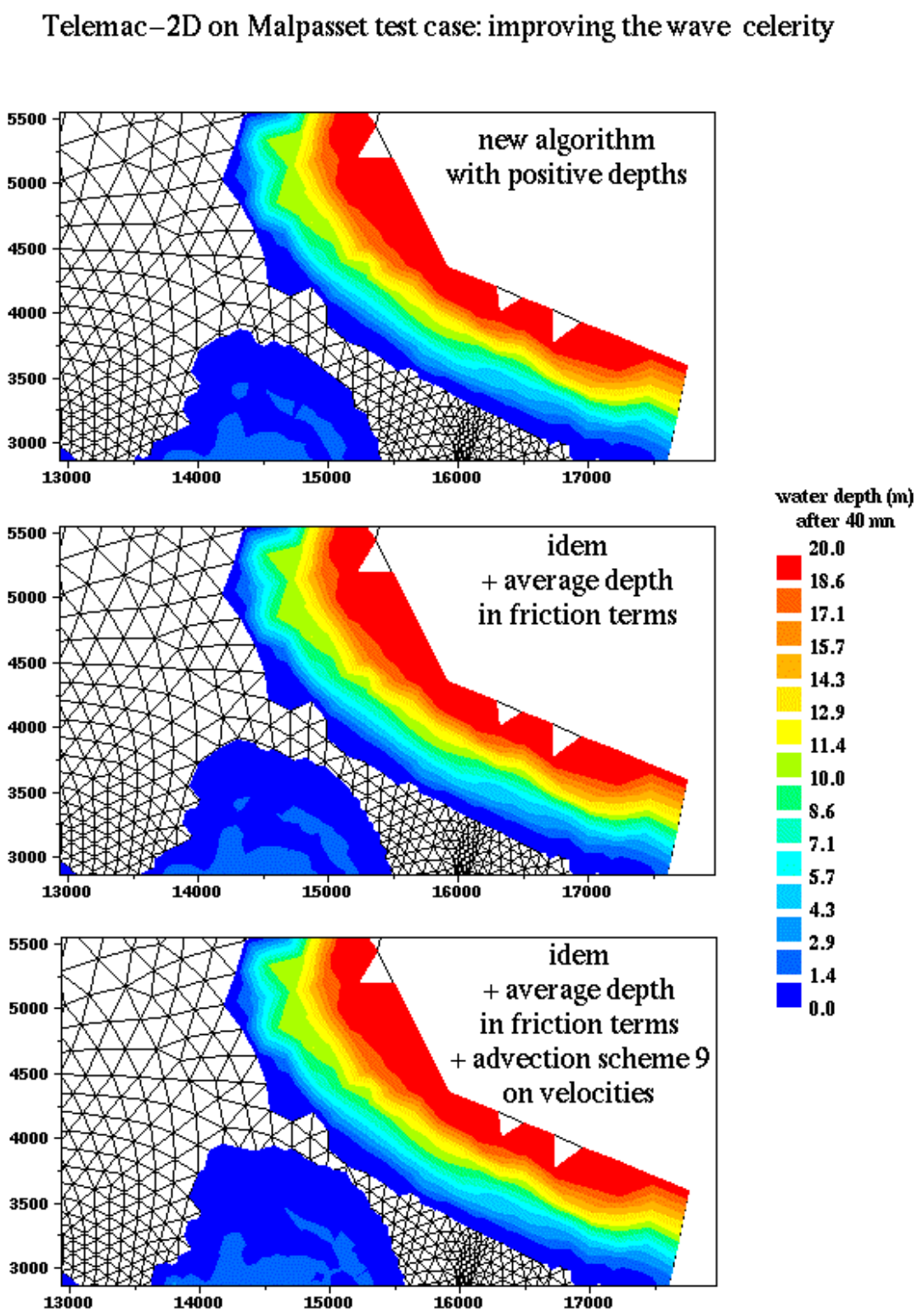

Figure 7.10: Malpasset case with Telemac-2D: effect of advection scheme and friction term

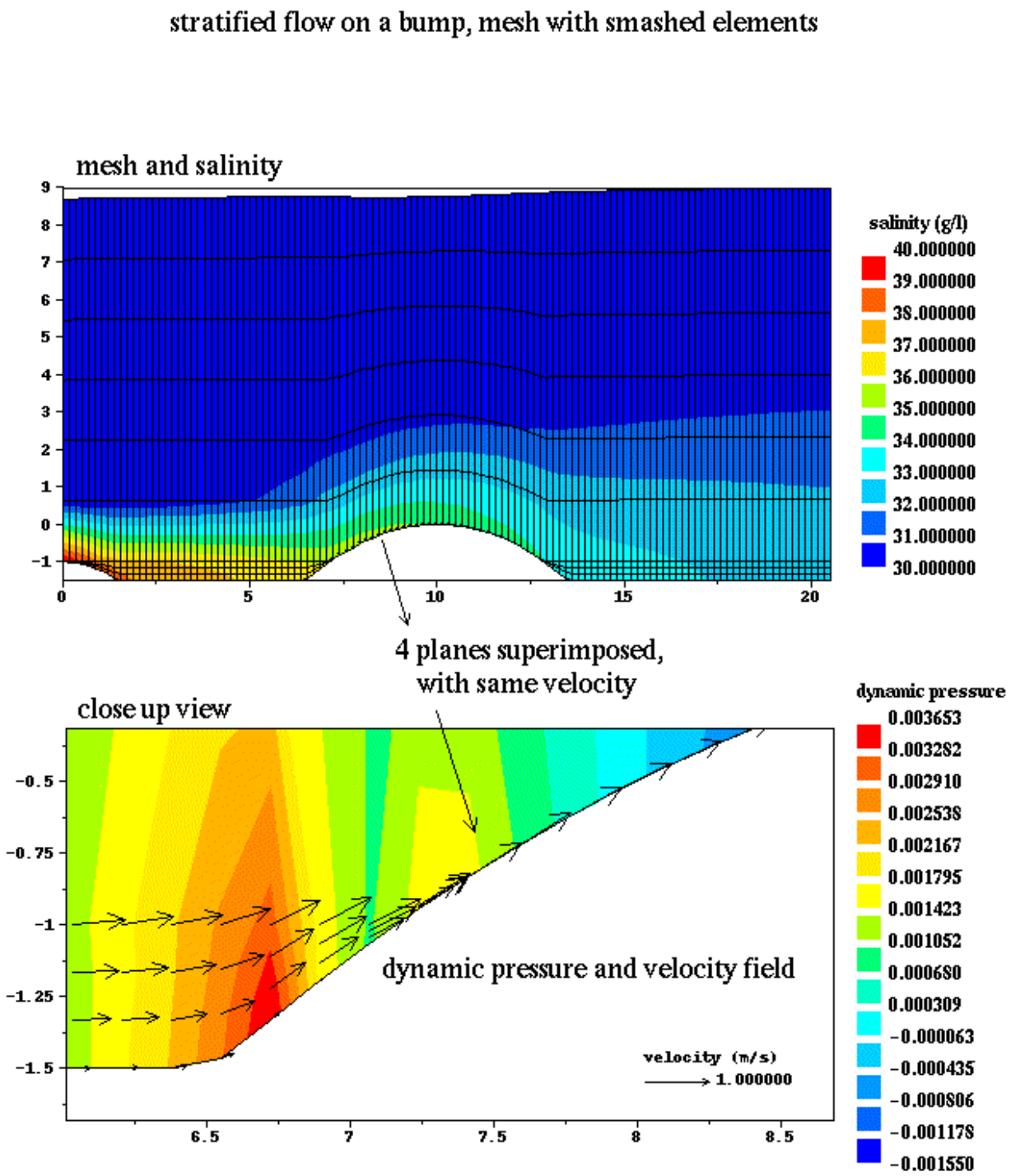

Figure 7.11: Telemac-3D: a mesh with planes superimposed on the bottom

Bibliography

[1] HERVOUET J.-M., Hydrodynamics of free surface flows, modelling with the finite element method. Wiley & sons. 2007

[2] POSTMA L., HERVOUET J.-M., Compatibility between finite volumes and finite elements in solutions of shallow water and Navier-Stokes equations. International Journal for Numerical Methods in Fluids. 53(9), pp.1495-1507. 2007.

[3] HERVOUET J.-M., PHAM C.-T.: Telemac version 5.7, release notes. Telemac-2D and Telemac-3D. 2007

[4] HERVOUET J.-M., RAZAFINDRAKOTO E., VILLARET C.: Telemac version 5.8, release notes. Telemac-2D, Telemac-3D and Sisyphe. 2008

[5] HERVOUET J.-M.: Telemac version 5.9, release notes. Bief, Telemac-2D, Telemac-3D and Sisyphe. 2009

[6] HERVOUET J.-M., MACHET C., VILLARET C.: dealing with rigid beds in saturated bed load transport equations. Proceedings of Coastal Engineering 2003, Cadiz, 23-25 June 2003, Spain. 2003