Institutionen för datavetenskap
Umeå universitet
Eddie Wadbro, Martin Berggren

16 november 2011
Teknisk beräkningsvetenskap II

Case study II: Models of transport, waves, and shallow waters
Part 1. Conservation laws and finite volume methods

In the computer exercises in section 3 below, we will acquaint ourself with a few basic numerical methods for transport phenomena modeled by conservation laws. In Part 2 of this case study, we will apply these methods and simulate the fluid flow resulting from opening a sluice gate between two water reservoirs at different water levels.

Before describing the exercises, we will in sections 1 and 2 review some basic facts about conservation laws and finite volume methods for such equations.

1 Scalar conservation laws

Let u be a physical quantity defined in a domain $\Omega \subset \mathbb{R}^n$. Typical examples of quantities of interest are densities of mass, momentum, energy, or the concentration of a chemical compound. (Here we assume, for simplicity of notation, that u is a scalar, but everything below can be generalized to the case when the quantity of interest is a vector, that is, to the case of a *system* of conservation laws. We will consider a system of conservation laws in Part 2). In many cases, the integral of quantity u over an arbitrary *control volume* $\omega \subset \Omega$ is *conserved* in the sense that it is only changed by fluxes of the quantity through the control volume's boundary $\partial\omega$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\omega} u\,dV = -\int_{\partial\omega} \boldsymbol{n}\cdot\boldsymbol{f}(u)\,dS, \tag{1}$$

where $\boldsymbol{n}$ is the outward-directed unit normal. Equation (1) says that the only factor that increases the integral of u over a control volume (the left side) is the net influx of the quantity through the boundary of the control volume (the right side). The function $\boldsymbol{f}$ is called a *flux function,* and it governs the flux of u through the boundary. The flux function is, in general, a nonlinear function of u. Applying the divergence theorem on the right side of equation (1) and using that control volume ω is arbitrary, we find that the equation can be written

$$\frac{\partial u}{\partial t} + \nabla\cdot\boldsymbol{f}(u) = 0 \quad \text{in } \Omega. \tag{2}$$

Equation (2) is the conservation law in *differential form,* whereas equation (1) is its *integral form.* The integral form is in a sense the more fundamental one: it follows directly from the physical principle of conservation, and it does not involve any derivatives of the flux function. The integral form is also the basis for the finite-volume methods presented below.

From now on, we will only consider conservation laws in one space dimension, that is,

$$u_t + f(u)_x = 0, \tag{3}$$

in differential form, and

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b u\,dx + f(u(b)) - f(u(a)) = 0, \tag{4}$$

in integral form. If everything is nice and smooth, the chain rule can be used to rewrite the *conservative form* (3) into the *primitive form,*

$$u_t + a(u)u_x = 0, \tag{5}$$

where $a(u) = f'(u)$.

An important concept in the theory of conservation laws is the *characteristics* associated with the conservation law in question. Loosely speaking, the characteristics are the paths in

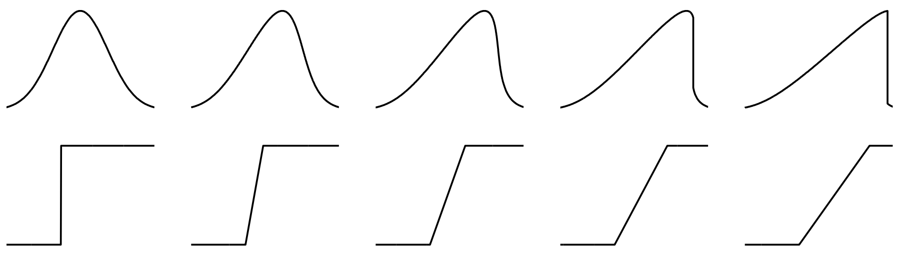

FIGURE 1. Solutions of Burgers' equation for two different initial conditions at (from left to right) times t=0, 0.5, 1, 1.5, and 2. In the first (upper) case, the initially smooth solution eventually develops a shock (discontinuity) that is visible from time $t = 1.5$. In the second (lower) case, a rarefaction wave is developed from the initially discontinuous solution.

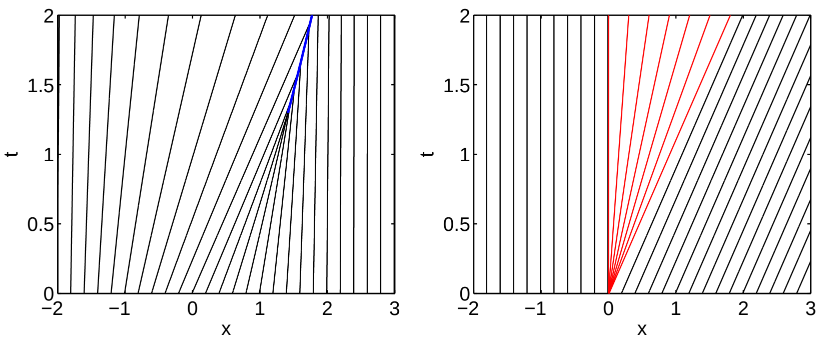

FIGURE 2. Characteristics for the two solutions of Burgers' equation in Figure 1. The left and right diagrams correspond to the solutions in the top and bottom row of Figure 1, respectively. The characteristic path corresponding to the shock in the top row of Figure 1 is marked blue and the characteristics representing the rarefaction wave in the bottom row of Figure 1 are marked red.

space-time along which information flows. The characteristics associated with problem (3) (or (5)) are the curves in the (x,t)-plane given by $(X(t),t)$, where $X(t)$ satisfies the ODE

$$\begin{aligned} X'(t) &= a\big(u(X(t))\big), \\ X(0) &= x_0. \end{aligned} \tag{6}$$

The characteristics have the the important property that *smooth solutions to the conservation law are constant along the characteristics*, which means that the initial condition of the conservation law is propagated into the (x,t) plane along the characteristics. This fact is proven by applying the chain rule on the solution evaluated along the characteristic curve:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}u(X(t),t) &= u_t(X(t),t) + u_x(X(t),t)X'(t) = \big[\text{by (6)}\big] \\ &= u_t(X(t),t) + u_x(X(t),t)a(u(X(t))) = \big[\text{by (5)}\big] \\ &= 0. \end{aligned} \tag{7}$$

Figure 1 illustrates two solutions of Burgers' equation $u_t + (u^2/2)_x = 0$ (a nonlinear conservation law that we will study further in the exercises), and figure 2 shows corresponding characteristics in the (x,t) plane. These figures illustrates two effects that are typical for nonlinear conservation laws, namely the appearance of a *shock* (sv. *stöt*), that is, a discontinuity in the solution, and a *rarefaction wave*, a smearing-out of an initial discontinuity.

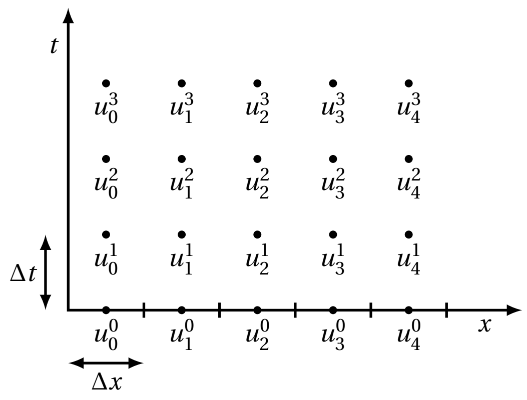

FIGURE 3. The finite volume scheme computes approximations u_i^n of cell averages in cells of width Δx at times $t_n = n\Delta t$, $n = 0, 1, \ldots$

It is crucial to have knowledge about the directions of the characteristics when setting boundary conditions to a conservation law. A boundary condition for a scalar conservation law can be imposed only when the characteristic at corresponding boundary is directed *into* to the domain. The boundary condition then provides the information that is flowing into the domain along the characteristic direction. However, if the characteristic is directed *out* of the domain at the boundary, no boundary condition can be set, since there is no information flowing into the domain at that boundary.

2 Finite volume methods for conservation laws in one space dimension

The starting point for finite volume methods is the conservation law in its integral form (4). The x-axis is divided into *computational cells* of uniform width Δx. We denote by x_i, $i = 1, 2, \ldots$ the center of the cells $(x_{i-1/2}, x_{i+1/2})$ with end points $x_{i\pm1/2} = x_i \pm \Delta x/2$. Moreover, we introduce a *time step* Δt and define $t_n = n\Delta t$, $n = 0, 1, \ldots$. Finite volume methods compute approximations to *cell averages* at discrete time steps,

$$u_i^n \approx \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x, t_n)\,dx. \tag{8}$$

Figure 3 visualizes the locations in the (x, t) plane of the u_i^ns.

Applying equation (4) to computational cell i yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_{i-1/2}}^{x_{i+1/2}} u\,dx + f\big(u(x_{i+1/2})\big) - f\big(u(x_{i-1/2})\big) = 0, \tag{9}$$

a form of the equation that suggests the following family of *explicit finite volume schemes*:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} = 0, \tag{10}$$

or, in a form that is closer to how it is actually implemented in the computer code,

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\big(F_{i+1/2}^n - F_{i-1/2}^n\big). \tag{11}$$

The quantities $F_{i\pm1/2}^n$ are called *numerical flux functions* and constitute approximations to $f\big(u(x_{i\pm1/2}, t_n)\big)$. Each particular member in the class of methods (11) is defined by the choice of flux function. In fact, the development of finite volume methods has largely been centered around the art of designing various ingenious flux functions, and numerous different suggestions have been made over the years. Here we will only consider a few of the simplest ones. Note that the flux $f(u(x_{i\pm1/2},t_n))$ involves the (unknown) exact solution at the interface between cells, and that the u_i^ns represent cell averages that are discontinuous at the cell interfaces. Typically, various combinations of u_j^n at different cells j in the vicinity of the cell boundary in question are used in the definition of the numerical fluxes.

2.1 Examples of finite volume schemes

2.1.1 The upwind scheme.

The primitive form (5) shows that information locally is transported from left to right when $f' > 0$ and from right to left when $f' < 0$. This directional property motivates the following choice of *upwind* numerical flux function:

$$F_{i-1/2}^n = \begin{cases} f(u_{i-1}^n) & \text{if } f' > 0, \\ f(u_i^n) & \text{if } f' < 0; \end{cases} \qquad F_{i+1/2}^n = \begin{cases} f(u_i^n) & \text{if } f' > 0, \\ f(u_{i+1}^n) & \text{if } f' < 0. \end{cases} \tag{12}$$

Substituting the flux function into the general scheme (11) yields the *upwind scheme*

$$u_i^{n+1} = \begin{cases} u_i^n - \frac{\Delta t}{\Delta x}\left(f(u_i^n) - f(u_{i-1}^n)\right) & \text{if } f' > 0, \\ u_i^n - \frac{\Delta t}{\Delta x}\left(f(u_{i+1}^n) - f(u_i^n)\right) & \text{if } f' < 0, \end{cases} \tag{13}$$

Thus, in the upwind scheme (as well as in many other methods not presented here), we need approximations of the local "wave speed" f' at the interfaces between cells. Such approximations can be computed, for instance, by checking the sign of the following finite-difference approximations of f':

$$\begin{aligned} a_{i+1/2} &= \begin{cases} \frac{f(u_{i+1}^n) - f(u_i^n))}{u_{i+1}^n - u_i^n} & \text{if } u_{i+1}^n \neq u_i^n, \\ f'(u_i^n) & \text{if } u_{i+1}^n = u_i^n; \end{cases} \\ a_{i-1/2} &= \begin{cases} \frac{f(u_i^n) - f(u_{i-1}^n))}{u_i^n - u_{i-1}^n} & \text{if } u_i^n \neq u_{i-1}^n, \\ f'(u_i^n) & \text{if } u_i^n = u_{i-1}^n. \end{cases} \end{aligned} \tag{14}$$

2.1.2 The Lax–Friedrichs method

The upwind scheme is very simple, but it needs information about the local "wind direction" f' in order to know from which direction to compute the flux. The Lax–Friedrichs method,

$$u_i^{n+1} = \frac{1}{2}\left(u_{i-1}^n + u_{i+1}^n\right) - \frac{\Delta t}{2\Delta x}\left(f(u_{i+1}^n) - f(u_{i-1}^n)\right). \tag{15}$$

is even simpler to implement, since it is a "central scheme" that does not need information about f'. The Lax–Friedrichs method also belongs to the family (10) of methods with the following numerical flux function:

$$\begin{aligned} F_{i-1/2}^n &= \frac{1}{2}\left(f(u_{i-1}^n) + f(u_i^n)\right) - \frac{\Delta x}{2\Delta t}\left(u_i^n - u_{i-1}^n\right), \\ F_{i+1/2}^n &= \frac{1}{2}\left(f(u_i^n) + f(u_{i+1}^n)\right) - \frac{\Delta x}{2\Delta t}\left(u_{i+1}^n - u_i^n\right). \end{aligned} \tag{16}$$

2.1.3 The Richtmyer two-step Lax–Wendroff method

The upwind and Lax–Friedrich methods are simple and very robust but only first-order accurate in space and time for smooth solutions. That is, the discretization error can only be expected to reduce in proportion to reductions in Δx and Δt. A method that is second-order accurate both

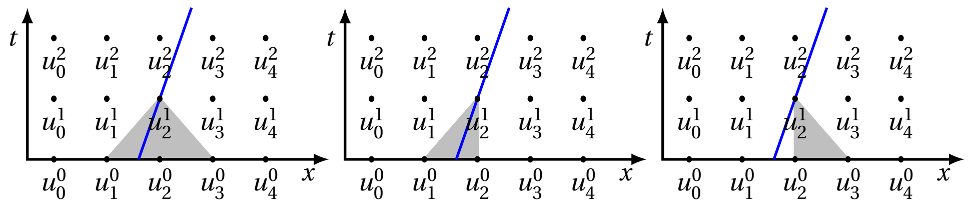
t
u_0^2 u_1^2 u_2^2 u_3^2 u_4^2
u_0^1 u_1^1 u_2^1 u_3^1 u_4^1
u_0^0 u_1^0 u_2^0 u_3^0 u_4^0
x

FIGURE 4. Numerical domain of dependence (marked gray) in the computation of u_2^1 for the Lax–Friedrichs and Lax–Wendroff methods (left), for the upwind scheme when $f' > 0$ (middle), and the the upwind scheme when $f' < 0$ (right). The blue line illustrates a characteristic curve for the transport equation $u_t + cu_x = 0$ for some $c > 0$. The CFL condition is satisfied in the left and middle picture and violated in the right picture.

in space and time is the *Richtmyer two-step Lax–Wendroff method*. The price for the increased accuracy is a somewhat more complex scheme that involves an intermediate state $u_{i+1/2}^{n+1/2}$,

$$\begin{aligned} u_{i+1/2}^{n+1/2} &= \frac{1}{2}\big(u_i^n + u_{i+1}^n\big) - \frac{1}{2}\frac{\Delta t}{\Delta x}\big(f(u_{i+1}^n) - f(u_i^n)\big), \\ u_i^{n+1} &= u_i^n - \frac{\Delta t}{\Delta x}\big(f(u_{i+1/2}^{n+1/2}) - f(u_{i-1/2}^{n+1/2})\big). \end{aligned} \tag{17}$$

This scheme can also be classified as a member of the family (10) of schemes. We omit corresponding somewhat complicated-looking expression for the numerical flux functions $F_{i\pm1/2}^n$.

2.2 The CFL condition

We define the *numerical domain of dependence* for a member of the family of schemes (11) as the region in the (x,t)-plane that precisely covers all points involved in the definition of a particular u_i^{n+1}. Figure 4 illustrates the numerical domain of dependence in the computation of u_2^1 for the numerical schemes presented above. Note that the domain of dependence is the same for the Lax–Friedrichs and Lax–Wendroff methods, whereas the domain of dependence for the upwind scheme depends on the local sign of f'. Figure 4 also depicts a characteristic curve passing through the point $(2\Delta x, \Delta t)$ for the transport equation $u_t + cu_x = 0$ in a case where $c > 0$. The *CFL condition*[1] states that a necessary condition for stability is that the characteristic curve passing through (x_i, t_{n+1}) is contained in the numerical domain of dependence. Another way of expressing the CFL condition is that the physical domain of dependence (that is, the characteristics) must be contained in the numerical domain of dependence. For the example in Figure 4, all characteristic curves $(X(t), t)$ passing through the point $(2\Delta x, \Delta t)$ must for $t \in [0, \Delta t]$ be contained in the region marked gray. Hence, for the particular case of the characteristic marked in blue, the Lax–Friedrichs and Lax–Wendroff methods, as well as the upwind scheme in the middle picture, all satisfy the CFL condition. However, the upwind scheme to the right does not satisfy the CFL condition and will thus be unstable. Note that an upwind scheme will always be unstable if the wrong upwind direction is chosen. For a correctly-chosen upwind direction and for the Lax–Friedrichs and Lax–Wendroff methods, it is always possible to ensure that the CFL condition is satisfied simply by choosing Δt small enough, since a continuing reduction of the time step will broaden the numerical domain of dependence in the x-direction until the characteristic curve (which does not depend on Δx or Δt) is contained within the numerical domain of dependence.

[1] named after Courant, Friedrichs, and Lewy, who discussed this condition in an article already in 1928!

3 Computer exercises

Hints and details on how to carry out the Matlab implementation are given in section 4.

1. Consider the transport equation
$$u_t + cu_x = 0, \tag{18}$$
where c is a positive scalar constant, with periodic boundary conditions $u(0,t) = u(1,t)$ and initial condition $u(x,0) = \sin 2\pi x$. For $c = 2$, solve the above equation
   (a) exactly,
   (b) either with the upwind scheme or the Lax–Friedrichs method (these methods behave similarly),
   (c) with the Richtmyer two-step Lax–Wendroff method.

   Test various combinations of time and space discretizations in the above schemes and compare with the exact solution. For a fixed space discretization, change Δt and find the smallest so-called *Courant number*, $c\Delta t/\Delta x$, for which the numerical solution starts behaving unstably. Change c and redo the above experiment. Does the smallest Courant number for which the system behaves unstable seem to depend on c, Δt, or Δx?

2. In problem 1, we investigated the behavior of the numerical schemes for smooth initial data. Here, we will study how the schemes behave under the discontinuous initial condition
$$u(x,0) = \begin{cases} 1 & \text{if } x < 0.5, \\ 0 & \text{else.} \end{cases} \tag{19}$$
We will *define* the solution to equation (18) under the initial condition (19) as the transport of the initial condition along the characteristics. This solution satisfies the equation in the integral form (4) and constitutes a *weak solution* to equation (18). Note that the solution is nondifferentiable, so it does not satisfy the differential form of the equation in a classical sense. Test also here various combinations of time and space discretizations and compare with the exact solution.

3. Consider Burgers' equation
$$u_t + f(u)_x = 0, \tag{20}$$
where $f(u) = u^2/2$. We want to solve this equation on the spatial interval $(0,1)$ with initial conditions $u(0,x) = 1 + 0.7\sin(10x)$ and boundary condition $u(t,0) = 1$.
   (a) Why is no boundary condition needed on the right boundary ($x = 1$)?
   (b) Roughly sketch the characteristics in (x,t) space for the above boundary conditions. Try to anticipate how the solution will develop in different regions of the interval $(0,1)$.
   (c) Solve the problem numerically using the upwind sceme and the two-step Lax–Wendroff method.

4 Implementation instructions and hints

4.1 General instructions and hints for all exercises

- There are a number of different ways to carry out the Matlab implementation, and as usual, it is a good idea to think first before starting to code! Here are some implementation options:
  - Write a separate Matlab function for each problem.
  - Write a single function with options for the different cases.
  - A more advanced approach is to write separate functions for the numerical schemes (upwind, Lax–Friedrichs, and Lax–Wendroff). Having written these functions, the rest of the computer exercise reduces to writing flux functions for the two equations involved and experimenting by varying the initial condition, Δx, and Δt.

- Let the number of cells M and the time–space step ratio $\lambda = \Delta t / \Delta x$ be input parameters.
- Remember that x_i, $i = 1, 2, \ldots, M$ are the cell *centers* located at $\Delta x/2, 3\Delta x/2, \ldots$, and that u_i, $i = 1, 2, \ldots, M$ are corresponding cell averages. The cell *interfaces* are located at $x_{i+1/2} = i\,\Delta x$, $i = 0, 1, \ldots, M$.
- When defining the numerical initial condition, you may simply evaluate the given initial condition formula in the cell centers. This strategy corresponds to approximating the cell averages of the initial condition using the mid-point quadrature rule in the cell-average integral.
- All problems above should be solved on the interval $(0, 1)$ using M cells of equal size Δx. To create a vector with the cell interface points, you can either use `linspace(0,1,M+1)` which creates a vector with $M + 1$ equally spaced numbers between 0 and 1, or write `0:delta_x:1`, which creates a vector with equally spaced numbers with spacing Δx starting at 0 and ending at (or as close as possible before) 1. Note that for example `0:0.3:1` yields a vector containing (floating point approximations of) the numbers 0, 0.3, 0.6, and 0.9. The vector of cell interface points is not really needed for the solution of the problem. However, it is useful for debugging and when plotting the solution as described below.
- The plotting routine `plotfvm`, available at the course homepage,[2] can be used to plot piecewise-constant functions. Routine `plotfvm` takes two input arguments, `x` and `u`, where `x` is an $M + 1$ vector that contains the cell interface points $x_{1/2}, \ldots, x_{M+1/2}$, created as described above, and `u` is an M vector containing cell averages $u_1, u_2, \ldots, u_M$. Note that the lengths of `x` and `u` are different!
- To plot the exact solution in the same figure as the numerical solution, write a function (say `plotexact`) that takes the current time t as input and generates a plot of the exact solution. Then use the following command sequence to get both plots in the same figure:

  ```
  plotfvm(x,u);
  hold on;
  plotexact(t);
  ```

4.2 Specific instructions and hints for exercise number [n]

[1,2] Use that the solution is constant along the characteristics to solve the transport problems exactly.

[1,2] When solving problems with periodic boundary conditions, think of the cells as being periodically wrapped, so that cell 1 has cell M as its left neighbor and cell M has cell 1 as its right neighbor. In Matlab, the vector `u([2:end 1])` contains `u` periodically shifted one position to the left. Similarly `u([end 1:end-1])` contains `u` periodically shifted one position to the right. Using this trick, you can avoid loops for spatial indices and perform u^n_{i+1} and u^n_{i-1}, with built-in periodic boundary conditions, for all i simultaneosly!

[2] The periodic square wave initial condition can in Matlab be written as an *inline function*:

```
f = inline('round(1/2+sin(2*pi*x)/2)','x');
```

[3] Normally, when using the upwind scheme for nonlinear conservation laws, the sign of the local wave speed f' needs to be estimated at each cell interface in order to know from which side the flux should be computed. However, with the current initial condition, the wave speed will be positive at all times, which means that a flux from the left may be used without checking f'.

[2] `http://www.cs.umu.se/kurser/5DV123/HT11/case_studies.php`

Note that the boundary condition $u(0,t) = 1$ should be used for this problem (and not the periodic boundary condition of exercises 1 and 2). A convenient way to impose the boundary condition is to appropriately modify the numerical flux associated with the left boundary of the first cell. Recall the general form (10) of our finite-volume schemes, and note that the scheme at the first cell, which is centered at $x_1 = \Delta x/2$, involves the numerical flux F_0^n which approximates the flux $f(u(0,t_n))$. To impose the boundary condition, substitute in the first cell the normal left numerical flux with $F_0^n = f(u(0,t_n)) = f(1)$. For the Richtmyer two-step Lax–Wendroff method, use $f(u(0,t_n))$ instead of $f(u_{i-1/2}^{n+1/2})$ in the first cell.

Although there is no boundary condition at the right boundary $x = 1$, the Lax–Wendroff method nevertheless requires information from both neighboring cells, that is, the method needs to access u_{M+1}^n. A standard way to handle this problem is to introduce a so-called *ghost cell*, an extra cell to the right of the last cell. The value of u in this $(M+1)$th cell may be set using linear interpolation based on the values in cells M and $M-1$, that is $u_{M+1} = 2u_M - u_{M-1}$. The procedure to implement the ghost cell update is as follows. At each n, first set the values u_i^{n+1} for $i = 1, \ldots, M$ using the finite volume scheme (with the modification described above for $i = 1$ in order to set the boundary condition at $x = 0$), then set the value $u_{M+1}^{n+1} = 2u_M^{n+1} - u_{M-1}^{n+1}$.