Antidiffusion techniques to refine the numerical solution of the advection equation

Case study: Smolarkiewicz' iterative approach

Gerhard Burger Jelmer Wolterink

Scientific Computing
Department of Mathematics
Utrecht University

Project presentations SOAC, 31 October 2011

Universiteit Utrecht

Outline

Advection

Transport mechanism of a **substance** by a *fluid*, due to the fluids motion in a particular direction.

Examples in ocean, atmosphere and climate modelling

- Transport of **trace gasses** by *air* due to wind
- Transport of **heat** by *ocean water* due to currents
- Transport of **warm and moist air** over a colder surface by *air* due to wind: advection fog

Advection equation

Continuity equation describing the advection of a nondiffusive quantity in a flow field:

$$\frac{\partial \psi}{\partial t} + \text{div}(V\psi) = 0 \tag{1}$$

where $\psi(x, y, z, t)$ is the nondiffusive scalar quantity, $V = (u, v, w)$ is the velocity vector and x, y, z, t are space and time independent variables.

Universiteit Utrecht

Advection simulation

Constraints on advection simulation [**?**]

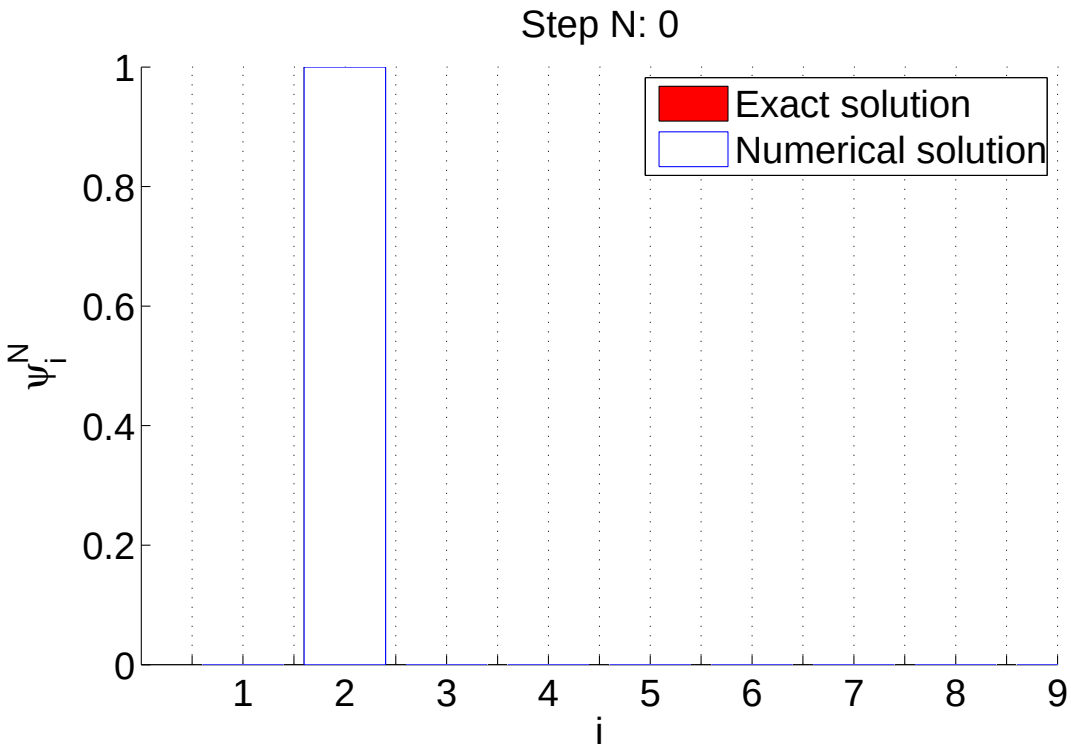
- Solutions should contain no unphysical overshoot or undershoot: positive definite
- Methods should be volume preserving. No loss of matter
- The solutions should be local: the solution at any one point should not be influenced by what is going on far away from that point
- The numerical solution should not introduce new extrema in the solution, because the continuous form of the solution would not
- The method should be cost effective: memory and computational requirements should be sufficiently small so that practical problems may be solved

Universiteit Utrecht

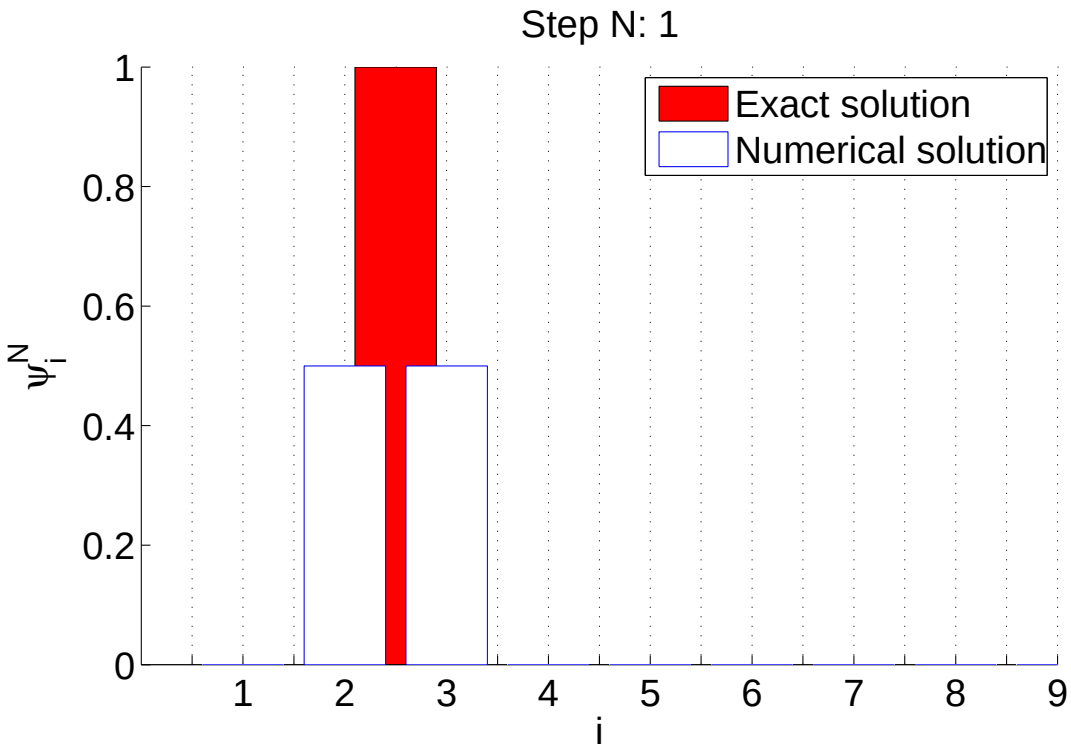
Problem: diffusion

- When simulating an advective process, we need to discretize space. When doing so, we introduce numerical constraints on the calculation and we get diffusion.
- The numerical solution spreads out.
- We get interactions, our simulations become less reliable.
- $\Longrightarrow$ example

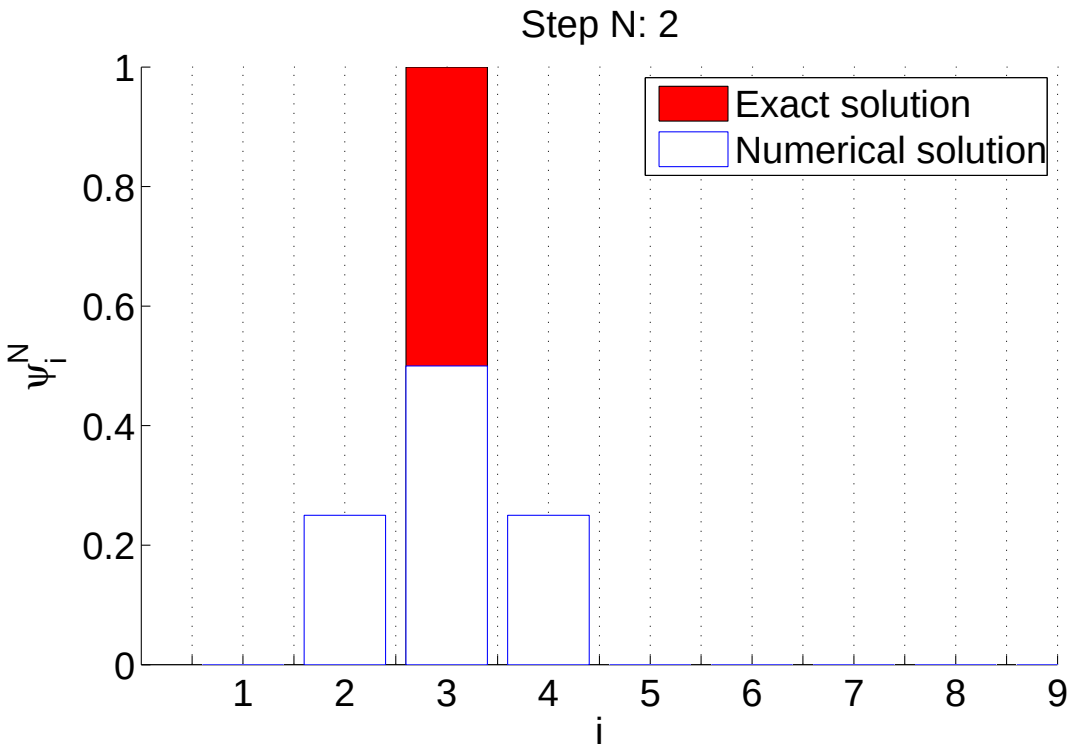
Example of diffusion with an upstream scheme



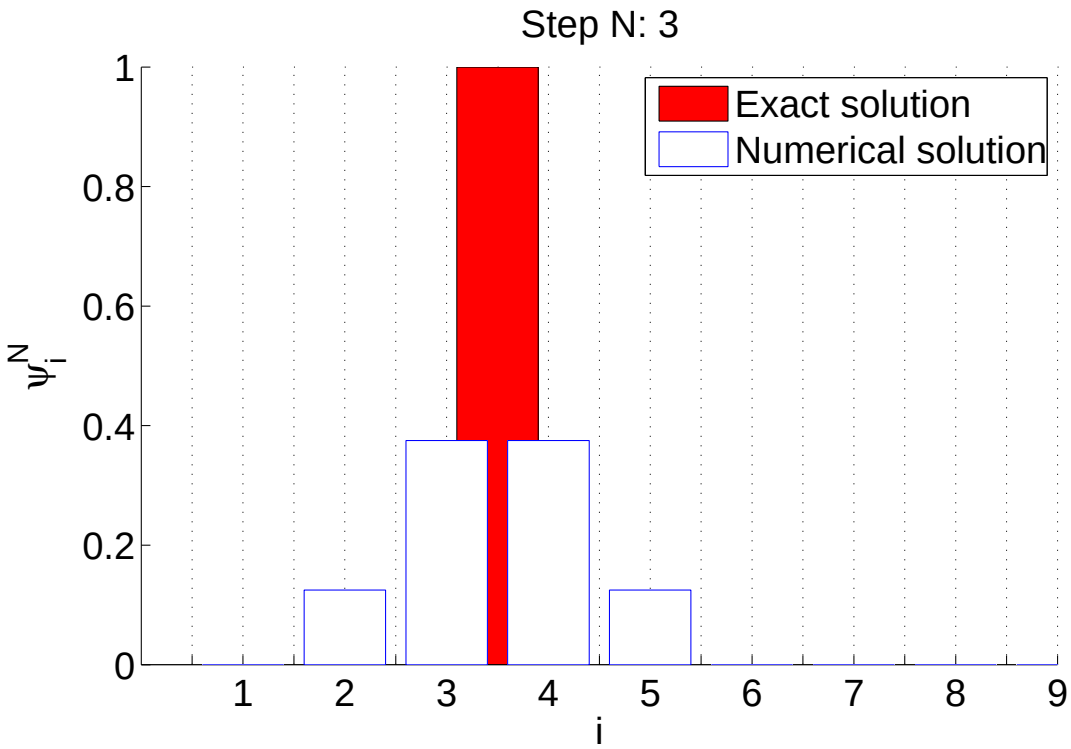
Example of diffusion with an upstream scheme



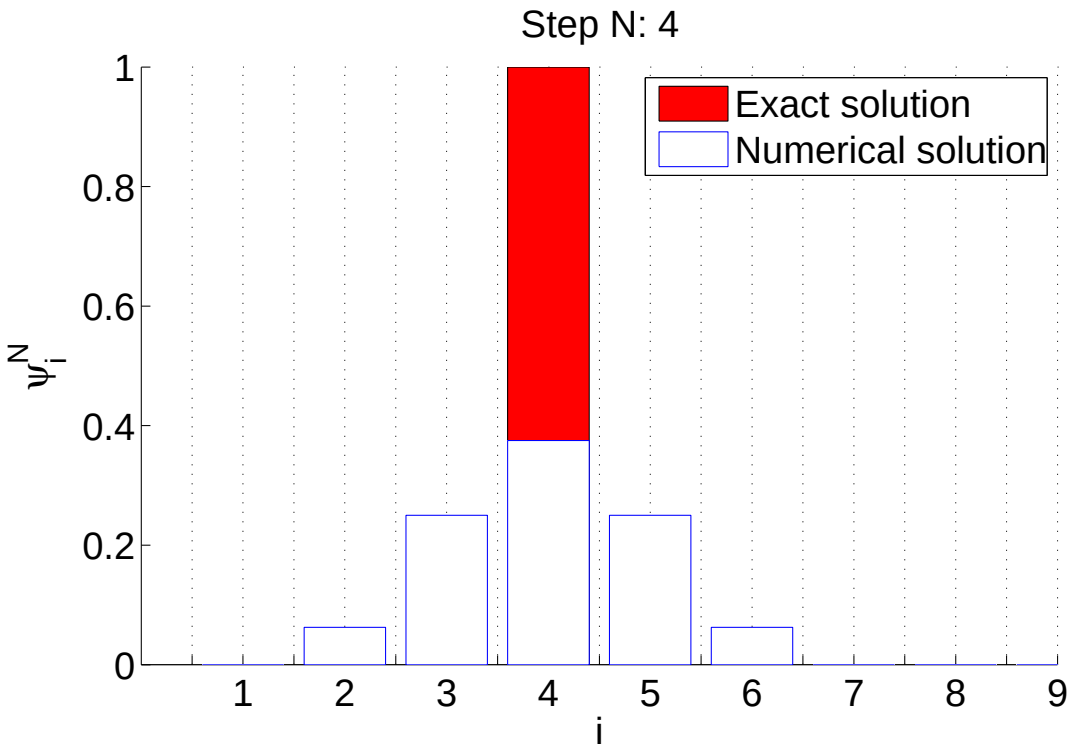
Example of diffusion with an upstream scheme



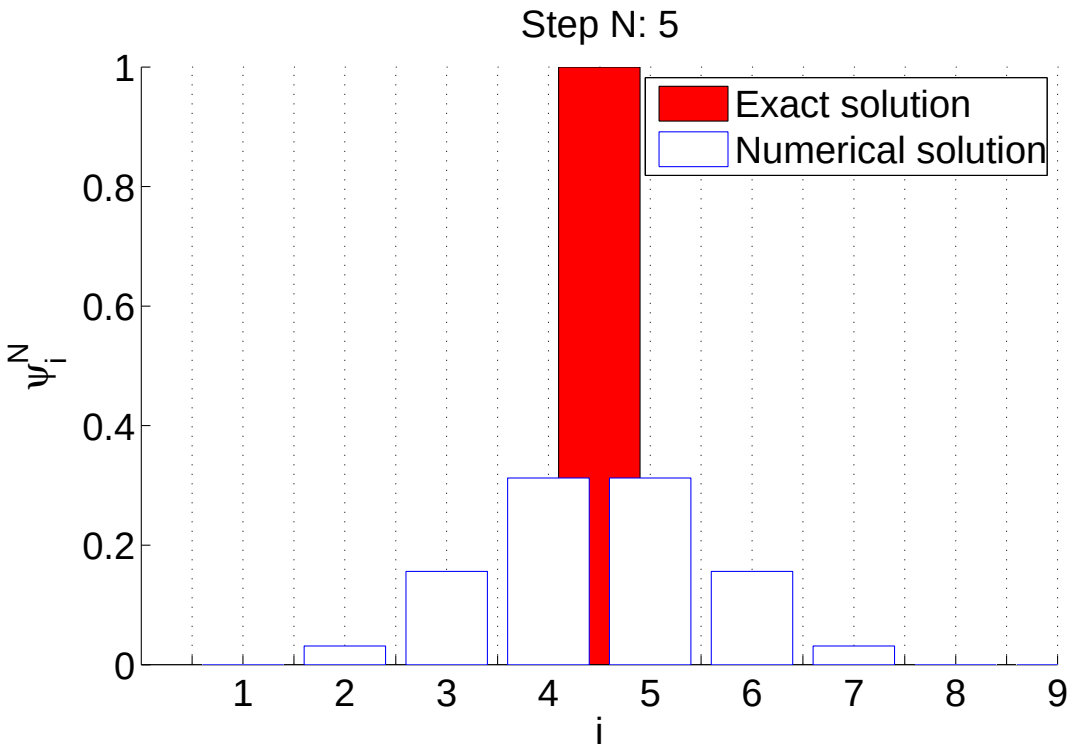
Example of diffusion with an upstream scheme



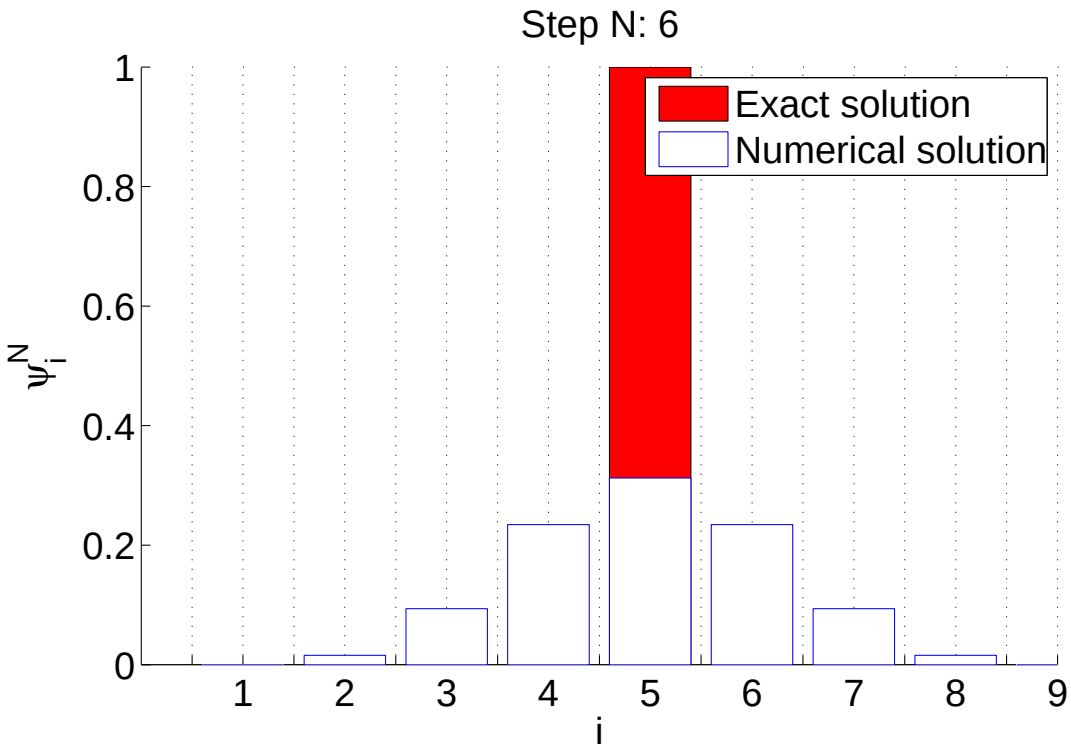
Example of diffusion with an upstream scheme



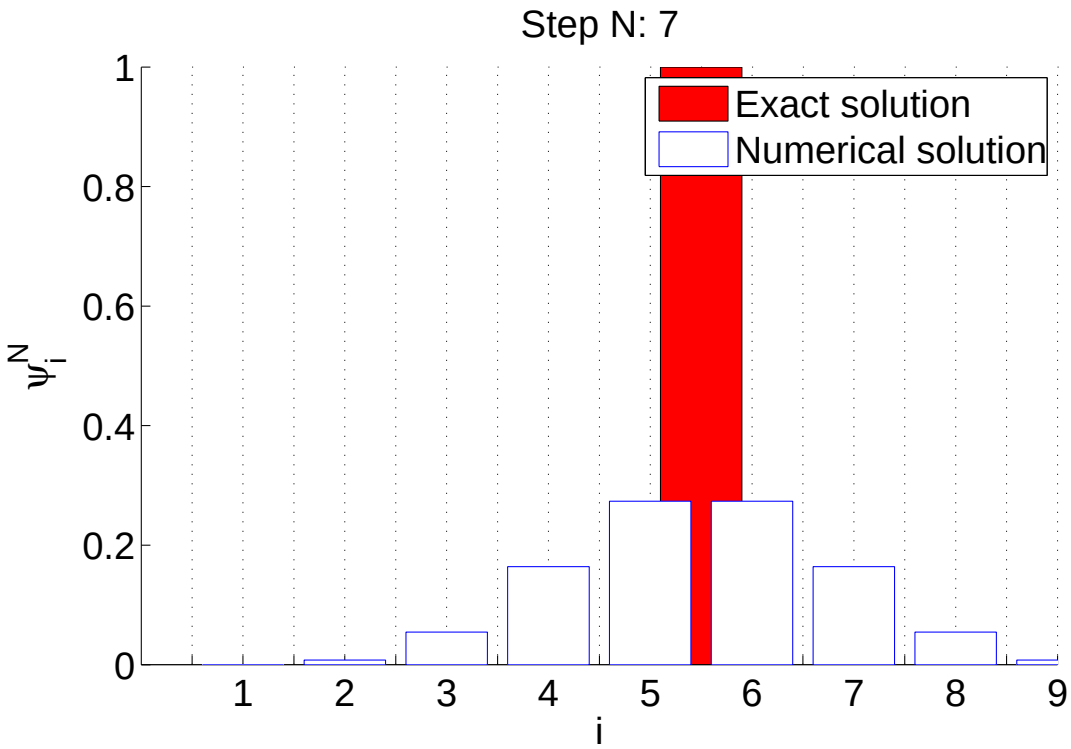
Example of diffusion with an upstream scheme



Example of diffusion with an upstream scheme



Example of diffusion with an upstream scheme



Antidiffusion methods 1

- Flux-corrected transport (FCT) method (Boris and Book, 1973),
- self-adjusting hybrid scheme (SAHS) (Harten en Zwas, 1972),

both can be very accurate but require excessive computing time, better is the

- hybrid-type scheme based on a Crowley advection scheme (Clark and Hall, 1979),

with more diffusion but half the computation time.

Antidiffusion methods 2

- Smolarkiewicz' iterative correction

less time consuming while results are comparable to those of the more complex hybrid schemes

Basis: upstream on a staggered grid

We start with the following upstream advection equation on staggered grid:

$$\psi_i^{N+1} = \psi_i^N - \left(F\left(\psi_i^N, \psi_{i+1}^N, u_{i+1/2}^N\right) - F\left(\psi_{i-1}^N, \psi_i^N, u_{i-1/2}^N\right) \right),$$

where

$$F\left(\psi_i^N, \psi_{i+1}^N, u_{i+1/2}^N\right) = \left(\left(u_{i+1/2}^N + \left| u_{i+1/2}^N \right| \right) \psi_i^N + \left(u_{i+1/2}^N - \left| u_{i+1/2}^N \right| \right) \psi_{i+1}^N \right) \frac{\Delta t}{2\Delta x}.$$

Universiteit Utrecht

Method of lines 1

Writing the scheme out and collecting terms gives

$$\begin{aligned}\psi_i^{N+1} = &\frac{\Delta t}{2\Delta x}\left(u_{i-1/2}^N + \left|u_{i-1/2}^N\right|\right)\psi_{i-1}^N \\ &+\left(1 - \frac{\Delta t}{2\Delta x}\left(u_{i+1/2}^N + \left|u_{i+1/2}^N\right| - u_{i-1/2}^N + \left|u_{i-1/2}^N\right|\right)\right)\psi_i^N \\ &-\frac{\Delta t}{2\Delta x}\left(u_{i+1/2}^N - \left|u_{i+1/2}^N\right|\right)\psi_{i+1}^N\end{aligned}$$

Universiteit Utrecht

Method of lines 2

This can be rewritten to

$$\psi_i^{N+1} = \alpha_i \psi_{i-1}^N + \beta_i \psi_i^N + \gamma_i \psi_{i+1}^N, \quad \text{for } i = 1, \ldots, M-1,$$

where we have that

$$\alpha_i = \frac{\Delta t}{2\Delta x}\left(u_{i-1/2}^N + \left|u_{i-1/2}^N\right|\right),$$

$$\beta_i = \left(1 - \frac{\Delta t}{2\Delta x}\left(u_{i+1/2}^N + \left|u_{i+1/2}^N\right| - u_{i-1/2}^N + \left|u_{i-1/2}^N\right|\right)\right),$$

$$\gamma_i = -\frac{\Delta t}{2\Delta x}\left(u_{i+1/2}^N - \left|u_{i+1/2}^N\right|\right).$$

Universiteit Utrecht

Method of lines 3

We can also write this in matrix form using Direchlet boundary conditions

- $y(0) = 0$ and $y(M) = 0$
- using periodic boundary conditions $y(0) = y(M)$

$$\begin{bmatrix} \psi_1^{N+1} \\ \psi_2^{N+1} \\ \vdots \\ \psi_{M-2}^{N+1} \\ \psi_{M-1}^{N+1} \end{bmatrix} = \begin{bmatrix} \beta_1 & \gamma_1 & & & & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \alpha_{M-2} & \beta_{M-2} & \gamma_{M-2} & \\ 0 & & & \alpha_{M-1} & \beta_{M-1} & \end{bmatrix} \begin{bmatrix} \psi_1^{N} \\ \psi_2^{N} \\ \vdots \\ \psi_{M-2}^{N} \\ \psi_{M-1}^{N} \end{bmatrix}$$

So the new values of ψ can be obtained by a sparse matrix-vector multiplication

Method of lines 3

We can also write this in matrix form using Direchlet boundary conditions

- $y(0) = 0$ and $y(M) = 0$
- using periodic boundary conditions $y(0) = y(M)$

$$\begin{bmatrix} \psi_1^{N+1} \\ \psi_2^{N+1} \\ \vdots \\ \psi_{M-2}^{N+1} \\ \psi_{M-1}^{N+1} \end{bmatrix} = \begin{bmatrix} \beta_1 & \gamma_1 & & & & \alpha_1 \\ \alpha_2 & \beta_2 & \gamma_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \alpha_{M-2} & \beta_{M-2} & \gamma_{M-2} & \\ \gamma_{M-1} & & & \alpha_{M-1} & \beta_{M-1} & \end{bmatrix} \begin{bmatrix} \psi_1^{N} \\ \psi_2^{N} \\ \vdots \\ \psi_{M-2}^{N} \\ \psi_{M-1}^{N} \end{bmatrix}$$

So the new values of ψ can be obtained by a sparse matrix-vector multiplication

Antidiffusion 1

To apply antidiffusion we need to redefine the scheme into

$$\psi_i^* = \psi_i^N - \left(F\left(\psi_i^N, \psi_{i+1}^N, u_{i+1/2}^N\right) - F\left(\psi_{i-1}^N, \psi_i^N, u_{i-1/2}^N\right) \right),$$
$$\psi_i^{N+1} = \psi_i^* - \left(F\left(\psi_i^*, \psi_{i+1}^*, \tilde{u}_{i+1/2}^N\right) - F\left(\psi_*^N, \psi_*^N, \tilde{u}_{i-1/2}^N\right) \right),$$

were the antidiffusion velocity $\tilde{u}_{i+1/2}$ is defined as

$$\tilde{u}_{i+1/2} = \frac{\left(\left| u_{i+1/2} \right| \Delta x - \Delta t u_{i+1/2}^2 \right) \left(\psi_{i+1}^* - \psi_i^* \right)}{\left(\psi_i^* + \psi_{i+1}^* + \epsilon \right) \Delta x}$$

Universiteit Utrecht

Antidiffusion 2

The Method of lines can be applied in the same way to the second step in the scheme,
so the whole scheme can be executed with two sparse matrix-vector multiplications and a matrix update.

Algorithm

input : Initial configuration ψ^0, velocities u, number of iterations iter
output: Final configuration ψ^N

$\psi \leftarrow \psi^0$;
mat1 $\leftarrow$ ComputeMatrix(u);
for $i \leftarrow 0$ **to** iter **do**
    $\psi \leftarrow$ MatrixMultiplication(mat1,ψ);
    for $j \leftarrow 1$ **to** iter **do**
        $\widetilde{u} \leftarrow$ ComputeAntidiffusionVelocity(ψ, u);
        mat2 $\leftarrow$ ComputeMatrix($\widetilde{u}$);
        $\psi \leftarrow$ MatrixMultiplication(mat2,ψ);
    end
end


Universiteit Utrecht