Well-balanced reconstruction operator for systems of balance laws: numerical implementation

I. Gómez-Bueno, M.J. Castro, C. Parés

Abstract In some previous works, two of the authors introduced a strategy to develop high-order well-balanced numerical methods for 1d systems of balance laws. There, a strategy which allows us to modify any standard reconstruction operator in order to be well-balanced was also described. This strategy involves a nonlinear problem at every cell, at every time step, that consists in finding the stationary solution whose average is the given cell value. Our goal is to present a general efficient implementation that can be applied to any system of balance laws by interpreting these nonlinear problems as control problems that are rewritten in functional form. Newton's and descent methods are applied and compared. Applications to the Burgers' equation with a nonlinear source term and to the 1d shallow water model are finally shown.

1 Introduction

Let us consider a PDE system of the form:

$$U_t(x,t) + f(U(x,t))_x = S(U(x,t))H_x(x), \quad x \in \mathbb{R},\ t > 0, \tag{1}$$

where $U(x,t)$ takes values on an open convex set $\Omega \subset \mathbb{R}^N$, $f : \Omega \longrightarrow \mathbb{R}^N$ is the flux function, $S : \Omega \longrightarrow \mathbb{R}^N$, and $H : \mathbb{R} \longrightarrow \mathbb{R}$ is a continuous known function (possibly the identity function). It is supposed that system (1) is strictly hyperbolic, that is, $D_f(U) = \dfrac{\partial f}{\partial U}(U)$ has N real different eigenvalues and eigenvectors.

I. Gómez-Bueno
University of Málaga. e-mail: igomezbueno@uma.es

M.J. Castro
University of Málaga. e-mail: mjcastro@uma.es

C. Parés
University of Málaga. e-mail: pares@uma.es

Systems of the form (1) have non trivial stationary solutions that satisfy the ODE system:

$$f(U)_x = S(U)H_x. \tag{2}$$

A numerical method is said to be well-balanced if it solves exactly or with enhanced accuracy all the stationary solutions of the system or, at least, a relevant family of them. The use of methods with this property is of major importance when the waves generated for small perturbations of a steady state are going to be simulated: this is the case, for instance, for tsunami waves in the Ocean. Well-balanced methods have been studied by many authors: see [2] and its references for a recent review on this topic.

Recently, in [5] the following family of semidiscrete high-order well-balanced finite-volume methods for (1) has been discussed:

$$\frac{dU_i}{dt} = -\frac{1}{\Delta x}\left(F_{i+\frac{1}{2}}(t) - F_{i-\frac{1}{2}}(t)\right) + \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} S(P_i^t(x))H_x(x)\,dx, \tag{3}$$

where:

- $I_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$ are the computational cells, whose length Δx is supposed to be constant for simplicity;
- $U_i(t)$ is the approximation of the average of the exact solution at the ith cell at time t, that is,
$$U_i(t) \cong \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U(x,t)\,dx;$$
- $P_i^t(x)$ is the approximation of the solution at the ith cell given by a high-order reconstruction operator from the sequence of cell averages $\{U_i(t)\}$:
$$P_i^t(x) = P_i(x; \{U_j(t)\}_{j\in\mathcal{S}_i});$$
where $\mathcal{S}_i$ denotes the set of indexes of the cells belonging to the stencil of the ith cell.
- $F_{i+\frac{1}{2}} = \mathbb{F}(U^{t,-}_{i+\frac{1}{2}}, U^{t,+}_{i+\frac{1}{2}})$, where $U^{t,\pm}_{i+\frac{1}{2}}$ are the reconstructed states at the intercells, i.e.
$$U^{t,-}_{i+\frac{1}{2}} = P_i^t(x_{i+\frac{1}{2}}), \quad U^{t,+}_{i+\frac{1}{2}} = P_{i+1}^t(x_{i+\frac{1}{2}}),$$
and $\mathbb{F}$ is a consistent first order numerical flux.

It can be then easily shown that, if the reconstruction operator is well-balanced for a stationary solution U of (1) then the numerical method is also well-balanced for U according to the following definitions:

Definition Given a stationary solution U of (1):

- The numerical method (3) is said to be well-balanced for U if the vector of cell averages of U is an equilibrium of the ODE system (3).
- The reconstruction operator is said to be well-balanced for U if

$$P_i(x) = U(x), \quad \forall x \in [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}], \forall i, \tag{4}$$

where P_i is the approximation of U obtained by applying the reconstruction operator to the vector of cell averages of U. □

The following strategy to design a well-balanced reconstruction operator P_i on the basis of a standard operator Q_i was introduced in [1]: given a family of cell values $\{U_i\}$, at every cell $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$:

1. Look for the stationary solution $U_i^*(x)$ such that:

$$\frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U_i^*(x)\, dx = U_i. \tag{5}$$

2. Apply the reconstruction operator to the cell values $\{\widetilde{U}_j\}_{j \in S_i}$ given by

$$\widetilde{U}_j = U_j - \frac{1}{\Delta x} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} U_i^*(x)\, dx,$$

to obtain:

$$Q_i(x) = Q_i(x; \{\widetilde{U}_j\}_{j \in S_i}).$$

3. Define

$$P_i(x) = U_i^*(x) + Q_i(x). \tag{6}$$

It can be then easily shown that the reconstruction operator P_i in (6) is well-balanced for every stationary solution provided that the reconstruction operator Q_i is exact for the null function. Moreover, P_i is conservative, i.e.

$$\frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} P_i(x)\, dx = U_i, \quad \forall i, \tag{7}$$

provided that Q_i is conservative, and it is also high-order accurate provided that the stationary solutions are smooth.

The main difficulty when this strategy is applied comes from the first step of the well-balanced reconstruction operator: a nonlinear problem of the form (5) has to be solved at every time step. Since the stationary solutions of (1) are the solutions of the ODE system (2), problem (5) is equivalent to find the solution of an ODE system with prescribed average in an interval. In some cases, the explicit form of the general solution of the ODE is known and (5) can be solved by hand or by using standard iterative methods for nonlinear problems: see [5] for examples.

Our goal is to describe a general methodology to solve numerically problems of the form (5) and to apply it to the implementation of well-balanced reconstruction operators for general systems of balance laws whether or not the analytical expression of the stationary solutions is known.

The organization of this work is as follows: in Section 2 problem (5) is interpreted as a control problem. It is first written in functional form; then, the gradient of the

functional is computed using the adjoint equation. Newton's and descent methods can be applied to solve numerically the problem: this is done in Section 3. A comparison between both methods is shown in Section 4, where different strategies for the choice of the descent step have been also tried. In practice, the state and the adjoint equations, the cell-averages, and the integral appearing at the source terms are computed numerically: two numerical tests are shown in Section 5 in order to check the accuracy and the well-balancedness of the methods. A scalar balance law and the shallow water model are considered. Finally, some conclusions are drawn.

2 Control Problem

In the first stage of the well-balanced reconstruction procedure one has to find a solution of the ODE problem

$$f(U)_x = S(U)H_x, \tag{8}$$

such that its average in the cell $[x_{i-1/2}, x_{i+1/2}]$ is the given cell value U_i. For stationary solutions such that $D_f(U(x))$ is regular for every x (i.e. no sonic state is reached), solving (8) is equivalent to solve the ODE system in normal form:

$$U_x = G(x, U), \tag{9}$$

where G is the function $G : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}^N$ defined by

$$G(U, x) = D_f(U)^{-1} S(U) H_x. \tag{10}$$

In this article we focus on the preservation of stationary solutions satisfying (9): the methods introduced here preserve supersonic or subsonic stationary solutions, but no transcritical ones. As it will be seen, in the algorithm developed here to compute the first stage of the well-balanced operator, Cauchy problems associated to the ODE system (9) will be numerically solved, what can be done with any standard ODE solver. To adapt the algorithm to the preservation of transcritical stationary solutions, Cauchy problems associated to (9) should be numerically computed, what can be difficult when the initial condition is a sonic state or if it is close to it. In this case the system may have no solution or to have more than one. The strategy followed here to deal with this difficulty consists on applying the standard reconstruction procedure whenever a sonic state is detected in the stencil. This simple modification allows us to overcome this difficulty and the algorithm is only modified in the stencils where a sonic point is detected. In Section 5.2.2 it will be seen that, in the case of the shallow water equations, this strategy allows one to correctly handle with transonic regimes and even to accurately preserve transonic stationary solutions. Nevertheless, to extend the algorithm so that these solutions are genuinely preserved is a challenging problem that is out of the scope of this work and will be faced in a forthcoming work.

Notice that, even if only stationary solutions satisfying (9) are sought, the problem consisting in finding a solution of this ODE system with given average may have no solution. Observe that if (5) has no solution at the ith cell then U_i cannot be the average of any stationary solution. Therefore, at this cell the standard reconstruction operator is applied, i.e. $U_i^* \equiv 0$ is chosen in the first step.

Let us assume thus that no sonic points have been detected in the stencil $\mathcal{S}_i$. Then, the nonlinear problem (5) to be solved at every cell can be then formulated as follows:

Find $U_{i-1/2} \in \Omega$ such that

$$\mathcal{F}(U_{i-1/2}) = U_i, \tag{11}$$

where $\mathcal{F} : \Omega \mapsto \mathbb{R}^N$ is given by

$$\mathcal{F}(U_0) = \frac{1}{\Delta x} \int_0^{\Delta x} V_i(x, U_0)\, dx, \tag{12}$$

where $V_i(x, U_0)$ denotes the solution of the Cauchy problem

$$\begin{cases} V_x = G(V, x + x_{i-1/2}), \\ V(0) = U_0. \end{cases} \tag{13}$$

Notice that Cauchy problem (13) is equivalent to solve (9) with initial condition

$$U(x_{i-1/2}) = U_{i-1/2}.$$

Once (11) has been solved, the sought stationary solution is

$$U_i^*(x) = V(x - x_{i-1/2}, U_{i-1/2}).$$

This problem can be interpreted as a control one, and the adjoint technique can be used to compute the gradient of $\mathcal{F}$, what gives:

$$D\mathcal{F}(U_0) = \frac{1}{\Delta x} \Lambda(0)^T,$$

where Λ denotes the matrix whose columns are the so-called adjoint variables $\boldsymbol{\lambda}_1(x), \ldots, \boldsymbol{\lambda}_N(x)$ that solve the following Cauchy problems:

$$\begin{cases} \dfrac{d\boldsymbol{\lambda}_j}{dx}(x) = -\mathbf{e}_j - \nabla_U G(U, x + x_{i+1/2})^T \cdot \boldsymbol{\lambda}_j, \\ \\ \boldsymbol{\lambda}_j(\Delta x) = 0, \end{cases} \tag{14}$$

where we denote by $\mathbf{e}_j$ the jth vector of the canonical basis and

$$\nabla_U G(U,x) = \begin{bmatrix} \frac{\partial G_1}{\partial u_1}(U,x) & \dots & \frac{\partial G_1}{\partial u_N}(U,x) \\ \vdots & \ddots & \vdots \\ \frac{\partial G_N}{\partial u_1}(U,x) & \dots & \frac{\partial G_N}{\partial u_N}(U,x) \end{bmatrix}; \tag{15}$$

(see [6] for details).

First and second order methods can be implemented in an easier way if the mid-point rule is used to approach the cell averages:

$$\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} U(x)\,dx \cong U(x_i).$$

In effect, in this case the first step in the reconstruction procedure reduces to look for the stationary solution U_i^* such that:

$$U_i^*(x_i) = U_i. \tag{16}$$

There is no need thus to solve nonlinear problems of the form (11): it is enough to solve standard Cauchy problems to compute U_i^*. Therefore, in what follows we focus on methods of order greater than two. In particular, the third order CWENO reconstruction (see [9], [3]) will be considered in the numerical experiments. The state and the adjoint differential equations will be numerically computed using the standard RK4 method.

3 Numerical Algorithms

Let us consider for simplicity that $x_{i-1/2} = 0$ and denote by $W \in \Omega$ the cell value, so that the problem to solve is:

Find $U_0 \in \Omega$ such that

$$\mathcal{F}(U_0) = W. \tag{17}$$

Since problems of this type have to be solved at every intercell at every time step, it is crucial to choose an efficient numerical method. Two different strategies are considered here:

3.1 Newton's Method

A sensible choice for the initial guess U_0^0 is W: if Δx is small, the average of the solution of the Cauchy problem is expected to be close to the initial condition. The algorithm is then as follows:

Algorithm Newton's method

- $U_0^0 = W$;
- For k = 0,1,2. . .
  - Compute the solution U_k of (13) with initial condition U_0^k in the interval $[0, \Delta x]$.
  - For $j = 1, \ldots, N$ compute the solution $\boldsymbol{\lambda}_j$ of (14) with $U = U_k$ in the interval $[0, \Delta x]$.
  - Compute V_k by solving the linear system:

$$\Lambda(0)^T V_k = \Delta x(\mathcal{F}(U_0^k) - W).$$

  - Update U_0^k:

$$U_0^{k+1} = U_0^k - V_k.$$

At every iteration of the method $N+1$ Cauchy problems and a $N \times N$ linear system have to be solved. The computational cost can be reduced by using the modified Newton's method in which the matrix $\Lambda(0)$ is only updated every K iterations, where K is a fixed integer.

3.2 Descent Methods

An alternative approach to solve (5) consists in solving the minimization problem:

$$\min_{U_0 \in \mathbb{R}^N} J(U_0) \tag{18}$$

with:

$$J(U_0) = \|\mathcal{F}(U_0) - W\|^2 .$$

If the euclidean norm is chosen, a simple computation shows that:

$$\nabla J(U_0) = \frac{2}{\Delta x} \Lambda(0)^T \cdot \left(\frac{1}{\Delta x} \int_0^{\Delta x} (U(x, U_0) - W)\, dx \right). \tag{19}$$

Gradient method write thus as follows:

Algorithm Gradient method

- $U_0^0 = W$.
- For $k = 0, 1, \ldots$:
  - Compute the solution U_k of (13) with initial condition U_0^k in the interval $[0, \Delta x]$.
  - For $j = 1, \ldots, N$ compute the solution $\boldsymbol{\lambda}_j$ of (14) with $U = U_k$ in the interval $[0, \Delta x]$.
  - Compute

$$\nabla J(U_k) = \frac{1}{\Delta x} 2\,\Lambda(0)^T \cdot \left(\frac{1}{\Delta x} \int_0^{\Delta x} (U_k - W)\, dx \right).$$

– Update U_0^k:

$$U_0^{k+1} = U_0^k - \rho_k \nabla J(U_k),$$

where ρ_k is the step.

Conjugate gradient methods are also considered with the descent directions:

$$d_k = \begin{cases} \nabla J(U_k) & if\ k = 0, \\ \nabla J(U_k) + \dfrac{\nabla J(U_k) \cdot (\nabla J(U_k) - \nabla J(U_{k-1}))}{\|\nabla J(U_{k-1})\|^2}\, d_{k-1} & if\ k \geq 1. \end{cases}$$

At every iteration of the gradient or the conjugate gradient methods, at least $N+1$ Cauchy problems have to be solved, but a new Cauchy problem has to be solved in every evaluation of the cost function in the search of the step ρ_k.

Search for the Optimal Step

Once the descent direction has been computed, the step ρ_k has to be chosen. Five different options of stepsize selection are discussed.

β Stepsize Method. First version.

At the kth iteration, when steps 1 and 2 of the descent algorithm have already been computed, use the following strategy to select the stepsize:

***Algorithm* β stepsize method. First version.**

- Set $\rho_k^0 = \rho_{k-1}$. (If $k = 1$, ρ_{-1} is arbitrarily chosen).
- Set $V_k^0 = J(U_{k-1})$
- Set $j = 1$:
  - Set $V_k^j = J(U_{k-1} - \rho_k^{j-1} d_{k-1})$.
  - If $V_k^j < V_k^{j-1}$, choose $\rho_k^j = \beta \rho_k^{j-1}$. In other case, choose $\rho_k^j = \dfrac{\rho_k^{j-1}}{\beta}$.
  - If $j \geq 2$ y $\rho_k^j = \rho_k^{j-2}$, stop.
- Set $\rho_k = \rho_k^j$.
- $j = j + 1$. □

Here β is a parameter to be chosen in the interval $(0, 1)$.

β Stepsize Method. Second version.

At the kth iteration, when steps 1 and 2 of the descent algorithm have already been computed, use the following strategy to select the stepsize:

***Algorithm* β stepsize method. Second version.**

- Set $V^d_{0,k} = J(U_{k-1})$, $d_{k-1} = \nabla J(U_{k-1})$.
- $\rho^d_{0,k} = \beta\rho_{k-1}$. (If $k = 1$, ρ_{-1} is arbitrarily chosen).
- Set $V^d_{1,k} = J(U_{k-1} - \rho^d_{0,k} d_{k-1})$.
- Set $j = 1$.
- While $V^d_{j,k} < V^d_{j-1,k}$ do:
  - Set $\rho^d_{j,k} = \beta\, \rho^d_{j-1,k}$ and $V^d_{j+1,k} = J(U_{k-1} - \rho^d_{j,k} d_{k-1})$.
  - $j = j + 1$.
- Set $V^h_{0,k} = J(U_{k-1})$.
- $\rho^h_{0,k} = \dfrac{1}{\beta}\rho_{k-1}$.
- Set $V^h_{1,k} = J(U_{k-1} - \rho^h_{0,k} d_{k-1})$.
- Set $j = 1$.
- While $V^h_{j,k} < V^h_{j-1,k}$ do:
  - Set $\rho^h_{j,k} = \dfrac{1}{\beta}\, \rho^h_{j-1,k}$ and $V^h_{j+1,k} = J(U_{k-1} - \rho^h_{j,k} d_{k-1})$.
  - $j = j + 1$.
- If $V^d_{j,k} < V^h_{j,k}$, set $\rho_k = \rho^d_{j,k}$. Else, $\rho_k = \rho^h_{j,k}$. □

β is again a parameter to be chosen in the interval $(0, 1)$.

Armijo Rule.

The following algorithm is based on the Armijo Rule. It requires two parameters: $\mu \in [0.01, 0.3]$ and $\beta \in [0.1, 0.8]$. Let us suppose that $\bar{U}$ is the approximation of the solution at the previous iteration and $\bar{d}$ is the descent direction. Let us consider the function $\mathbf{J}(\rho) = J(\bar{U} - \rho\bar{d})$. Then the first order approximation of $\mathbf{J}(\rho)$ at $\rho = 0$ is given by $\mathbf{J}(0) - \rho\,\mathbf{J}'(0)$. Define $\hat{\mathbf{J}}(\rho) = \mathbf{J}(0) - \mu\,\rho\,\mathbf{J}'(0)$. A stepsize $\bar{\rho}$ is considered acceptable by the Armijo Rule if $\mathbf{J}(\bar{\rho}) \le \hat{\mathbf{J}}(\bar{\rho})$.

When the steps 1, 2 and 3 of the general algorithm described above have been computed at the kth iteration, define the functions:

$$\mathbf{J}(\rho_k) = J(U_k - \rho_k d_k), \quad \hat{\mathbf{J}}(\rho_k) = \mathbf{J}(0) - \mu\,\rho_k \mathbf{J}'(0) = J(U_k) - \mu\,\rho_k d_k \cdot d_k.$$

The stepsize ρ_{k+1} is chosen using the next rule: if $\mathbf{J}(\rho_k) > \hat{\mathbf{J}}(\rho_k)$, then $\rho_{k+1} = \beta\rho_k$, and otherwise take $\rho_{k+1} = \rho_k$.

This algorithm can be improved by applying the Armijo Rule as many times as possible at every iteration, provided that the objective function decreases as in the second version of the β stepsize method.

Wolfe Conditions

The choice of the step is based on Wolfe conditions: see [8]. It requires two parameters: $m_1 \in [0.01, 0.3]$ and $m_2 \in [0.5, 1]$. At the kth iteration, once steps 1, 2 and 3 of the general algorithm have been computed, follow the following algorithm:

Algorithm **Wolfe conditions.**

a. Set $\rho_s^k = 0$ and $\rho_b^k = 0$.
b. Compute the functions:

$$\mathbf{J}(\rho_k) = J(U_k - \rho_k d_k),$$

$$\hat{\mathbf{J}}(\rho_k) = J(U_k) - \mu\, \rho_k\ (d_k \cdot d_k)\,.$$

- If $\mathbf{J}(\rho_k) \leq \hat{\mathbf{J}}(\rho_k)$ and $\mathbf{J}'(\rho_k) \geq m_2\, \mathbf{J}'(0)$, take $\rho_{k+1} = \rho_k$ and stop the algorithm.
- If $\mathbf{J}(\rho_k) > \hat{\mathbf{J}}(\rho_k)$, set $\rho_b^k = \rho_k$ and go to the step **c**.
- If $\mathbf{J}(\rho_k) \leq \hat{\mathbf{J}}(\rho_k)$ and $\mathbf{J}'(\rho_k) < m_2\, \mathbf{J}'(0)$, set $\rho_s^k = \rho_k$ and go to **c**.

c. Use the following rule to choose ρ_{k+1} :

- If $\rho_b^k = 0$, take $\rho_{k+1} = a\rho_k$, where $a > 1$.
- Else, take $\rho_{k+1} = \dfrac{\rho_s^k + \rho_b^k}{2}$. □

Fixed Stepsize

The first step ρ_0 is computed using any of the previous algorithms and then

$$\rho_k = \rho_0, \quad \forall k.$$

3.3 Numerical Integration

In practice, a quadrature rule in $[0, \Delta x]$ is used to compute the averages appearing in the definition of $\mathcal{F}$ (12), in the expression of the gradients (19), or in the computation of W (if it is the average of a known function):

$$\int_0^{\Delta x} g(x)\, dx \cong \Delta x \sum_{l=0}^{M} \alpha_l g(x_l),$$

and the Cauchy problems to compute U_k and λ_j are solved with a numerical method for ODE using a mesh of the interval $[0, \Delta x]$ whose maximum step will be denoted by h. This mesh will be chosen so that all the quadrature points x_l are nodes. The order of the method and the size of h will be chosen so that errors are close to machine precision.

Therefore, in practice the following discrete problems have to be solved :

Find U_0 such that

$$\mathcal{F}_h(U_0) := \sum_{l=0}^{M} \alpha_l U_{h,l} = \widetilde{W},$$

where $U_{h,l}$ represents the numerical approximation of $U(x, U_0)$ at the quadrature point x_l given by the numerical method chosen to solve the ODE and $\widetilde{W}$ the approximation of W obtained with que quadrature formula.

4 A Numerical Test for the Control Problem

The Newton's and descent methods with different strategies for the choice of the descent steps have been tested and compared in order to check their efficiencies.

We will discuss the following scalar problem: find $u_0 \in \mathbb{R}$ such that

$$\mathcal{F}_{\Delta x}(u_0) = w, \tag{20}$$

where

$$\mathcal{F}_{\Delta x}(u_0) = \frac{1}{\Delta x} \int_0^{\Delta x} u(x, u_0)\, dx,$$

$w = 1$, and $u(x, u_0)$ is the solution of:

$$\begin{cases} u_x = \dfrac{\sin(u)}{u}, \\ u(0) = u_0. \end{cases} \tag{21}$$

The two-points Gauss quadrature rule is considered to approximate the integrals and the fourth order Runge-Kutta method is applied to solve the Cauchy problems using the mesh consisting of the extremes of the interval $[0, \Delta x]$ and the two quadrature points.

In order to check the efficiency of the Newton's and descent methods we compare the number of iterations and the CPU times required for solving problem (21) a big number of times (say 10000 times) with decreasing values of Δx and different error tolerances ε (see Tables 1 and 2). The values of Δx considered are in the range of those used in applications to the numerical solution of systems of balance laws: remember that problems (11) are solved at every computational cell. For these small values of Δx, the number of iterations of the gradient methods is independent in this case of the strategy followed to select the stepsizes.

	Tolerance $\varepsilon = 10^{-8}$		Tolerance $\varepsilon = 10^{-12}$	
Δx	Newton's method	Gradient methods	Newton's method	Gradient methods
1	2	6	3	10
0.5	2	4	2	7
0.1	1	2	1	4

Table 1: Number of iterations for different intervals $[0, \Delta x]$

Δx	Newton's method	Gradient method				
		$\beta = 0.5$ 1V.	$\beta = 0.5$ 2V.	Armijo 1V.	Armijo 2V.	Wolfe
Tolerance $\varepsilon = 10^{-8}$						
1	16	47	94	31	31	45
0.5	10	31	63	16	16	31
0.1	2	19	36	11	11	18
Tolerance $\varepsilon = 10^{-12}$						
1	31	210	266	125	125	188
0.5	10	142	168	78	78	131
0.1	2	105	136	61	61	79

Table 2: CPU times in milliseconds for different intervals $[0, \Delta x]$

The value $\rho_0 = 0.5$ is considered as the initial stepsize for the descent methods. We denote by $\beta = 0.5$ 1V the first version of the β stepsize algorithm taking $\beta = 0.5$, and $\beta = 0.5$ 2V is used for the second version. For both versions of the strategy based on the Armijo rule, $\mu = 0.1$ and $\beta = 0.5$ have been used as parameters and $m_1 = 0.1, m_2 = 0.6$ and $a = 2.0$ are taken when the Wolfe conditions are considered.

In both versions of the β stepsize method, $\beta \in (0, 1)$ is a parameter to be chosen. Up to now, $\beta = 0.5$ has been chosen. In order to study the influence of this parameter in the numerical simulations, the problem test has been solved for different values of β, taking $\varepsilon = 10^{-12}$ as tolerance and $\Delta x = 1$ (see Table 3).

β stepsize method				
	First version		Second version	
β	Iterations	CPU time	Iterations	CPU time
0.1	10	212	10	266
0.2	10	210	10	267
0.3	10	211	10	266
0.4	10	210	10	266
0.5	10	210	10	266
0.6	10	212	9	245
0.7	6	152	6	234
0.8	5	139	5	215
0.9	4	125	4	190
0.99	4	290	3	320

Table 3: Number of iterations and computational time for different values of β

Notice that, in both versions of the algorithm, the number of iterations decreases as β tends to 1 and the computational cost increases as β tends to 0 or to 1. In view of the results, the first version of the β stepsize method with $\beta = 0.9$ seems to be the best choice for this problem test.

As it can be observed, in spite of the bigger number of Cauchy problems to be solved at every iteration, Newton's method perform better than gradient methods both in number of iterations and in CPU time. Moreover, the differences increase as Δx and the tolerance decrease.

Furthermore, the fixed stepsize strategy combined with all the strategies considered to select the step has been also tested. Despite the fact that the CPU times obtained when applying this strategy are smaller than the ones shown in Table 2, Newton's method is still cheaper in every case. The conjugate gradient method, which has been also applied, increases the computational cost in relation to the gradient methods. Therefore, we conclude that in our case, the Newton's method is the most efficient strategy in order to solve the control problem.

In order to confirm this statement, a similar study has been performed in the numerical examples considered in the next Section. The conclusion has been the same in all cases: for the typical values of Δx used in the applications, Newton's method is always the most efficient in terms of the computational cost.

5 Numerical Experiments

In order to implement the high-order well-balanced numerical methods introduced in Section 1 we consider:

- Rusanov numerical flux;
- the third order CWENO reconstructions (see [9], [3]);
- the third order TVD Runge-Kutta for solving the ODE system (3): see [7];
- the Gauss two points quadrature rule;
- the standard fourth order Runge-Kutta method for solving the state and the adjoint ODE problems related to the control problem at every cell $[x_{i-1/2}, x_{i+1/2}]$. The submesh considered in the cell consists of three $(N_p + 1)$-point uniform partitions of the subintervals

$$[x_{i-1/2}, x_0^i], [x_0^i, x_1^i], [x_1^i, x_{i+1/2}],$$

where x_j^i, $j = 0, 1$ are the quadrature points. The total number of points of the submesh is thus of $3N_p + 1$

When the initial condition is a stationary solution U^* in an interval $[a, b]$, we approximate its cell averages either by applying the quadrature formula to the exact solution (when it is avalable) or by

$$U_{h,i}^* = \sum_{l=0}^{M} \alpha_l^i U_{h,l}^{*,i}$$

where $U^{*,i}_{h,j}$ are the approximations at the quadrature points obtained using RK4 to approximate (2) with initial condition

$$U(a) = U^*(a).$$

Observe that the only information about the particular problem required by the numerical method is f, S, H, G, ∇G (see (1), (10), (15)) what leads to very general algorithms.

The following symbols will be used in this section to denote the different methods considered:

- SM3: numerical method of third order based on the Rusanov flux and the standard reconstruction operators.
- NWBM3: numerical method of third order based on the Rusanov flux and the well-balanced reconstruction operators in which problems (5) are solved numerically using the Newton's method.

5.1 Burgers Equation with a Nonlinear Source Term

We consider Burgers equation with a non-linear source term:

$$\begin{cases} u_t + \left(\dfrac{u^2}{2}\right)_x = \sin(u), \quad x \in \mathbb{R},\ t > 0, \\ u(x,0) = u_0(x). \end{cases} \tag{22}$$

This problem is the particular case of (1) corresponding to:

$$U = u, \quad f(U) = \frac{u^2}{2}, \quad S(U) = \sin(u), \quad H(x) = x.$$

The ODE satisfied by the stationary solutions is

$$\frac{du}{dx} = \frac{\sin(u)}{u}. \tag{23}$$

Therefore:

$$G(x,U) = \frac{\sin(u)}{u}, \quad \partial_U G(x,U) = \frac{u\cos(u) - \sin(u)}{u^2}.$$

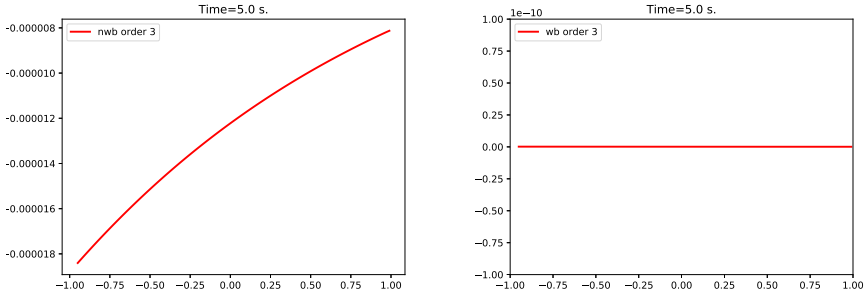
In this case, the stationary solutions satisfy the EDO studied in the test problem introduced in Section 4. These stationary solutions cannot be expressed in terms of elementary functions so that (5) has to be numerically solved.

We consider $x \in [-1,1]$, $t \in [0,5]$ and $CFL = 0.9$. The initial condition is the solution of the Cauchy problem consisting of (23) with initial condition

$$u(-1) = 2,$$

which is a stationary solution of the problem. This solution is approximated using the RK4 method and $N_p = 1$ is considered.

$u(-1, t) = 2$ is imposed at $x = -1$ and free boundary conditions are considered at $x = 1$.



(a) SM3 (b) NWBM3

Fig. 1: Test 5.1. Differences between the stationary solution and the numerical solutions at $t = 5s$. Number of cells: 100

Figure 1 shows the differences between the stationary solution and the numerical results obtained with SM3 (left) and NWBM3 (right). Table 4 show the L^1-errors and the empirical order of convergence of SM3. Notice that the non well-balanced methods perturb the stationary solution.

Cells	SM3		NWBM3
	Error	Order	Error
100	7.66E-5	-	2.54E-13
200	9.62E-8	10.506	3.60E-14
400	1.21E-10	7.254	2.12E-14
800	1.51E-11	2.922	9.11E-14

Table 4: Test 5.1. Errors in L^1 norm and convergence rates for SM3 and errors in L^1 norm and convergence rates for NWBM3

The maximum number of iterations required to solve the nonlinear problem (5) applying Newton's method is two and it converges in only one iteration for meshes with 200 cells or more. As expected, the well-balanced modification increases the computational effort. The computational cost required for solving the problem with 100 cells is 50 ms for SM3 and 760 ms if NWBM3 is applied. If the number of

cells considered is 200, the CPU time for SM3 is 190 ms, whereas for NWBM3 is 2220 ms. In any case, this extra computational cost is lower than the one that would be required to lead the discretization errors to close to zero machine by refining the mesh or increasing the order of non-well-balanced methods.

5.2 Shallow Water Equations

Let us consider the shallow water model, which is the particular case of (1) corresponding to the choices $N = 2$,

$$U = \begin{pmatrix} h \\ q \end{pmatrix}, \quad f(U) = \begin{pmatrix} q \\ \dfrac{q^2}{h} + \dfrac{g}{2}h^2 \end{pmatrix}, \quad S(U) = \begin{pmatrix} 0 \\ gh \end{pmatrix}.$$

The variable x makes reference to the axis of the channel and t is the time; $q(x,t)$ and $h(x,t)$ are the discharge and the thickness, respectively; g is the gravity and $H(x)$ is the depth function measured from a fixed reference level. We denote by $u = q/h$ the depth-averaged velocity and $c = \sqrt{gh}$.

The eigenvalues of the Jacobian matrix $D_f(U)$ of the flux function $f(U)$ are the following:

$$r_1 = u - \sqrt{c}, \quad r_2 = u + \sqrt{c}.$$

The Froude number, given by $Fr(U) = |u|/c$ indicates the flow regime: subcritical ($Fr < 1$), critical ($Fr = 1$) or supercritical ($Fr > 1$).

If $Fr(U) \neq 1$ the system of ODE satisfied by the stationary solutions is

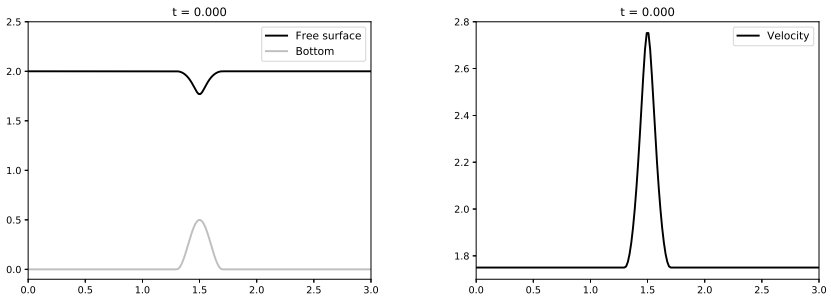
$$\begin{cases} q_x = 0, \\ h_x = \dfrac{ghH_x}{-u^2 + gh}. \end{cases} \tag{24}$$

5.2.1 A Subcritical Stationary Solution

Let us consider a test case taken from [4]: $x \in [0,3]$, $t \in [0,5]$, and the depth function is given by:

$$H(x) = \begin{cases} -0.25(1 + \cos(5\pi(x + 0.5))) \text{ if } 1.3 \leq x \leq 1.7, \\ 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{25}$$

As initial condition, we consider the subcritical stationary solution of (24) with initial conditions $h(0) = 2$, $q(0) = 3.5$, (see Figure 2).



(a) Initial condition. Free surface and bottom.

(b) Initial condition. Velocity.

Fig. 2: Test 5.2.1. Initial condition: a subcritical stationary solution computed with RK4.

As boundary conditions, $q(0,t) = 3.5$ is set upstream, while the water height is imposed to be $h(3,t) = 2.0$ downstream. The CFL parameter is set again to 0.9. The conclusions are similar to the previous test case: Figure 4 shows the differences between the stationary solution and the numerical results obtained with SM3 (up) and NWBM3 with $N_p = 3$ (down). Table 5 shows the L^1-errors and the empirical order of convergence for SM3 and NWBM3 with $N_p = 1$ and $N_p = 3$. Errors in NWBM3 are due to the numerical approximation of the stationary solution with RK4 and thus the empirical order of convergence is 4. In any case they are significantly lower than those corresponding to SM3.

Concerning the computational cost, we have checked the effect of using Newton's method or its modification in which the adjoint variable is recomputed every K iterations. Since, in this case, the maximum number of iterations of Newton's method throughout the computations is 6, we have compared the computational effort for values of K ranging from 1 (the adjoint variable is recomputed at every iteration) to 6 (it is only computed once at the beginning in all cases): Figure 3 shows the CPU times for the third order method. As it can be seen, the best option is to solve the adjoint problem only once at the beginning.

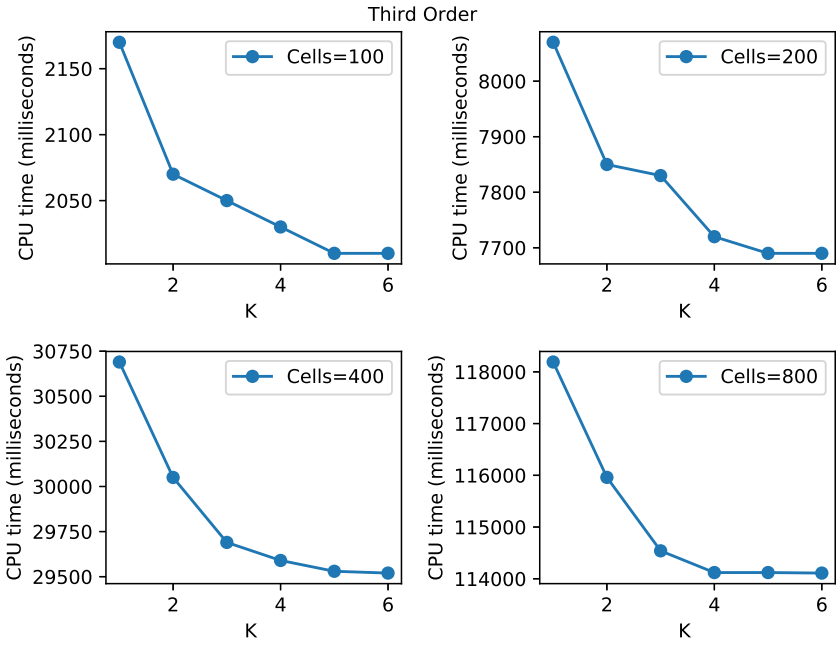
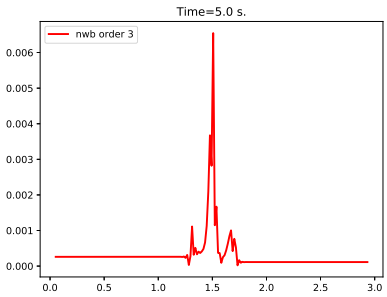
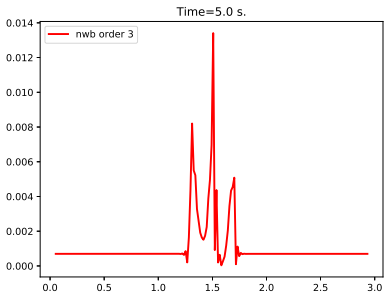


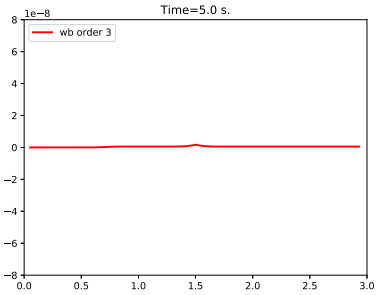
Fig. 3: Test 5.2.1. CPU times corresponding to NWBM3 with different number of cells and different values of K



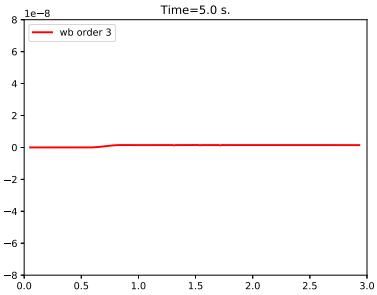
(a) SM3: h



(b) SM3: q



(c) NWBM3 (with $N_p = 3$): h



(d) NWBM3 (with $N_p = 3$): q

Fig. 4: Test 5.2.1. Differences between the stationary solution and the numerical solutions at $t = 5s$. Number of cells: 200

Cells	SM3		NWBM3			
			$N_p = 1$		$N_p = 3$	
	Error	Order	Error	Order	Error	Order
h						
100	5.98E-3	-	7.45E-6	-	3.75E-8	-
200	9.16E-4	2.707	1.93E-7	5.271	1.40E-9	4.743
400	1.21E-4	2.920	6.63E-9	4.863	6.50E-11	4.429
800	1.60E-5	2.919	2.42E-10	4.776	3.23E-12	4.331
q						
100	2.12E-2	-	1.83E-5	-	9.16E-8	-
200	3.23E-3	2.714	4.70E-7	5.283	3.39E-9	4.756
400	4.26E-4	2.923	1.62E-8	4.859	1.58E-10	4.423
800	5.47E-5	2.961	5.94E-10	4.769	8.53E-12	4.051

Table 5: Test 5.2.1. Errors in L^1 norm and convergence rates for SM3 and NWBM3

Again, the well-balanced procedure increases the computational effort. The computational cost required for solving the problem with 100 cells is 300 ms for SM3 and 2010 ms if NWBM3 with $N_p = 1$ is applied and with 200 cells, the CPU time for SM3 is 1020 ms, whereas for NWBM3 with $N_p = 1$ is 7690 ms. The computational cost increases linearly with N_p.

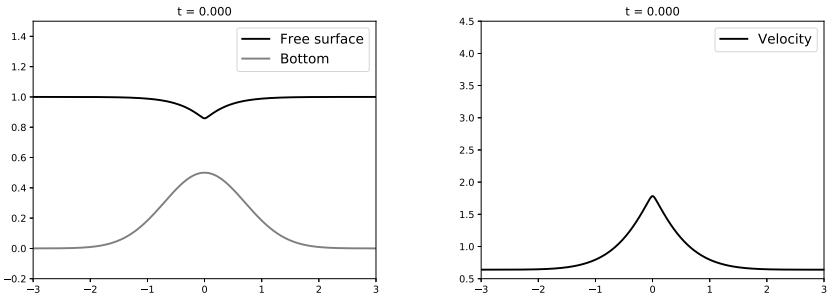
5.2.2 A Transcritical Solution

As it has been mentioned in Section 2 the method introduced here only preserves stationary solutions whose regime doesn't change, i.e. subcritical or supercritical stationary solutions. To do this, when a critical state is detected in the stencil $\mathcal{S}_i$, i.e. if there exists x in the stencil such that $D_f(U(x))$ is singular, the standard CWENO reconstructions is applied. The detection of critical states is performed by using a threshold ϵ: if the Froude number Fr is close to one, in the sense that $|Fr-1| < \epsilon$, the standard CWENO reconstruction operator is applied. Otherwise, the well-balanced reconstruction operator is computed.

To check if the numerical methods behave correctly in the presence of transcritical regimes, the following test has been considered: the shallow water equations are solved in the space interval $[-3, 3]$ and the time interval $t \in [0, 20]$ with the depth function:

$$H(x) = -\frac{1}{2}e^{-x^2}; \tag{26}$$

As initial condition, the subcritical stationary solution of (24) satisfying $h(-3) = 1$, $q(-3) = 0.64$, is imposed (see Figure 5).

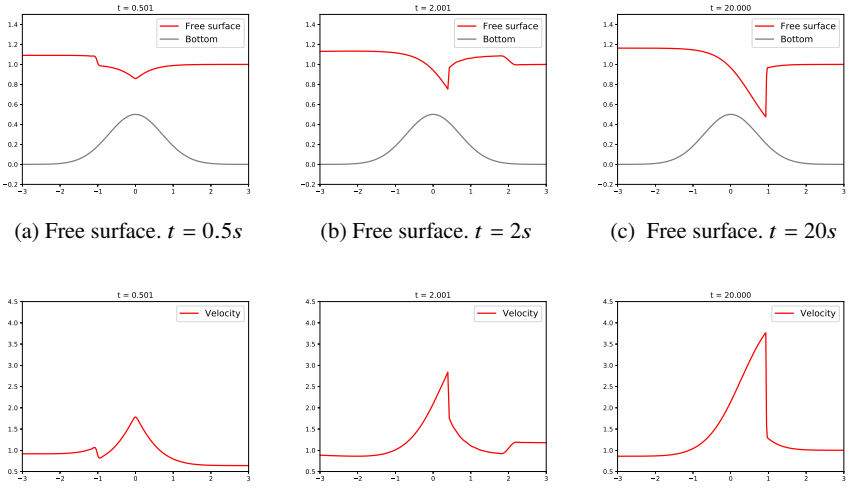


(a) Initial condition. Free surface and bottom.

(b) Initial condition. Velocity.

Fig. 5: Test 5.2.2. Initial condition: a subcritical stationary solution computed with RK4

As boundary conditions, $q(-3,t) = 1$ is set upstream, while the water height is imposed to be $h(3,t) = 1$ downstream. The CFL parameter is set to 0.5, $\Delta x = 0.02$, and $N_p = 1$ is considered. Figure 6 shows the evolution of the solution obtained with NWBM3: after the passage of the wave generated by the boundary condition, a critical state is reached at the point of minimal depth linking a subcritical regime to the left and a supercritical regime to the right, followed by a stationary hydraulic jump that links the supercritical region to the subcritical one on the right. As it can be seen the transcritical regime is well captured by the numerical method and a stationary solution is reached.



(a) Free surface. $t = 0.5s$ (b) Free surface. $t = 2s$ (c) Free surface. $t = 20s$

(d) Velocity. $t = 0.5s$ (e) Velocity. $t = 2s$ (f) Velocity. $t = 20s$

Fig. 6: Test 5.2.2. Evolution from a subsonic to a transonic regime simulated with NWBM3. Number of cells: 300

Therefore, this basic strategy described in this section allows us to simulate the evolution of transcritical stationary solutions.

6 Conclusions

The methodology presented in [1] has been followed to obtain a general family of high-order well-balanced numerical methods that can be applied to 1d systems of balance laws. The main difficulty related to the implementation of these methods is that a nonlinear problem has to be solved at every cell and at every time step

consisting in finding a stationary solution whose average is the given cell value. This problem has been interpreted as a control one related to an ODE system, in which the constraint is the given average and the control is the initial condition. The problem has been then written in functional form and the gradient of the functional has been computed with the help of the adjoint system. Once the expression of the gradient has been obtained, Newton's and descent methods have been applied. In order to test the efficiency of both methods, they have been tested in several problems, showing that the Newton's method is more efficient: a nonlinear scalar test problem has been shown as an example.

In order to test the efficiency and the well-balancedness of the methods, they have been applied to two problems: the Burgers equation with a nonlinear source term and the shallow water model. The tests put on evidence that the well-balanced modification increases the computational cost. In any case, this extra computational cost is lower than the one that would require to lead the discretization errors to (close to) zero machine by refining the mesh or increasing the order of non-well-balanced methods.

Further developments include applications of the introduced technique to:

- Systems of balance laws (1) in which the function H has jump discontinuities.
- Transcritical stationary solutions.
- Multidimensional problems.

References

1. Castro, M.J., Gallardo, J.M., López-García, J.A., Parés, C.: Well-balanced high order extensions of Godunov method for linear balance laws. SIAM Journal on Numerical Analysis, **46**, 1012–1039 (2008)
2. Castro M.J., Morales de Luna, T., Parés, C.: Well-balanced schemes and path-conservative numerical methods. In: Abgrall, R., Shu, C.-W. (eds.) Handbook of Numerical Methods for Hyperbolic Problems. Handbook of Numerical Analysis, vol. **18**, pp. 131 – 175. Elsevier (2017)
3. Cravero, I., Semplice, M.: On the accuracy of WENO and CWENO reconstructions of third order on nonuniform meshes. Journal of Scientific Computing, **67**, 1219–1246 (2016)
4. Castro, M.J., López-García, J.A., Parés, C.: In High order exactly well-balanced numerical methods for shallow water systems. Journal of Computational Physics, **246**, 242–264 (2013)
5. Castro M.J., Parés, C.: Well-Balanced High-Order Finite Volume Methods for Systems of Balance Laws. Journal of Scientific Computing, **82**, 48, (2020)
6. Gómez-Bueno, I., Castro, M.J., Parés, C.: High-order well-balanced methods for systems of balance laws: a control-based approach. Applied Mathematics and Computation. Accepted, (2021)
7. Gottlieb, S., Shu, C.-W.: Total variation diminishing Runge-Kutta schemes. Mathematics of Computation, **67**, 73–85 (1998)
8. Hager, W.W., Xhang, H.: A survey of nonlinear conjugate gradient methods. Pacific journal of Optimization, **2**, 35–58 (2006)
9. Levy, D. Puppo, G., Russo, G.: Compact central WENO schemes for multidimensional conservation laws. SIAM Journal on Scientific Computing, **22**, 656–672 (2000)