Proceedings of ALGORITMY 2009
pp. 81–90

WELL-BALANCED FINITE VOLUME EVOLUTION GALERKIN METHODS FOR THE 2D SHALLOW WATER EQUATIONS ON ADAPTIVE GRIDS *

A. BOLLERMANN†, M. LUKÁČOVÁ-MEDVIĎOVÁ‡, AND S. NOELLE†

Abstract. We extend a well-balanced finite volume evolution Galerkin (FVEG) method to non-uniform grids. As a model problem, we consider the two-dimensional shallow water equations with a source term modelling the bottom topography. Our work is based on the well-balanced scheme proposed in (Lukáčová, Noelle, Kraft, *J.Comp.Physics*, 221, 2007). We present selected test cases to demonstrate the capabilities of the scheme.

Key words. Well-balanced schemes; Adaptive grids; Shallow water equations; Evolution Galerkin schemes

AMS subject classifications. 65M50, 35L45, 76B15, 76M12

1. Introduction. We consider the finite volume evolution Galerkin (FVEG) schemes of Lukáčová, Morton and Warnecke, cf. [1, 3, 2]. These methods are used to solve hyperbolic conservation laws by introducing an evolution operator to predict values for the finite volume update. The evolution operators are based on the theory of bicharachteristics, allowing to consider all of the infinitely many directions of wave propagation. These schemes have been proven to be very accurate compared to standard finite volume schemes, and they also allow an efficient implementation, which makes them competitive in terms of computational cost.

By now, FVEG schemes are basically always implemented on uniform, cartesian grids. In contrast, most geophysical phenomena show a very localized behaviour, i.e. we have small regions with strong interactions in a larger surrounding area with almost steady solutions. To resolve the interesting structures correctly, we need a small gridsize, leading to a large number of cells and therefore long computations on uniform grids. To explore further the potential of the FVEG schemes, in this work we want to extend them to non-uniform, adaptive grids, allowing fine resolutions in the area of interest and a minimal cell number in steady regions.

We will apply our schemes to the two-dimensional shallow water equations. The only source term we take into account is the influence of the ground elevation. The presence of a source term gives rise to another difficulty: For steady states, we have to make sure that the numerical flux and the numerical source term cancel each other exactly, so that no spurious oscillations occur. This property is called $well-balancing$. In [2] Lukáčová, Noelle and Kraft proposed a well-balanced variant of FVEG schemes for the two-dimensional shallow water equations. We will start from this method to develop an adaptive scheme maintaining the properties from the uniform case.

This work is organized as follows. In Section 2 we give a brief introduction to the shallow water equations. Section 3 is dedicated to the FVEG scheme derived in

*This work was supported by DFG-Grant "Adaptive semi-implicit FVEG methods for multidimensional systems of hyperbolic balance laws"

†IGPM, RWTH Aachen, Templergraben 55, 52062 Aachen, Germany.

‡Department of Mathematics, University of Technology Hamburg, Schwarzenbergstraße 95, 21073 Hamburg

[2] and our modifications for adaptive grids. Finally, we present selected test cases in Section 4 and summarize our results in the conclusions in Section 5.

2. The shallow water equations.

2.1. Balance law form.

We consider the shallow water system in balance form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u}) = \mathcal{S}(\mathbf{u}, \vec{x}). \tag{2.1}$$

The conserved variables and the flux are given by

$$\mathbf{u} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \qquad \mathcal{F} = [\mathcal{F}_1\, \mathcal{F}_2] = \begin{pmatrix} hu & hv \\ hu^2 + g\frac{h^2}{2} & huv \\ huv & hv^2 + g\frac{h^2}{2} \end{pmatrix}, \tag{2.2}$$

where h denotes the relative water height, $\vec{u} = (u, v)^T$ the flow speed and g the (constant) gravity acceleration. The source term $\mathcal{S}(\mathbf{u}, \vec{x})$ is given by

$$\mathcal{S}(\mathbf{u}, \vec{x}) = -gh \begin{pmatrix} 0 \\ \frac{\partial b(\vec{x})}{\partial x_1} \\ \frac{\partial b(\vec{x})}{\partial x_2} \end{pmatrix} \tag{2.3}$$

with $b(\vec{x})$ the local bottom height. We also introduce the *free surface level*, or total water height, h_{tot} with

$$h_{tot}(\vec{x}) = h(\vec{x}) + b(\vec{x}). \tag{2.4}$$

2.2. Quasi-linear form.

For the derivation of the evolution operators, it is helpful to rewrite (2.1) in primitive variables. The system then takes the form

$$\mathbf{w}_t + A_1(\mathbf{w})\mathbf{w}_{x_1} + A_2(\mathbf{w})\mathbf{w}_{x_2} = \mathbf{t} \tag{2.5}$$

with

$$\mathbf{w} = \begin{pmatrix} h \\ u \\ v \end{pmatrix}, \quad A_1 = \begin{pmatrix} u & h & 0 \\ g & u & 0 \\ 0 & 0 & u \end{pmatrix}, \quad A_2 = \begin{pmatrix} v & 0 & h \\ 0 & v & 0 \\ g & 0 & v \end{pmatrix}, \quad \mathbf{t} = \begin{pmatrix} 0 \\ -gb_{x_1} \\ -gb_{x_2} \end{pmatrix}.$$

We now define for each angle $\theta \in [0, 2\pi)$ the direction $\vec{\xi}(\theta) := (\cos\theta, \sin\theta)$. As system (2.1) is hyperbolic, for each direction $\vec{\xi}(\theta)$ and fixed $\mathbf{w}$ the matrix

$$A(\mathbf{w}) = \vec{\xi}_1 A_1(\mathbf{w}) + \vec{\xi}_2 A_2(\mathbf{w}) \tag{2.6}$$

has real eigenvalues. With $c = \sqrt{gh}$ denoting the speed of sound they read

$$\lambda_1 = \vec{u} \cdot \vec{\xi} - c, \quad \lambda_2 = \vec{u} \cdot \vec{\xi}, \quad \lambda_3 = \vec{u} \cdot \vec{\xi} + c. \tag{2.7}$$

2.3. Lake at rest solution.

One obvious solution of (2.1) is the so-called lake at rest solution, where the water is steady, i.e. $\vec{u} = (0, 0)^T$, and the free surface level is constant, i.e. $h_{tot}(\vec{x}) = h_0$. From (2.4) we immediately get

$$\nabla h = -\nabla b \tag{2.8}$$

and therefore (with (2.1), (2.2), (2.3) and $\vec{u} = (0, 0)^T$)

$$\begin{pmatrix} 0 \\ g\frac{h^2}{2} \\ 0 \end{pmatrix}_{x_1} + \begin{pmatrix} 0 \\ 0 \\ g\frac{h^2}{2} \end{pmatrix}_{x_2} = -gh \begin{pmatrix} 0 \\ \frac{\partial b(\vec{x})}{\partial x_1} \\ \frac{\partial b(\vec{x})}{\partial x_2} \end{pmatrix}. \tag{2.9}$$

A scheme fulfilling a discrete analogon of (2.9) exactly is called *well balanced*.

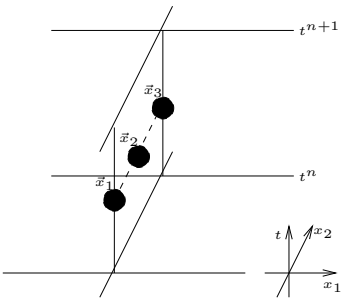

FIG. 3.1. *Quadrature points $\vec{x}_j$ for finite volume update*

3. FVEG schemes on adaptive grids. Many classical finite volume schemes are based on the solution of one-dimensional Riemann-problems at cell interfaces. This leads to a bad resolution of problems where the solution is not aligned with the grid. In contrast, the finite volume evolution Galerkin methods considered in this work are based on a truly multidimensional approach to the problem that is introduced by the evolution operators. The two following sections will describe the finite volume update (Sec. 3.1) and the evolution operator (Sec. 3.2), both on uniform grids. The changes we applied for adaptive grids are presented in Section 3.3.

3.1. Discretization and finite volume update. To obtain approximate solutions of (2.1), we divide our computational domain Ω in quadratic cells C_i with side length or diameter s_i. The cell interfaces, or edges, are denoted by E_i. We denote by $\mathbf{u}_i^n$ the approximate cell-averaged solution on cell C_i at time t^n, where the initial approximation is given by $\mathbf{u}_i^0 := \mathbf{u}_i(0) = \frac{1}{|C_i|}\int_{C_i} \mathbf{u}(\vec{x})\, d\vec{x}$. On each cell, we now integrate (2.1) and apply the Gauss theorem, yielding the update

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \frac{1}{|C_i|}\int_{t^n}^{t^{n+1}} \left(\int_{\partial C_i} \mathcal{F}(\mathbf{u}(\vec{x},t)) \cdot \vec{n}\, d\vec{x}dt + \int_{C_i} \mathcal{S}(\mathbf{u}(\vec{x}))\, d\vec{x} \right) \tag{3.1}$$

where $\vec{n}$ is the outward pointing normal at the boundary.

To eventually perform the update (3.1), we have to give an approximation in time and space for the flux on each edge of the grid. We use the midpoint rule in time, i.e. the flux on each edge will be evaluated at $t^{n+\frac{1}{2}}$. To approximate the flux in space, for each edge we define three quadrature points $\vec{x}_j^E, j = 1,2,3$. $\vec{x}_1^E$ and $\vec{x}_3^E$ are the two endpoints of edge E, and $\vec{x}_2^E$ is the center of the edge. Fig. 3.1 shows the position of the quadrature points. To simplify notation, we will drop the superscript E, and j will always denote a local index with respect to an edge. Using Simpson's rule, we can then define the approximate flux $\mathcal{F}_E$ on each edge via

$$\mathcal{F}_E := \sum_{j=1}^{3} \alpha_j \mathcal{F}(\mathbf{u}_j^{n+1/2}) \cdot \vec{n} \approx \frac{1}{\Delta t |E|}\int_{t^n}^{t^{n+1}} \int_E \mathcal{F}(\mathbf{u}(\vec{x},t)) \cdot \vec{n}\, d\vec{x}dt \tag{3.2}$$

with $\alpha_{1,3} = \frac{1}{6}$ and $\alpha_2 = \frac{2}{3}$ and Δt is the time step. The approximated values $\mathbf{u}_j^{n+1/2} \approx \mathbf{u}(\vec{x}_j, t^{n+1/2})$ will be predicted by the evolution operators described in section 3.2.

For the sourceterm, the gradient of the ground elevation in the i-th cell is approx-

imated by

$$S_i := -g \sum_{j=1}^{3} \alpha_j \begin{pmatrix} 0 \\ \frac{1}{2}(\hat{h}_j^r + \hat{h}_j^l)(b_j^r - b_j^l) \\ \frac{1}{2}(\hat{h}_j^t + \hat{h}_j^b)(b_j^t - b_j^b) \end{pmatrix} \approx \frac{1}{s_i \Delta t} \int_{t^n}^{t^{n+1}} \int_{C_i} \mathcal{S}(\mathbf{u})\, d\vec{x}, \tag{3.3}$$

where $\hat{h}_j$ is the first component of $\mathbf{u}_j^{n+1/2}$, $b_j = b(\vec{x}_j)$ and the superscripts denote the edges surrounding the cell, namely the **r**ight, **l**eft, **t**op and **b**ottom edge. So the fully discrete scheme reads

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{|C_i|} \left(\sum_{E, E \subset \partial C_i} (|E| \mathcal{F}_E) + s_i \mathcal{S}_i \right) \tag{3.4}$$

In [2], it is proven that scheme (3.4) is well-balanced on uniform grids. As every edge in the sum of (3.4) appears in two different cells with opposed normals, it is also conservative by design.

3.2. Evolution operators. We restrict ourselves to the presentation of the main ideas, as the development of the evolution operators is not the focus of this paper. As mentioned before, we need the evolution operator to predict point values of the solution at the quadrature points of our finite volume scheme. We will make use of the fact that the solution can be described by a superposition of different waves. We start by fixing one of the quadrature points $P = (\vec{x}, t^{n+1/2})$. To derive a solution, we start at this point and look back in time where the waves determing the solution come from. To simplify things, we linearize (2.5) by plugging in a suitable mean value $\bar{\mathbf{w}}$ around $P' = (\vec{x}, t)$, leading to

$$\mathbf{w}_t + A_1(\bar{\mathbf{w}})\mathbf{w}_{x_1} + A_2(\bar{\mathbf{w}})\mathbf{w}_{x_2} = \mathbf{t}. \tag{3.5}$$

The crucial point in this procedure is to identify the curves where the waves propagate and to find a representation of the solution along them. We therefore perform a one-dimensional characteristic decomposition of the system (3.5) for each direction $\vec{\xi}(\theta)$. This gives us three directions of wave propagation which correspond to the eigenvalues (2.7). We follow this directions starting from P going backwards in time. The bicharacteristic curves are illustrated on the left side of Fig. 3.2, thanks to the linearization they are straight lines. Along these curves, the characteristic decomposition also delivers a representation of the solution. We can now integrate the solution from t^n till $t^{n+1/2}$ along these curves, giving a representation of the solution at the point P.

To respect the multidimensional behaviour of the solution, we average the point values at P over all angles $\theta \in [0, 2\pi)$. This also leads to a welcome simplification of the representation formula. The combination of all the bicharacteristics results in the so-called *bicharacteristic cone* depicted on the right side of Fig. 3.2. The integral representation of the solution on this cone is the core of all evolution operators used in the FVEG schemes.

The resulting exact integral representation is very complex and not suitable for the use in computations. In previous work, Lukáčová *et.al.* derived several approximations of the evolution operator (see [2] and the references therein). Here we will work with the approximative operators given in equations (3.25) and (3.26) in [2]. These operators allow to compute an approximation of the integrals along the cone based only on the data given at the sonic circle, which is the intersection of the cone with the $\vec{x}$-plane at time t^n.

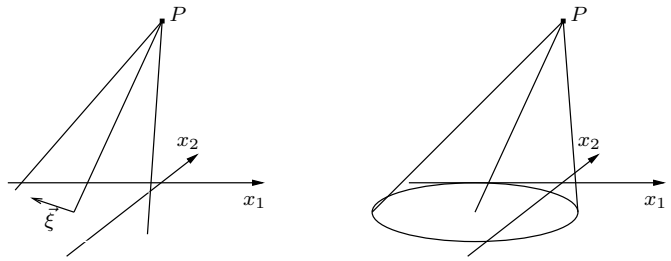

FIG. 3.2. *Bicharacteristic decompostion. Left: Bicharacteristic curves for a fixed direction* $\vec{\xi}$. *Right: Bicharacteristic cone.*

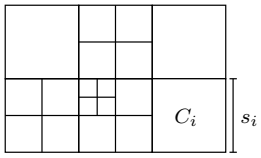

FIG. 3.3. *Example of a structured, non-uniform grid*

3.3. Modifications on adaptive grids. The grids we consider in this work are not uniform, but we limit ourselves to structured, cartesian meshes, see Fig. 3.3 for an example. To distinguish cells of different sizes, we introduce the *level* of refinement l, where a higher level stands for a finer grid. For the transfer of the FVEG schemes to adaptive grids, we have to check how the use of a non-uniform grid changes the evaluation of both the evolution operator and the finite volume update.

Concerning the evolution operator, the transfer is straightforward. To evaluate the operators, one has to identify the cells containing the sonic circle of each quadrature point, and the length of the arc in each cell. Fig. 3.4 shows the most common situations. Only at hanging nodes a new situation occurs, but this is obviously a combination of the known ones and can be easily implemented.

For the finite volume update, Fig. 3.5 depicts the necessary changes in (3.4). First, we perform the flux evaluation (3.2) on all edges of the finest local grid level. On cells at the interface between two grid levels, we then compute the flux across the coarse edges as sum of the fluxes across the finer edges, i.e. in (3.4) we use

$$|E^l|\mathcal{F}_{E^l} = \sum_{E^{l+1}\subset E^l} |E^{l+1}|\mathcal{F}_{E^{l+1}}. \tag{3.6}$$

However, when we introduce a non-flat bottom topography, the scheme is not well-balanced anymore in cells at the interface between two grid levels. The reason is evident from the left picture in Fig. 3.6: While we use five points altogether for the flux update, the source term in the coarse cell uses only three quadrature points at each edge.

We reestablish the well-balancing property by the following modification. In cells with a situation like in the left picture of Fig. 3.6 we temporarily add two additional quadrature points like shown in the right picture of Fig. 3.6 and interpolate h and b linearly from the neighboring points. We then apply (3.3) separately for the upper and lower half of the cell. Provided that the bottom height in the middle quadrature point of each edge is the mean bottom height of its neighboring points, the resulting

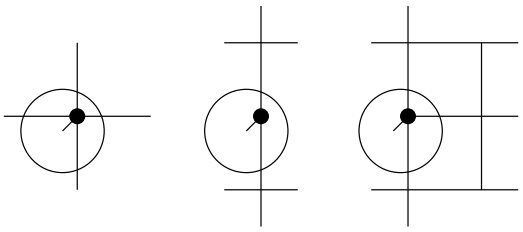

Fig. 3.4. *Typical intersections of the sonic circle with grid cells. Left: Corner node. Center: Node at interface between two cells. Right: Hanging node.*

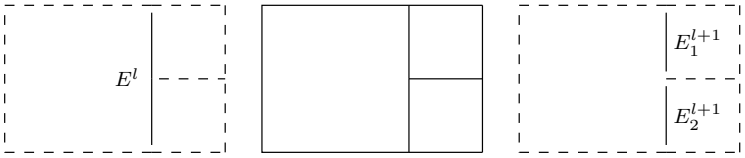

Fig. 3.5. *Interface between two refinement levels l and $l+1$. Left: Single edge for left cell. Right: Two edges for right cells.*

scheme is again well-balanced for the lake at rest situation. The last assumption is easy to fulfill as the evolution operators in (3.25) and (3.26) from [2] give no rigorous restriction to the choice of $b(P)$.

Finally, to preserve the well-balanced behavior even when adapting the grid, we have to pay some attention to the mapping of values between two grids. First, the mapping has to be conservative. As a second point, we have to ensure that a constant free surface is kept between two grids. In the case of a refined cell, i.e. cell C_0 on the old grid is replaced by four child cells $\bar{C}_1, \ldots, \bar{C}_4$ on the new grid, we set

$$h(\bar{C}_k) = h_{tot}(C_0) - b(\bar{C}_k), \quad k = 1, \ldots, 4. \tag{3.7}$$

This definition demands

$$b(C) = \frac{1}{4} \sum_{k=1}^{4} b(\bar{C}_k), \tag{3.8}$$

so that we use this as a recursive definition for the bottom height on each cell. The recursion stops when the level of refined cells reaches the maximal grid level, where we set $b(C) = b(\vec{x}_C)$, with $\vec{x}_C$ the center of the cell. For the other variables we set

$$hu(\bar{C}_k) = h(\bar{C}_k)u(C_0), \quad k = 1, \ldots, 4, \tag{3.9}$$

$$hv(\bar{C}_k) = h(\bar{C}_k)v(C_0), \quad k = 1, \ldots, 4, \tag{3.10}$$

which keeps the velocity constant on a refined cell. In the case of a coarsened cell, we just average the values on the old grid to get the values on the new grid. That is consistent with (3.7) – (3.10).

4. Numerical results. We will perform two test cases to demonstrate the behaviour of our adaptive schemes and compare them to the uniform case. The actual scheme used in the computation is the second order scheme from [2] called EG5. We aim to investigate the general properties of our schemes here and will concentrate

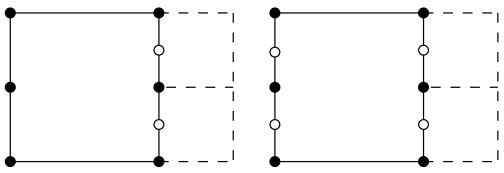

FIG. 3.6. *Quadrature points on interface between two refinement levels l and $l+1$. Left: Solid points used at coarse and fine level, blank points only used at fine level. Right: Modification for well-balanced source term computation*

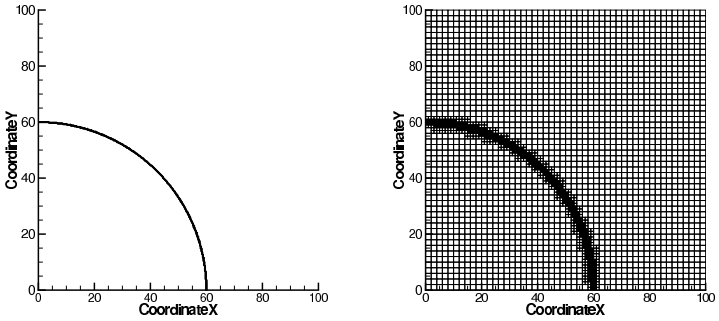

FIG. 4.1. *Circular dam break. Left: Water height contour lines of initial solution. Right: Initial grid for the adaptive scheme.*

on efficiency in future work. For the adaptation, we chose a simple, residual based strategy. For each cell C, we define the local cell residual R as

$$R(C) = \left\| \int_{\partial C_i} \mathcal{F}(\mathbf{u}) \cdot \vec{n}\, dx + \int_{C_i} \mathcal{S}(\mathbf{u})\, dx \right\|_1 .$$

The average residual $\bar{R}$ is given as

$$\bar{R} = \frac{1}{n} \sum_{i=1}^{n} R(C_i)$$

with n the number of cells in the grid. We then refine a cell if $R(C) > 1.5\bar{R}$ and coarsen the cell if $R(C) < \frac{1}{5}\bar{R}$. Our numerical experiments show that the accuracy of the results strongly depends on these choices. The investigation of better error indicators is part of our ongoing work.

4.1. 2D circular dam break. To compare the behaviour of the uniform and adaptive methods, we consider a two-dimensional dam break problem. The circular dam separates two basins with water heights $h_1 = 10$ and $h_2 = 5$, see Fig. 4.1. The computational domain is $\Omega = [0, 100]^2$, the radius of the dam is 60. We assume that at $t = 0$ the whole dam breaks down and compute the flow of water until $t = 3$.

Tables 4.1 and 4.2 show CPU times and errors on different uniform and adaptive grids. The initial resolution of the adaptive grids is 2500 cells, in the computations we allow up to three levels of refinement. The results are visualized in Fig. 4.2. On the left side, we show the water level of the two solutions with finest grids. The

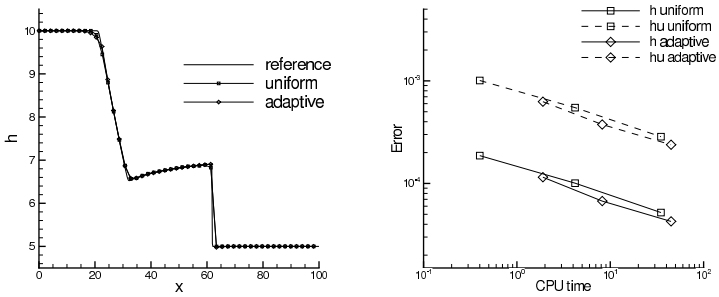

FIG. 4.2. *Solution of the dam break problem. Left: Water height level at line $x_1 = x_2$. Right: Error vs. CPU time*

# cells	CPU time	$\|\tilde{h} - h\|_{L^1}$	$\|\tilde{hu} - hu\|_{L^1}$
2500	0.4s	1.86665E-04	1.00903E-03
10000	4.2s	1.00602E-04	5.48497E-04
40000	34.9s	5.20116E-05	2.85568E-04

TABLE 4.1
CPU times and error on uniform grids

two solutions are very close to each other and approximate very well the reference solution. Looking at the relation between accuracy and computational cost, the right side of Fig. 4.2 shows that the adaptive scheme has a slight advantage over the uniform method.

4.2. Two-dimensional quasi-stationary problem. We want to demonstrate the well-balancing property by a classical test case where we compute a small perturbation of the lake at rest over a smooth bottom topography. The computational domain is $\Omega = [0, 2] \times [0, 1]$ and we apply periodic boundary conditions in the x_2-direction and far field boundary conditions in the x_1-direction. The bottom topography is given by

$$b(x_1, x_2) = 0.8e^{-5(x_1-0.9)^2-50(y-0.5)^2}$$

and the initial values are given by

$$h_0(\vec{x}) = \begin{cases} 1.01 - b(\vec{x}) & \text{if } 0.05 < x_1 < 0.15 \\ 1.0 - b(\vec{x}) & \text{otherwise} \end{cases}$$
$$\vec{u}_0(\vec{x}) = (0, 0)^T.$$

We performed this test case on a uniform grid with 400×200 cells and on an adaptive grid with an initial resolution of 50×25 cells and up to three levels of refinement. In Fig. 4.3 we present contour lines of both solutions at times $t = 0.12$, $t = 0.24$, $t = 0.36$ and $t = 0.48$. The initial adaptive grid has 3725 cells, the number of cells for the presented times are 5585, 6080, 6806 and 9863. We clearly see that both solutions are very close to each other. The results compare very well to those published in literature, see e.g. [4, 5] and the references therein.

# cells start	# cells end	l_{max}	CPU time	$\|\tilde{h} - h\|_{L^1}$	$\|\tilde{hu} - hu\|_{L^1}$
2818	3865	1	1.9s	1.15072E-04	6.26842E-04
3463	7384	2	8.2s	6.73611E-05	3.75620E-04
4951	18346	3	44.8s	4.26710E-05	2.38546E-04

TABLE 4.2
CPU times and error on adaptive grids with different levels of refinement

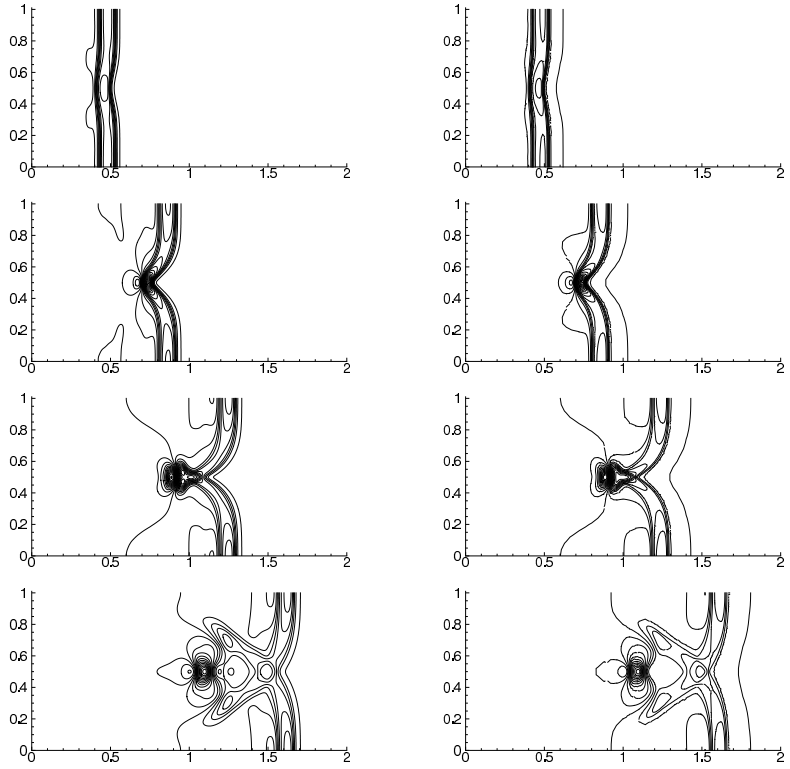

FIG. 4.3. *Two-dimensional quasi-stationary problem, 30 contour lines of free surface between 0.992 and 1.0115. From top to bottom:* $t = 0.12$, $t = 0.24$, $t = 0.36$, $t = 0.48$. *Left: Uniform solution. Right: Adaptive solution.*

In Fig. 4.4 the solution is displayed along the line $x_2 = 0.5$. The solution is perfectly flat in regions not yet reached by the wave, so we have a numerical verification of the well-balancing. However, we can see that the solution on the adaptive grid has lost some features of the uniform solution, see e.g. the region around $x_1 = 1.2$ at $t = 0.48$. This coincides with a relatively coarse mesh resolution in this region (not shown), so that these problems can probably be cured by a more sophisticated refinement strategy. Nevertheless, the results on the adaptive grid are very promising, taking into account that we compute with much fewer cells ($< 15\%$).

5. Conclusions. We have shown that the FVEG schemes from [2] can be used on adaptive grids without losing any of the positive properties, especially the accuracy

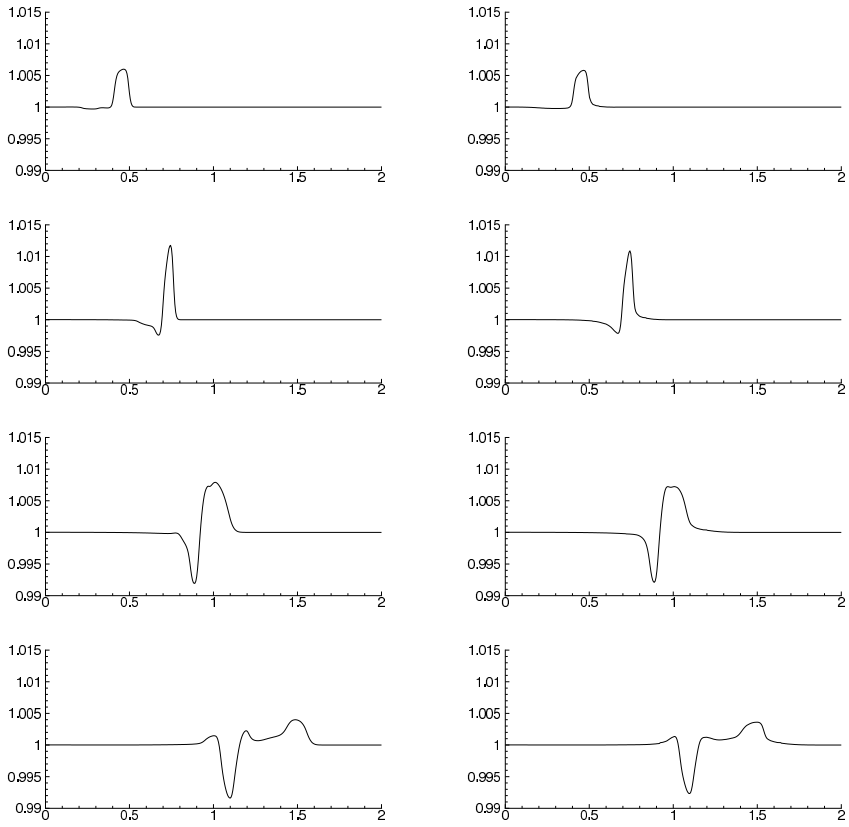

Fig. 4.4. *Two-dimensional quasi-stationary problem, free surface over x_1 at $x_2 = 0.5$. From top to bottom: $t = 0.12$, $t = 0.24$, $t = 0.36$, $t = 0.48$. Left: uniform solution. Right: adaptive solution.*

and the well-balancing. As this paper is the first step towards FVEG schemes on adaptive grids, the gain in CPU time is relatively small. We expect this to significantly improve by the use of more sophisticated error indicators in the refinement strategy.

REFERENCES

[1] Lukáčová-Medviďová M., Morton K.W., Warnecke G., *Finite volume evolution Galerkin (FVEG) methods for hyperbolic systems*, SIAM J. Sci. Comp. 26(1) (2004), pp 1–30.

[2] Lukáčová-Medviďová M., Noelle S., Kraft M., *Well-balanced Finite Volume Evolution Galerkin Methods for the Shallow Water Equations*, J. Comp. Phys. 221 (2007), pp 122–147.

[3] Lukáčová-Medviďová, M., Saibertová, J., Warnecke, G., *Finite volume evolution Galerkin methods for nonlinear hyperbolic systems*, J. Comp. Phys. 183 (2002), pp 533–562.

[4] Noelle, S., Pankratz, N., Puppo, G., Natvig, J. R. *Well-balanced finite volume schemes of arbitrary order of accuracy for shallow water flows* J. Comp. Phys. 213 (2006), pp 474–499

[5] Ricchiuto, M. and Bollermann, A. *Stabilized residual distribution for shallow water simulations*, J. Comp. Phys. 228 (2009), pp 1071–1115