Adaptive multi-scale scheme based on numerical entropy production for conservation laws.

Mehmet Ersoy [1], Frédéric Golay [2] and Lyudmyla Yushchenko [3]

Journées NTM Porquerolles, 11-13 Juin, 2012

1. *Mehmet.Ersoy@univ-tln.fr*
2. *Frederic.Golay@univ-tln.fr*
3. *Lyudmyla.Yushchenko@univ-tln.fr*

Outline of the talk

- Modélisation de la dynamique de l'atmosphère et analyse mathématique (existence et stabilité).
- Analyse d'E.D.P. hyperbolique à gradient discontinu (Problème de Riemann, solveur de Godunov).
- Système hyperbolique avec termes sources (modélisation des écoulements en conduite fermée, modèle PFS, schéma numérique VF, schéma "well-balanced", entropique, schéma cinétique, VFRoe,...).
- Lois de conservation scalaire stationnaire et contrôle (analyse de solution stationnaire en temps fini, contrôle, ADM,...)
- Cinétique (sédimentation, Vlasov, Exner)

Outline

We focus on general non linear hyperbolic conservation laws

$$\begin{cases} \dfrac{\partial \boldsymbol{w}}{\partial t} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial x} = 0,\ (t,x) \in \mathbb{R}^+ \times \mathbb{R} \\ \boldsymbol{w}_0(x) = \boldsymbol{w}(0,x),\ x \in \mathbb{R}. \end{cases} \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ stands for the vector state and $\boldsymbol{f}$ the flux governing the physical description of the flow (Gas dynamics, fluid dynamics, road traffic, ...).

Numerical solution of (1) is a challenging problem since it is well known that solutions can and will breakdown at a finite time even if the initial data are smooth.

Serre. Systems of conservation laws (99-00) ;

We focus on general non linear hyperbolic conservation laws

$$\begin{cases} \dfrac{\partial \boldsymbol{w}}{\partial t} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial x} = 0,\ (t,x) \in \mathbb{R}^+ \times \mathbb{R} \\ \boldsymbol{w}_0(x) = \boldsymbol{w}(0,x),\ x \in \mathbb{R}. \end{cases} \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ stands for the vector state and $\boldsymbol{f}$ the flux governing the physical description of the flow (Gas dynamics, fluid dynamics, road traffic, ...).

Numerical solution of (1) is a challenging problem since it is well known that solutions can and will breakdown at a finite time even if the initial data are smooth.

Several attempts to the construction of

- high order numerical scheme and NOSC schemes $\rightarrow$ large complexity

have been proposed to try to capture efficiently discontinuous solutions.

Serre. Systems of conservation laws (99-00) ; LeVeque. Numerical methods for conservation laws (92) ;

We focus on general non linear hyperbolic conservation laws

$$\begin{cases} \dfrac{\partial \boldsymbol{w}}{\partial t} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial x} = 0,\ (t,x) \in \mathbb{R}^+ \times \mathbb{R} \\ \boldsymbol{w}_0(x) = \boldsymbol{w}(0,x),\ x \in \mathbb{R}. \end{cases} \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ stands for the vector state and $\boldsymbol{f}$ the flux governing the physical description of the flow (Gas dynamics, fluid dynamics, road traffic, ...).

Numerical solution of (1) is a challenging problem since it is well known that solutions can and will breakdown at a finite time even if the initial data are smooth.

Several attempts to the construction of

- high order numerical scheme and NOSC schemes $\rightarrow$ large complexity
- adaptive grids $\rightarrow$ well-known for FE methods and should provide an efficient framework for FV methods

have been proposed to try to capture efficiently discontinuous solutions.

Serre. Systems of conservation laws (99-00) ; LeVeque. Numerical methods for conservation laws (92) ; Puppo. ICOSAHOM, (02) ; Karni, Kurganov and Petrova. *J. Comput. Phys.* (02) ; Karni and Kurganov *Adv. Comput. Math.* (05), Puppo, preprint, (11).

Our aim is then to compute such solutions

- accurately
- fastly

i.e., we make use of :

Our aim is then to compute such solutions

- accurately
- fastly

i.e., we make use of :

- automatic refinement tools based on entropy production

Puppo. ICOSAHOM, (02) ; Golay. *CRAS*, (05)

Our aim is then to compute such solutions

- accurately
- fastly

i.e., we make use of :

- automatic refinement tools based on entropy production
- multi-time step technique

Puppo. ICOSAHOM, (02) ; Golay. *CRAS*, (05) Harten. *Comm. on Pure and Appl. Math.*, (95) ; Gottschlich-Müller and Müller. *Hyperbolic problems : Theory, numerics, applications*, (99) ; Cohen, Kaber, Muller, and Postel. *Mathematics of Computation*, (03) ; Altmann, Belat, Gutnic, Helluy, Mathis, Sonnendrücker, Angulo, and Hérard. *ESAIM*, (09) ; Puppo. preprint, (11)

Our aim is then to compute such solutions

- accurately
- fastly

i.e., we make use of :

- automatic refinement tools based on entropy production
- multi-time step technique

both we will save the computational time keeping the order of acuracy in Finite Volume framework

Puppo. ICOSAHOM, (02) ; Golay. *CRAS*, (05) Harten. *Comm. on Pure and Appl. Math.*, (95) ; Gottschlich-Müller and Müller. *Hyperbolic problems : Theory, numerics, applications*, (99) ; Cohen, Kaber, Muller, and Postel. *Mathematics of Computation*, (03) ; Altmann, Belat, Gutnic, Helluy, Mathis, Sonnendrücker, Angulo, and Hérard. *ESAIM*, (09) ; Puppo. preprint, (11)

The concept of entropy refers to the convex continuous entropy $s(\boldsymbol{w})$ of flux $\psi(\boldsymbol{w})$, for which one has

$$\frac{\partial s(\boldsymbol{w})}{\partial t} + \frac{\partial \psi(\boldsymbol{w})}{\partial x} \begin{cases} \equiv & 0 & \text{classical solution} \\ < & 0 & \text{weak solution (discontinuous solution)} \end{cases}$$

with

$$\psi' = s' f'$$

The concept of entropy refers to the convex continuous entropy $s(\boldsymbol{w})$ of flux $\psi(\boldsymbol{w})$, for which one has

$$\frac{\partial s(\boldsymbol{w})}{\partial t} + \frac{\partial \psi(\boldsymbol{w})}{\partial x} \begin{cases} \equiv & 0 & \text{classical solution} \\ < & 0 & \text{weak solution (discontinuous solution)} \end{cases}$$

with

$$\psi' = s' f'$$

It provides an *a posteriori* error indicator and a useful tool to construct automatic refinement.

Puppo. ICOSAHOM, (02) ; Puppo. preprint, (11)

How ?

We automatically construct

- low resolution (coarsest cells) whenever it fails to be zero for smooth flows
- high resolution (fine cells) whenever solution develops shocks

How ?

We automatically construct

- low resolution (coarsest cells) whenever it fails to be zero for smooth flows
- high resolution (fine cells) whenever solution develops shocks

But the CFL condition implies a time restriction since $\Delta t = \dfrac{h}{\alpha}$ where h is the meshsize of the finest cells

$\Longrightarrow$ CPU time increases rapidly !

How ?

We automatically construct

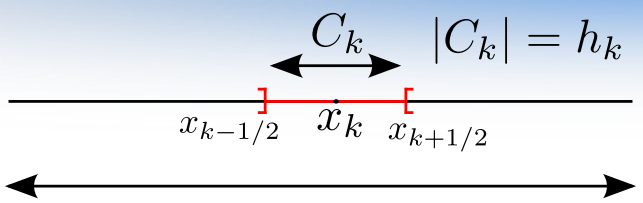
- low resolution (coarsest cells) whenever it fails to be zero for smooth flows
- high resolution (fine cells) whenever solution develops shocks

But the CFL condition implies a time restriction since $\Delta t = \dfrac{h}{\alpha}$ where h is the meshsize of the finest cells

$$\Longrightarrow \text{CPU time increases rapidly !}$$

Nonetheless, using multi time steps method one can save the CPU-time.

Outline



Integrating

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial x} = 0$$

$$\frac{\partial s(\boldsymbol{w})}{\partial t} + \frac{\partial \psi(\boldsymbol{w})}{\partial x} \leqslant 0$$

over each cells $C_k \times (t_n, t_{n+1})$ we obtain :

$$\int_{C_k} \boldsymbol{w}\,(t_{n+1}, x)\; dx - \int_{C_k} \boldsymbol{w}\,(t_n, x)\; dx + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{w}(t, x_{i+1/2})) - \boldsymbol{f}(\boldsymbol{w}(t, x_{i-1/2}))\, dt = 0$$

$$S = \int_{C_k} s(\boldsymbol{w}(t_{n+1}, x))\, dx - \int_{C_k} s(\boldsymbol{w}(t_n, x))\, dx + \int_{t_n}^{t_{n+1}} \psi(\boldsymbol{w}(t, x_{i+1/2})) - \psi(\boldsymbol{w}(t, x_{i-1/2}))\, d$$

where

$$S \leqslant 0.$$

It is called the density of entropy production.

Outline

For any discretisation of the finite volume formulation of the system, S fails to be zero even if the solution is smooth.

As a consequence, it can be used as a local error indicator for smooth flows and *a priori* reaches large negative value when a shock crosses the cell.

More precisely, one has

Theorem (Puppo 02)

$$S = \begin{cases} O(h^r) & \textit{for smooth flow} \\ O(1/h) & \textit{for discontinuous flow} \end{cases}$$

where r is the order of the Finite Volume approximation.

Choosing

$$\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) \text{ as a suitable approximation of } \frac{1}{\delta t_n}\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{w}(x_{k\pm 1/2}, s)\, ds,$$

noting $\delta t_n = t_{n+1} - t_n$ and

$$\boldsymbol{w}_k^n \simeq \frac{1}{h_k}\int_{C_k} \boldsymbol{w}\,(x, t_n)\; dx$$

we obtain :

$$\boldsymbol{w}_k^{n+1} = \boldsymbol{w}_k^n - \frac{\delta t_n}{h_k}\left(\boldsymbol{F}_{k+1/2}^n - \boldsymbol{F}_{k-1/2}^n\right) \;,$$

Choosing

$$\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) \text{ as a suitable approximation of } \frac{1}{\delta t_n}\int_{t_n}^{t_{n+1}} \boldsymbol{f}(\boldsymbol{w}(x_{k\pm 1/2}, s)\, ds,$$

noting $\delta t_n = t_{n+1} - t_n$ and

$$\boldsymbol{w}_k^n \simeq \frac{1}{h_k}\int_{C_k} \boldsymbol{w}\,(x, t_n)\; dx$$

we obtain :

$$\boldsymbol{w}_k^{n+1} = \boldsymbol{w}_k^n - \frac{\delta t_n}{h_k}\left(\boldsymbol{F}_{k+1/2}^n - \boldsymbol{F}_{k-1/2}^n\right)\;,$$

Similarly, one has

$$S_k^n = \frac{s_k^{n+1} - s_k^n}{\delta t_n} + \frac{\psi_{k+1/2}^n - \psi_{k-1/2}^n}{h_k}\;,$$

which is the numerical density of entropy production.

We note

$$\mathcal{P} = \sum_{n,k} S_k^n \delta t_n h_k$$

the total numerical entropy production.

How to define the numerical entropy production ?

For the Godunov scheme, we have $\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) = \boldsymbol{f}(\boldsymbol{w}_{k+1/2}^*)$ where $\boldsymbol{w}_{k+1/2}^*$ is the exact solution of Riemann problem with data $(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$.

Thus, the numerical entropy flux is defined as

$$\boldsymbol{w}_{k+1/2}^* = \boldsymbol{w}_{k+1/2}^*(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$$

i.e., one has

$$S_k^n = \frac{s_k^{n+1} - s_k^n}{\delta t_n} + \frac{\psi(\boldsymbol{w}_{k+1/2}^*) - \psi(\boldsymbol{w}_{k-1/2}^*)}{h_k} \leqslant 0.$$

How to define the numerical entropy production ?

For the Godunov scheme, we have $\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) = \boldsymbol{f}(\boldsymbol{w}_{k+1/2}^*)$ where $\boldsymbol{w}_{k+1/2}^*$ is the exact solution of Riemann problem with data $(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$.

Thus, the numerical entropy flux is defined as

$$\boldsymbol{w}_{k+1/2}^* = \boldsymbol{w}_{k+1/2}^*(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$$

i.e., one has

$$S_k^n = \frac{s_k^{n+1} - s_k^n}{\delta t_n} + \frac{\psi(\boldsymbol{w}_{k+1/2}^*) - \psi(\boldsymbol{w}_{k-1/2}^*)}{h_k} \leqslant 0.$$

For other numerical scheme for which the interface state $\boldsymbol{w}^*$ is not known, one can use

- either $\boldsymbol{w}_{k+1/2}^*$ as $\dfrac{\boldsymbol{w}_k^n + \boldsymbol{w}_{k+1}^n}{2}$

but $S_k^n \leqslant 0$ is not automatically satisfied.

How to define the numerical entropy production ?

For the Godunov scheme, we have $\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) = \boldsymbol{f}(\boldsymbol{w}_{k+1/2}^*)$ where $\boldsymbol{w}_{k+1/2}^*$ is the exact solution of Riemann problem with data $(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$.

Thus, the numerical entropy flux is defined as

$$\boldsymbol{w}_{k+1/2}^* = \boldsymbol{w}_{k+1/2}^*(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$$

i.e., one has

$$S_k^n = \frac{s_k^{n+1} - s_k^n}{\delta t_n} + \frac{\psi(\boldsymbol{w}_{k+1/2}^*) - \psi(\boldsymbol{w}_{k-1/2}^*)}{h_k} \leqslant 0.$$

For other numerical scheme for which the interface state $\boldsymbol{w}^*$ is not known, one can use

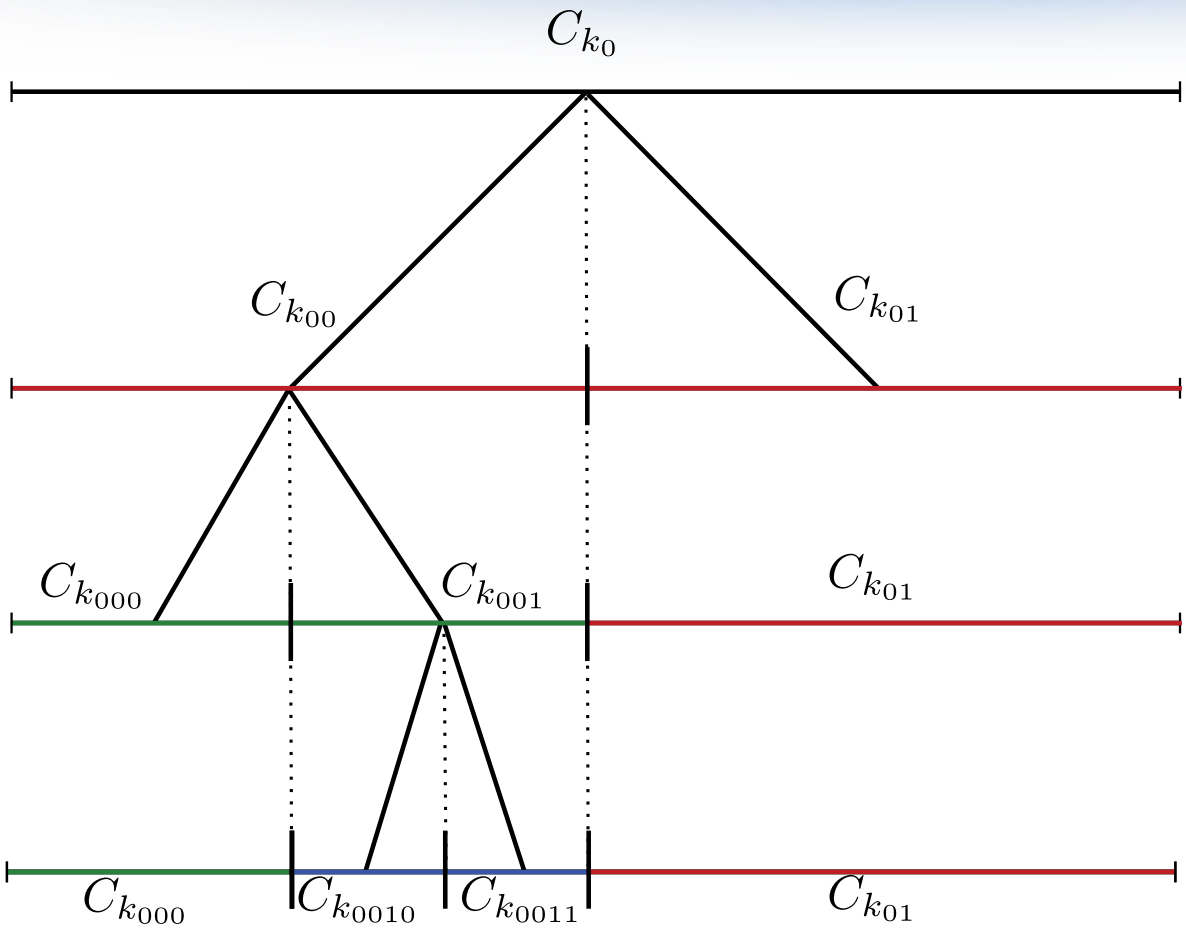
- either $\boldsymbol{w}_{k+1/2}^*$ as $\dfrac{\boldsymbol{w}_k^n + \boldsymbol{w}_{k+1}^n}{2}$
- or

$$\psi_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n) \approx \frac{1}{\delta t_n} \int_{t_n}^{t_{n+1}} \psi(\boldsymbol{w}(x_{k\pm1/2}, s)\, ds,$$

  (already used to define $\boldsymbol{F}_{k+1/2}(\boldsymbol{w}_k^n, \boldsymbol{w}_{k+1}^n)$).

but $S_k^n \leqslant 0$ is not automatically satisfied.

Outline



Let k_b the index which makes reference to the macro cell k and b a binary number which contains the hierarchical information of a sub-cell. The level of a sub-cell C_{k_b} is defined as the $\mathrm{length(b)} - 1$.

We first define a mesh refinement criterion $\bar{S}$. For instance, it can be the mean value over the domain Ω :

$$\bar{S} = \frac{1}{|\Omega|} \sum_{k_b} S^n_{k_b}$$

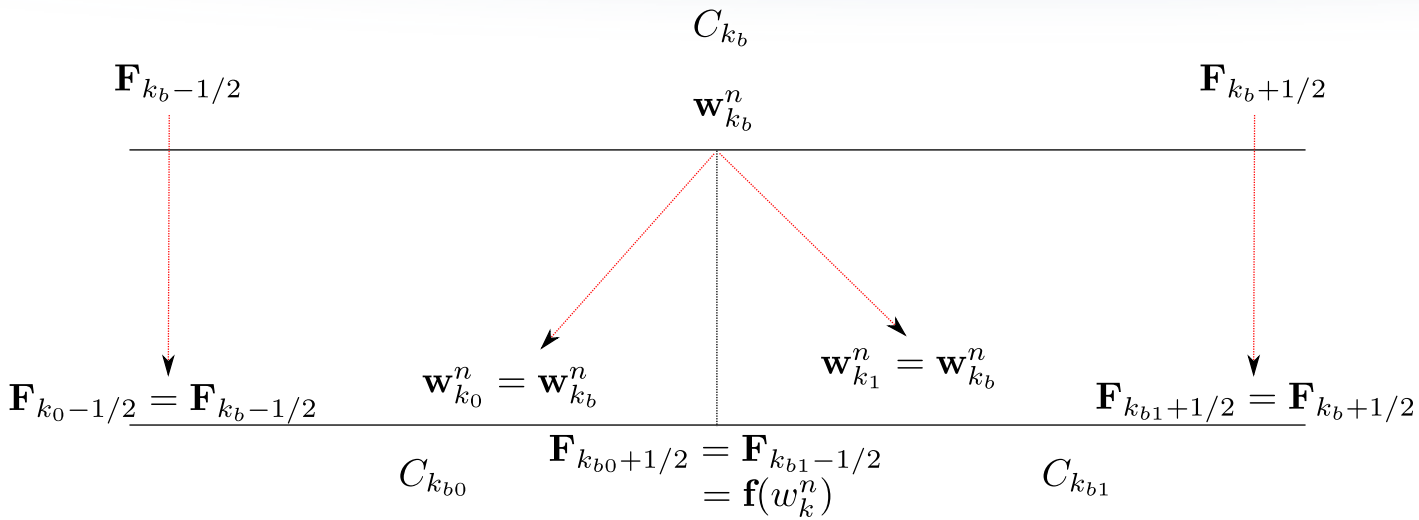
or simply a fixed small parameter.

We then define two coefficients α_{min} and α_{max}, which determine the ratio of numerical production of entropy leading to mesh refinement or mesh coarsening.

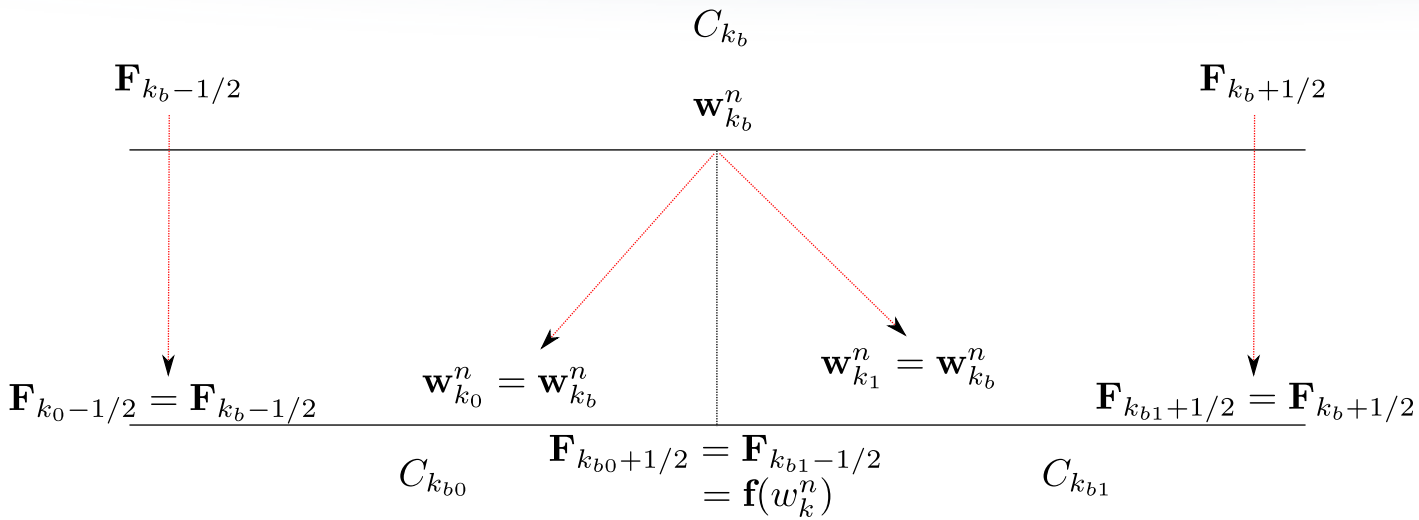
Thus, for each cell C_{k_b} :

- if $S^n_{k_b} > \bar{S}\alpha_{max}$, the mesh is refined and split into two sub-cells $C_{k_{b0}}$ and $C_{k_{b1}}$,
- if $S^n_{k_{b0}} < \bar{S}\alpha_{min}$ and $S^n_{k_{b1}} < \bar{S}\alpha_{min}$, the mesh is coarsened into a cell C_{k_b}.

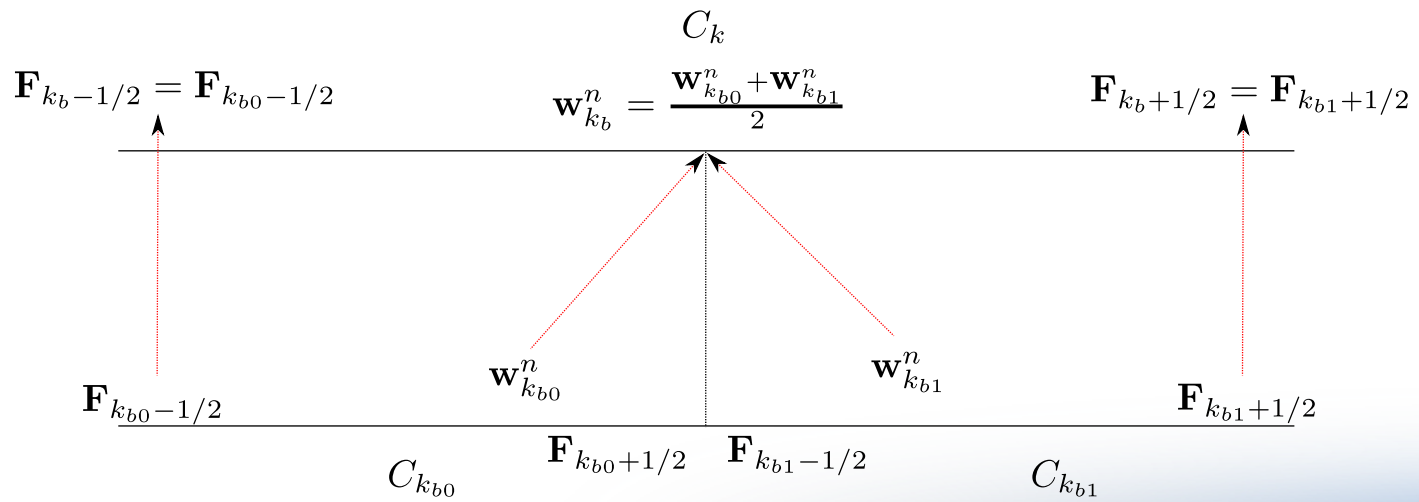
- If a cell C_{k_b} is split into two sub-cells $C_{k_{b0}}$ and $C_{k_{b1}}$, new subcells are initialized as follows :



- If a cell C_{k_b} is split into two sub-cells $C_{k_{b0}}$ and $C_{k_{b1}}$, new subcells are initialized as follows :



- if two sub-cells $C_{k_{b0}}$ and $C_{k_{b1}}$ is merged, the new cell C_{k_b} is initialized as follows :



For practical purpose we impose a maximal level.

But

the numerical scheme being stable under a CFL condition

$$\frac{\delta t_n}{\min_k h_k} \max \lambda(D_{\boldsymbol{w}} \boldsymbol{f}) < 1$$

$\Longrightarrow$ the less h_k the less δt_n

$\Longrightarrow$ time restriction

But

the numerical scheme being stable under a CFL condition

$$\frac{\delta t_n}{\min_k h_k} \max \lambda(D_{\boldsymbol{w}} \boldsymbol{f}) < 1$$

$\Longrightarrow$ the less h_k the less δt_n

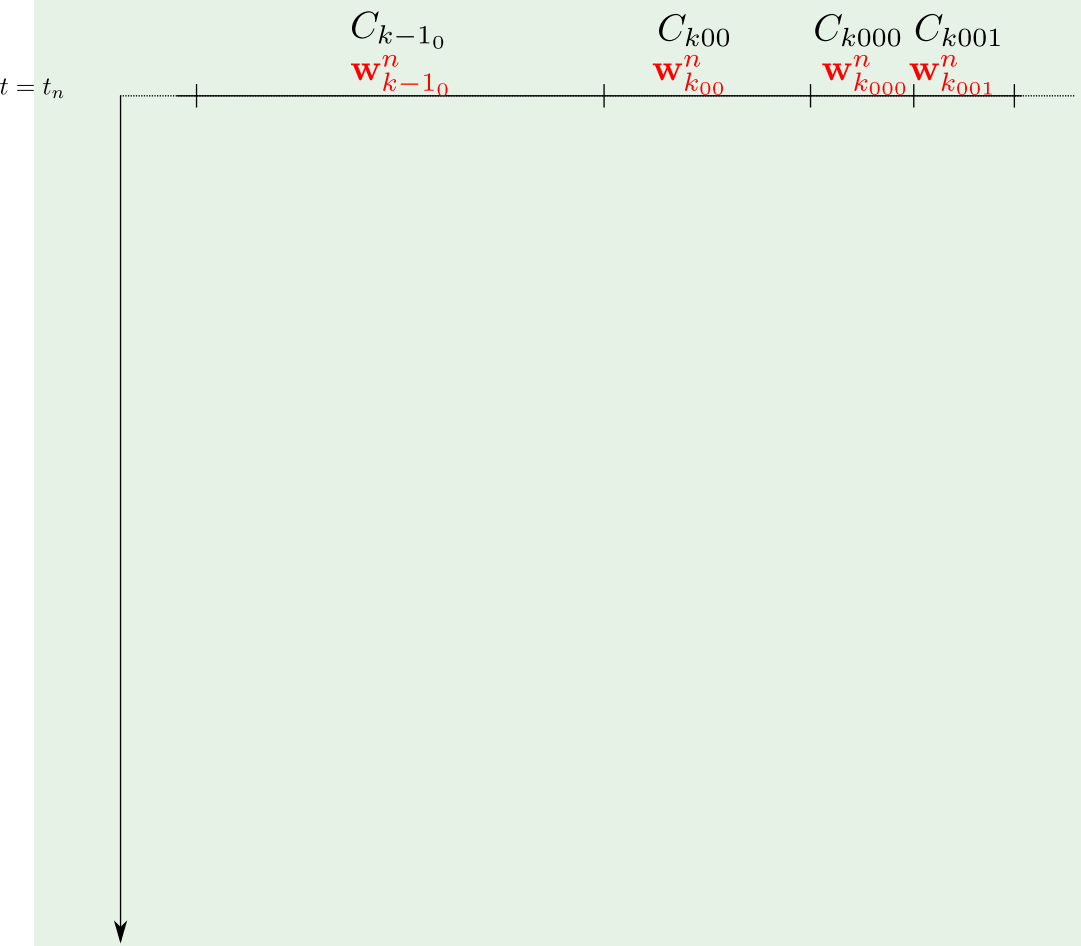
$\Longrightarrow$ time restriction

One can overpass using local time step method and save the CPU time

Outline

Compute first the numerical approximation on all finest cells (following an hierarchical algorithm) and update the value of $\boldsymbol{w}$ at time t_{n+1}. For the sake of clarity, let us note $\delta \boldsymbol{F}_k^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k, \boldsymbol{w}_{k+1}) = \left(\boldsymbol{F}_{k+1/2}^n(\boldsymbol{w}_k, \boldsymbol{w}_{k+1}) - \boldsymbol{F}_{k-1/2}^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k)\right)$

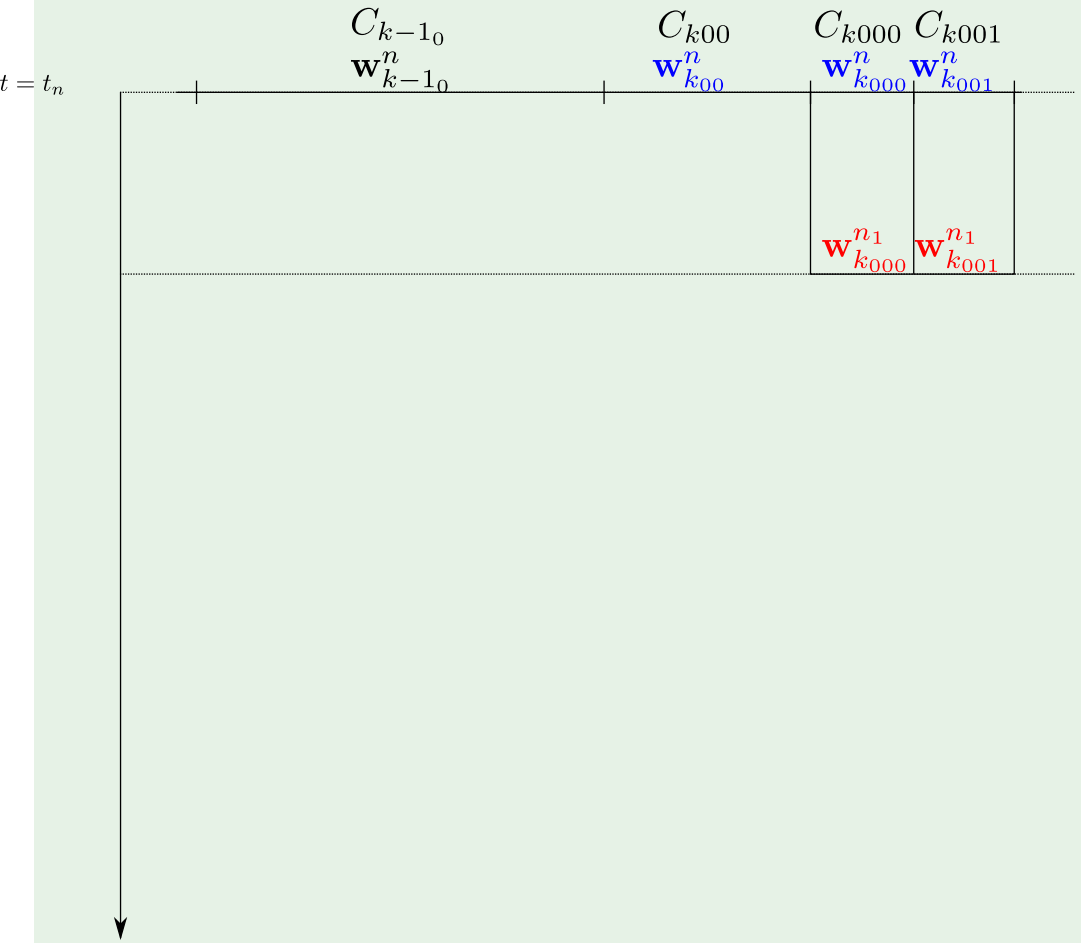
Example (local time step)



macro time step : $\Delta t_n = 2^N \delta t_n$
micro time step : δt_n
N : stands for the maximum level
here $N = 2$

Compute first the numerical approximation on all finest cells (following an hierarchical algorithm) and update the value of $\boldsymbol{w}$ at time t_{n+1}. For the sake of clarity, let us note $\delta \boldsymbol{F}_k^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k, \boldsymbol{w}_{k+1}) = \left(\boldsymbol{F}_{k+1/2}^n(\boldsymbol{w}_k, \boldsymbol{w}_{k+1}) - \boldsymbol{F}_{k-1/2}^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k)\right)$

Example (local time step)

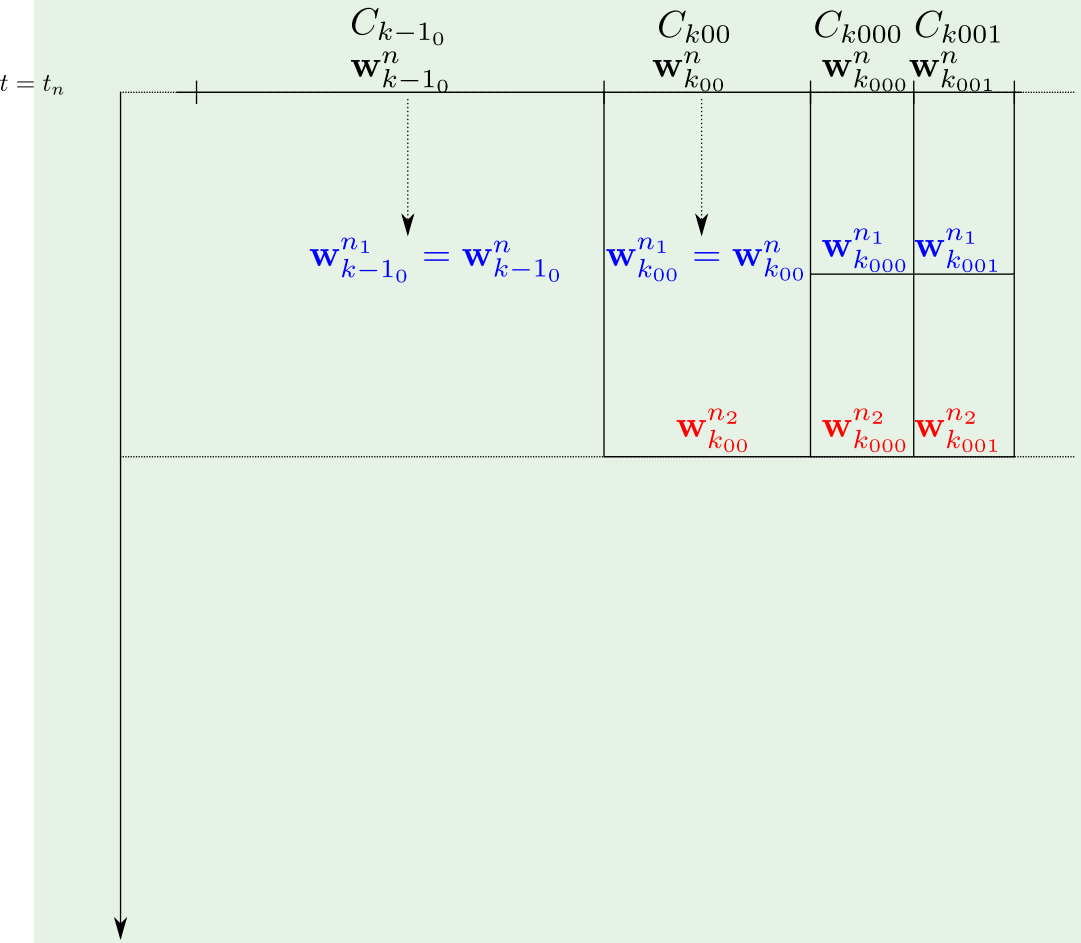


$$\boldsymbol{w}_{k000}^{n_1} = \boldsymbol{w}_{k000}^n - \frac{\delta t_n}{h_{k000}} \delta \boldsymbol{F}_{k000}^n(\boldsymbol{w}_{k00}, \boldsymbol{w}_{k000}, \boldsymbol{w}_{k001})$$

$$\boldsymbol{w}_{k001}^{n_1} = \boldsymbol{w}_{k001}^n - \frac{\delta t_n}{h_{k001}} \delta \boldsymbol{F}_{k000}^n(\boldsymbol{w}_{k000}, \boldsymbol{w}_{k001}, \boldsymbol{w}_{k+1_0}$$

Compute first the numerical approximation on all finest cells (following an hierarchical algorithm) and update the value of $\boldsymbol{w}$ at time t_{n+1}. For the sake of clarity, let us note $\delta \boldsymbol{F}_k^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k, \boldsymbol{w}_{k+1}) = \left(\boldsymbol{F}_{k+1/2}^n(\boldsymbol{w}_k, \boldsymbol{w}_{k+1}) - \boldsymbol{F}_{k-1/2}^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k)\right)$

Example (local time step)



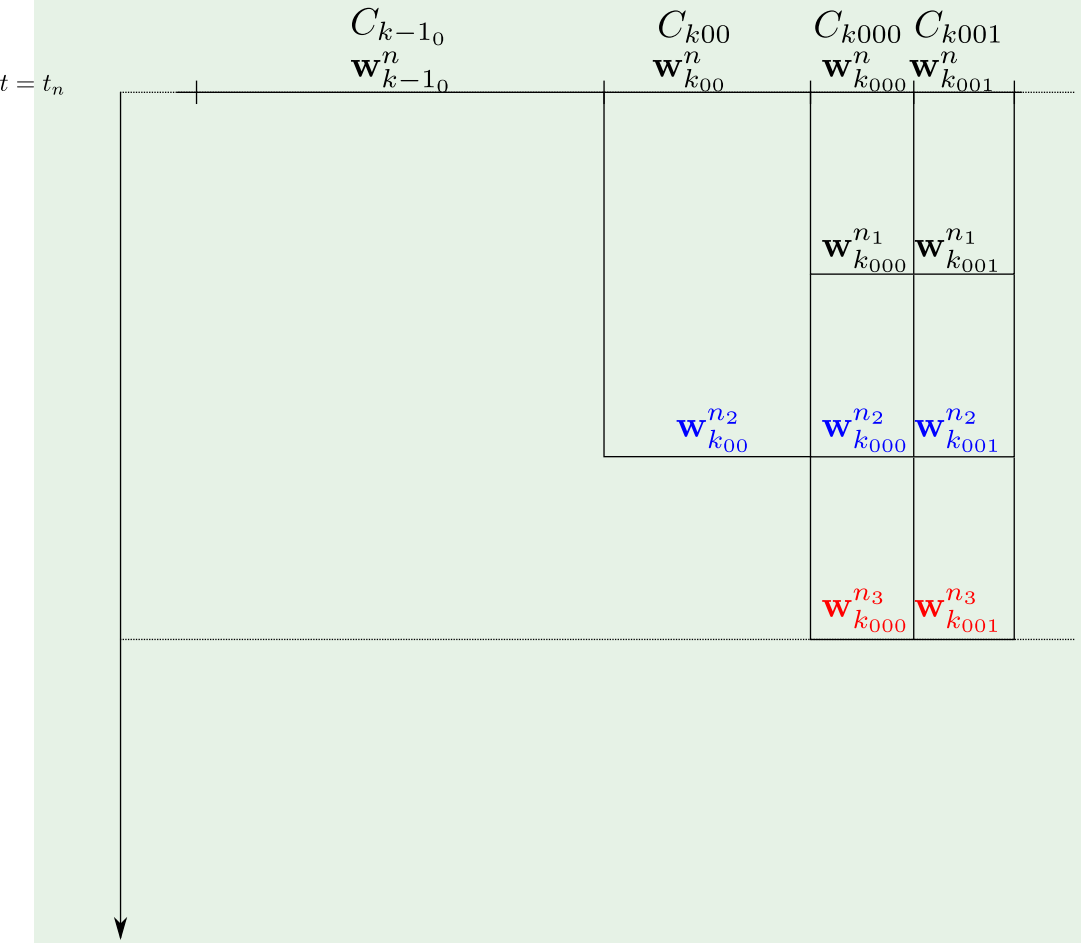
$$\boldsymbol{w}_{k00}^{n_2} = \boldsymbol{w}_{k00}^{n_1} - \frac{\delta t_n}{h_{k00}} \delta \boldsymbol{F}_{k000}^{n_1}(\boldsymbol{w}_{k-1_0}, \boldsymbol{w}_{k00}, \boldsymbol{w}_{k000})$$

$$\boldsymbol{w}_{k000}^{n_2} = \boldsymbol{w}_{k000}^{n_1} - \frac{\delta t_n}{h_{k000}} \delta \boldsymbol{F}_{k000}^{n_1}(\boldsymbol{w}_{k00}, \boldsymbol{w}_{k000}, \boldsymbol{w}_{k001})$$

$$\boldsymbol{w}_{k001}^{n_2} = \boldsymbol{w}_{k001}^{n_1} - \frac{\delta t_n}{h_{k001}} \delta \boldsymbol{F}_{k000}^{n_1}(\boldsymbol{w}_{k000}, \boldsymbol{w}_{k001}, \boldsymbol{w}_{k+1_0})$$

Compute first the numerical approximation on all finest cells (following an hierarchical algorithm) and update the value of $\boldsymbol{w}$ at time t_{n+1}. For the sake of clarity, let us note $\delta \boldsymbol{F}_k^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k, \boldsymbol{w}_{k+1}) = \left(\boldsymbol{F}_{k+1/2}^n(\boldsymbol{w}_k, \boldsymbol{w}_{k+1}) - \boldsymbol{F}_{k-1/2}^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k)\right)$

Example (local time step)

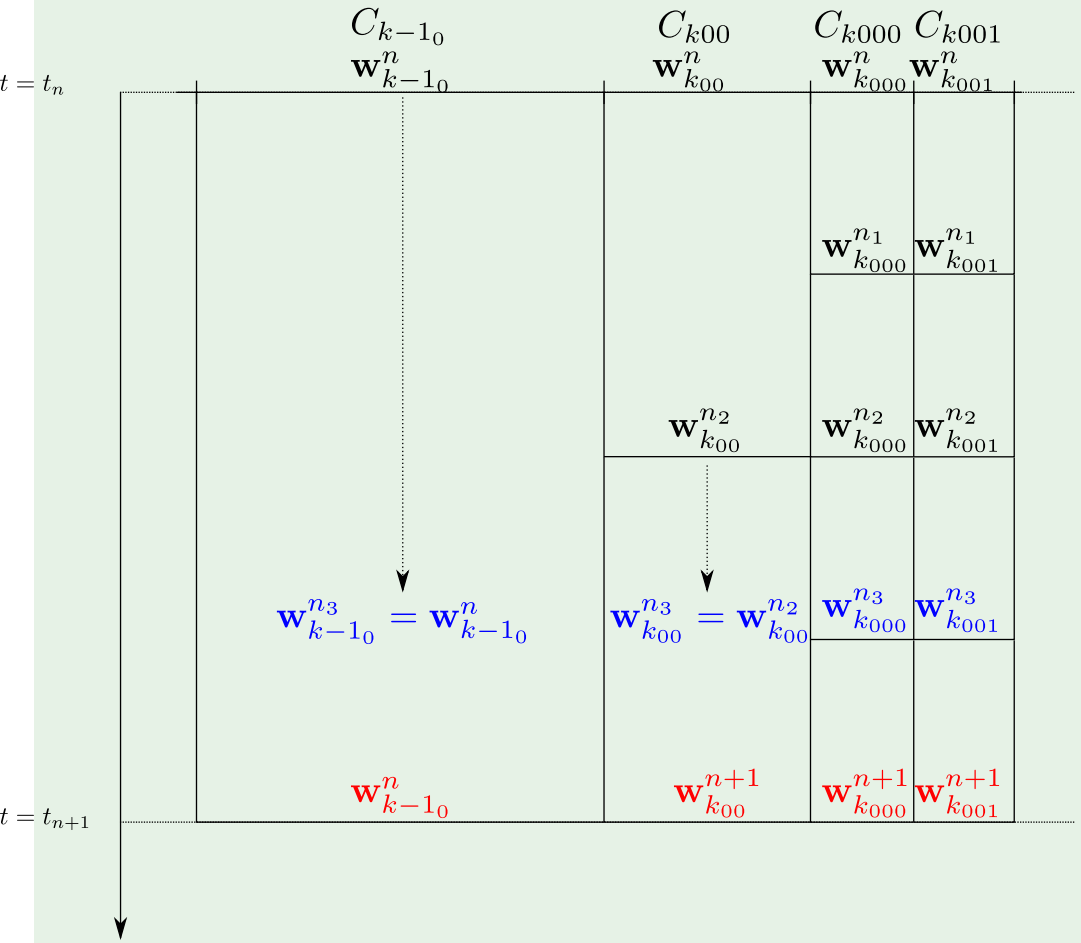


$$\boldsymbol{w}_{k000}^{n_3} = \boldsymbol{w}_{k000}^{n_2} - \frac{\delta t_n}{h_{k000}} \delta \boldsymbol{F}_{k000}^{n_2}(\boldsymbol{w}_{k00}, \boldsymbol{w}_{k000}, \boldsymbol{w}_{k001})$$

$$\boldsymbol{w}_{k001}^{n_3} = \boldsymbol{w}_{k001}^{n_2} - \frac{\delta t_n}{h_{k001}} \delta \boldsymbol{F}_{k000}^{n_2}(\boldsymbol{w}_{k000}, \boldsymbol{w}_{k001}, \boldsymbol{w}_{k+1}$$

Compute first the numerical approximation on all finest cells (following an hierarchical algorithm) and update the value of $\boldsymbol{w}$ at time t_{n+1}. For the sake of clarity, let us note $\delta \boldsymbol{F}_k^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k, \boldsymbol{w}_{k+1}) = \left(\boldsymbol{F}_{k+1/2}^n(\boldsymbol{w}_k, \boldsymbol{w}_{k+1}) - \boldsymbol{F}_{k-1/2}^n(\boldsymbol{w}_{k-1}, \boldsymbol{w}_k)\right)$

Example (local time step)



$$\boldsymbol{w}_{k-1_0}^{n+1} = \boldsymbol{w}_{k-1_0}^{n_3} - \frac{\delta t_n}{h_{k-1_0}} \delta \boldsymbol{F}_{k-1_0}^{n_3}(\boldsymbol{w}_{k-2_b}, \boldsymbol{w}_{k-1_0}, \boldsymbol{w}$$

$$\boldsymbol{w}_{k00}^{n+1} = \boldsymbol{w}_{k00}^{n_3} - \frac{\delta t_n}{h_{k00}} \delta \boldsymbol{F}_{k000}^{n_3}(\boldsymbol{w}_{k-1_0}, \boldsymbol{w}_{k00}, \boldsymbol{w}_{k000})$$

$$\boldsymbol{w}_{k000}^{n+1} = \boldsymbol{w}_{k000}^{n_3} - \frac{\delta t_n}{h_{k000}} \delta \boldsymbol{F}_{k000}^{n_3}(\boldsymbol{w}_{k00}, \boldsymbol{w}_{k000}, \boldsymbol{w}_{k001})$$

$$\boldsymbol{w}_{k001}^{n+1} = \boldsymbol{w}_{k001}^{n_3} - \frac{\delta t_n}{h_{k001}} \delta \boldsymbol{F}_{k000}^{n_3}(\boldsymbol{w}_{k000}, \boldsymbol{w}_{k001}, \boldsymbol{w}_{k+1}$$

One can easily

- increase the order of the time integration using Adams-Bashforth integration technique. For instance, for the order two, one has

$$\boldsymbol{w}_k(t_{n+1}) = \boldsymbol{w}_k(t_n) - \frac{3\delta t_n}{2h_k}\ \delta \boldsymbol{F}_k(t_n) + \frac{\delta t_n}{2h_k}\ \delta \boldsymbol{F}_k(t_{n-1})\ .$$

  which means we just have to store the value of the numerical fluxes at time t_{n-1} and t_n.
- include a second order MUSCL (Monotone Upstream-centered Schemes) reconstruction.

Outline

Numerical solutions [4] are computed in the case of the one-dimensional gas dynamics equations for ideal gas :

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial \rho u}{\partial x} = 0 \\ \dfrac{\partial \rho u}{\partial t} + \dfrac{\partial \left(\rho u^2 + p\right)}{\partial x} = 0 \\ \dfrac{\partial \rho E}{\partial t} + \dfrac{\partial \left(\rho E + p\right) u}{\partial x} = 0 \\ p = (\gamma - 1)\rho\varepsilon \end{cases} \quad \text{where} \quad \begin{array}{lcl} \rho(t,x) & : & \text{density} \\ u(t,x) & : & \text{velocity} \\ p(t,x) & : & \text{pressure} \\ \gamma := 1.4 & : & \text{ratio of the specific heats} \\ E(\varepsilon, u) & : & \text{total energy} \end{array}$$

where $E := \varepsilon + \dfrac{u^2}{2}$ (where ε is the internal specific energy).

Using the conservative variables $\boldsymbol{w} = (\rho, \rho u, \rho E)^t$, we classically define the convex continuous entropy

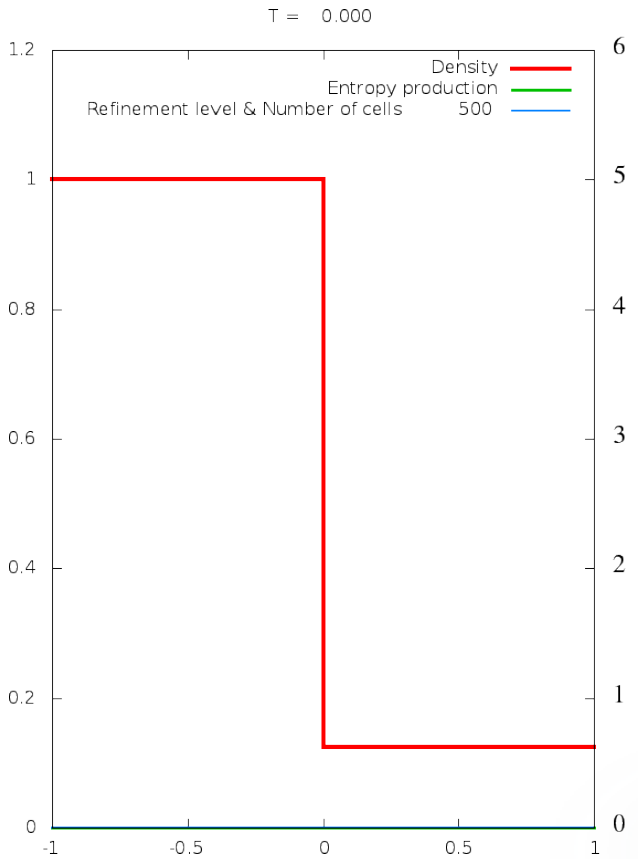
$$s(\boldsymbol{w}) = -\rho \ln\left(\frac{p}{\rho^\gamma}\right) \text{ of flux } \psi(\boldsymbol{w}) = u\, s(\boldsymbol{w}) \ .$$

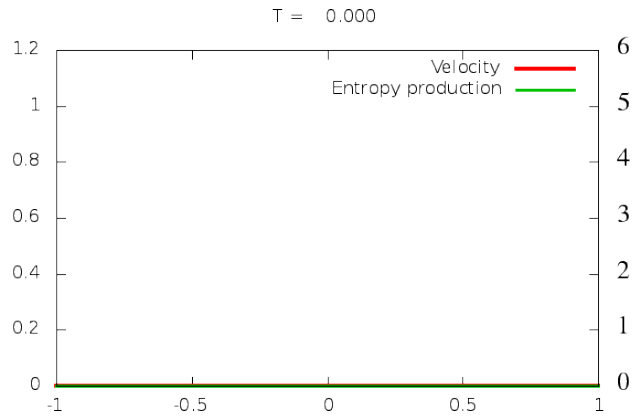
4. We have used the Godunov solver and displayed $-S$ instead of S. All tests have been performed on *Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz*

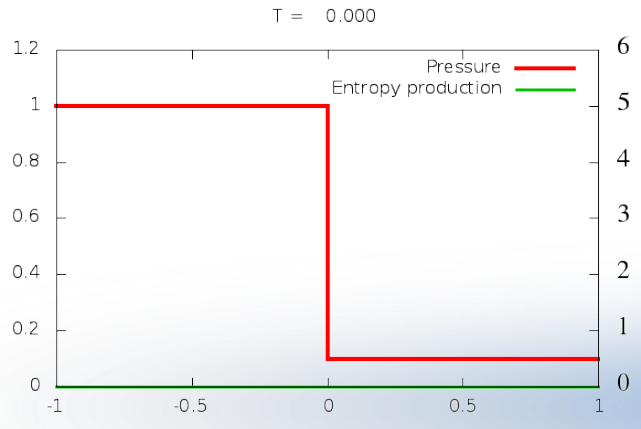
Parameters :

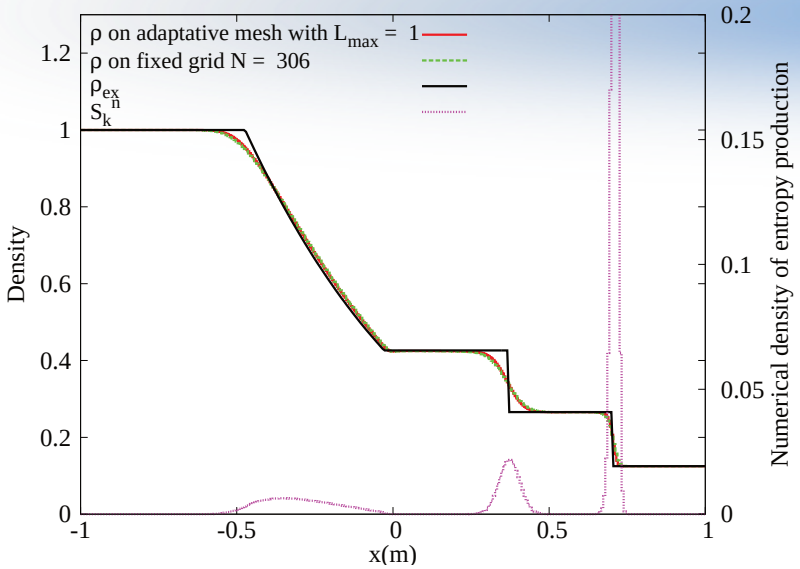
CFL	:	0.25,
Simulation time (s)	:	0.4,
Initial number of cells	:	200,
maximum level of refinement	:	$\mathcal{L}_{max}$,
mesh refinement parameter α_{max}	:	1,
mesh unrefinement parameter α_{min}	:	1,
mesh criterion $\bar{S}$	:	$1.0\,10^{-6}$

The initial conditions are :

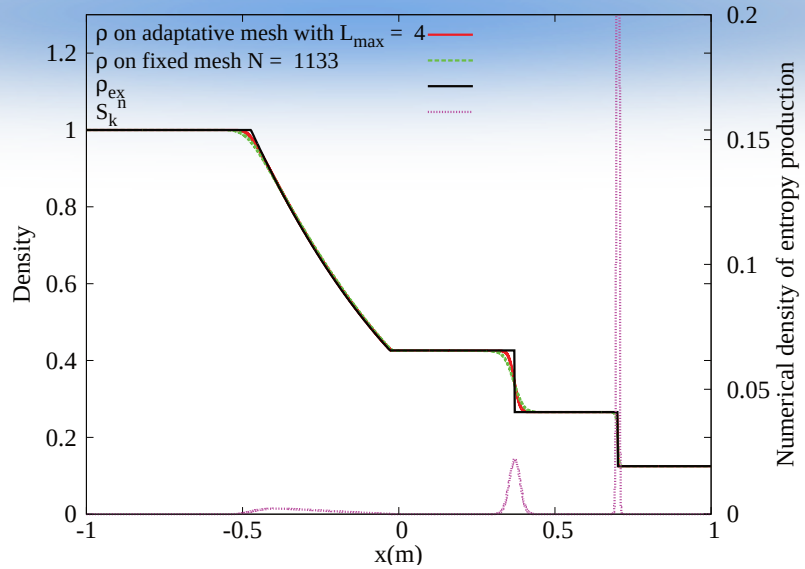




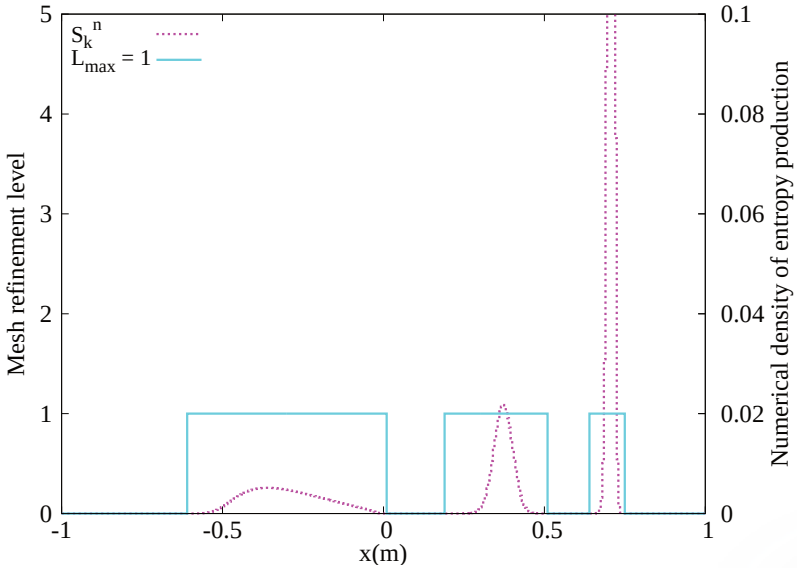




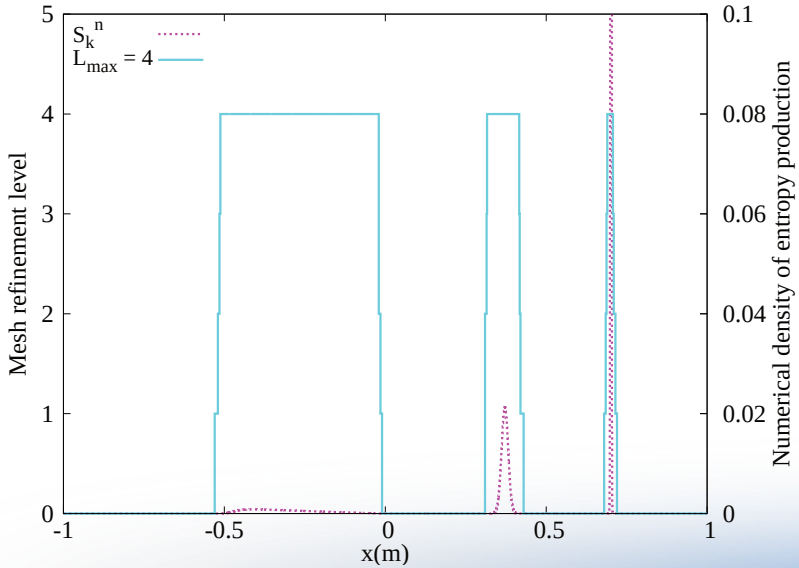
(a) Density–level $\mathcal{L}_{max} = 1$ at time $T = 0.4$.



(b) Density–level $\mathcal{L}_{max} = 4$ at time $T = 0.4$.

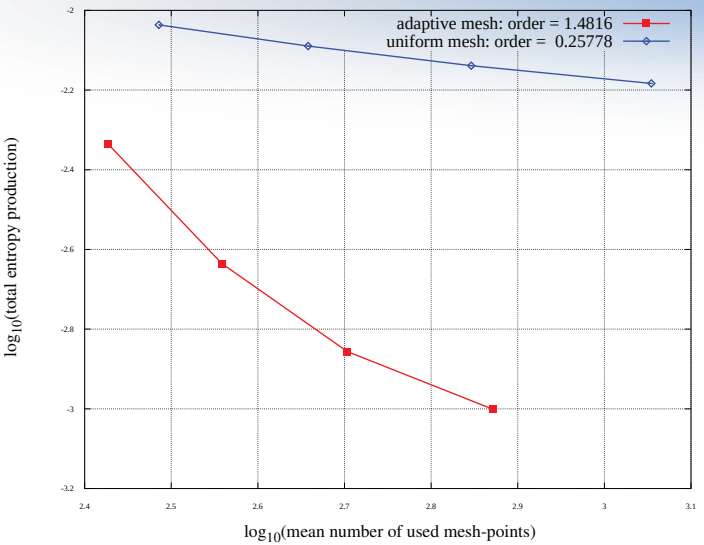


(c) Mesh level $\mathcal{L}_{max} = 1$ at time $T = 0.4$.

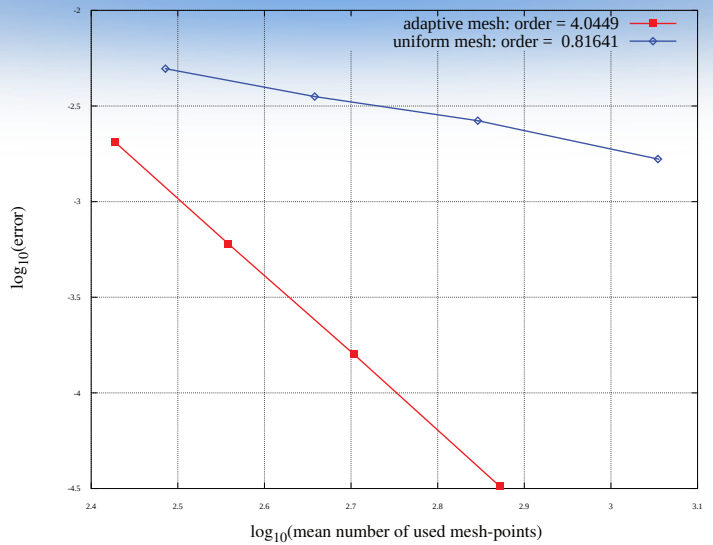


(d) Mesh level $\mathcal{L}_{max} = 4$ at time $T = 0.4$.

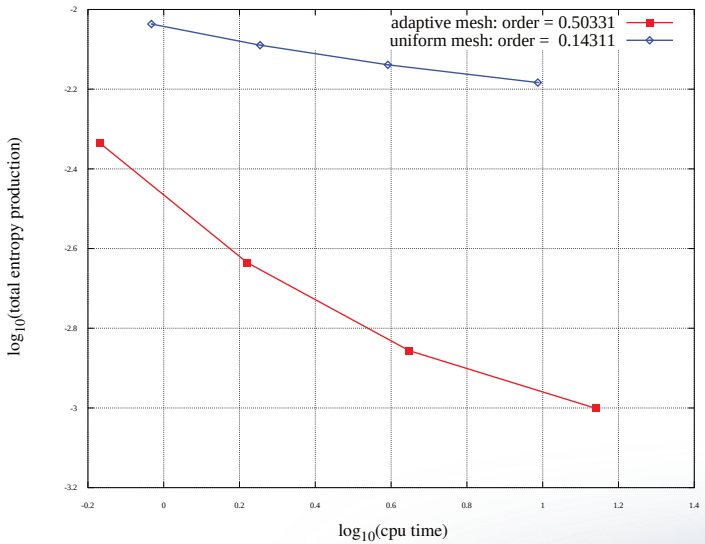
Accuracy and cpu-time



(e) Total entropy production.



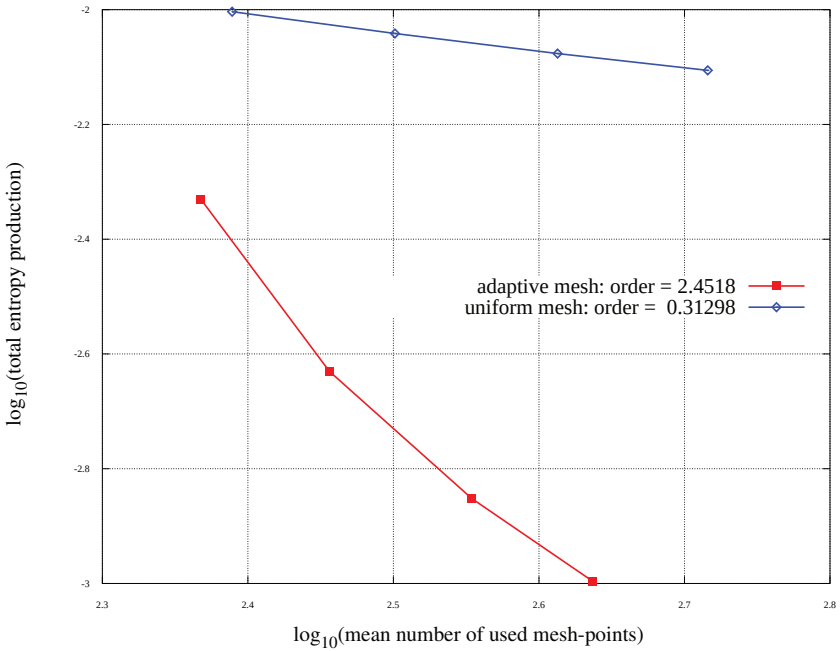
(f) Discrete norm $l_t^1 l_x^1$ on the density error.



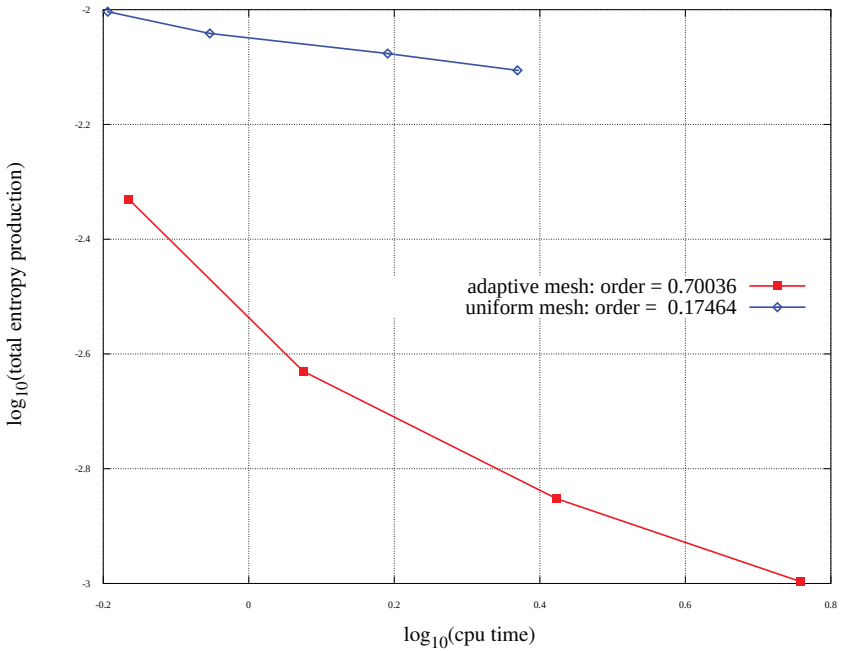
(g) Accuracy and cpu-time.

One can improve previous performance. For instance

$$\begin{aligned} \text{mesh refinement parameter } \alpha_{max} &: 1, \\ \text{mesh unrefinement parameter } \alpha_{min} &: 0.1, \end{aligned}$$



(h) Increasing numerical order.



(i) Reducing cpu-time.

Outline

Conclusion

Efficient adaptive numerical scheme for conservation laws based on numerical entropy production have been proposed.

Conclusion

Efficient adaptive numerical scheme for conservation laws based on numerical entropy production have been proposed.
Entropy production is used as

- a discontinuity detector, and
- a local error indicator

Conclusion

Efficient adaptive numerical scheme for conservation laws based on numerical entropy production have been proposed.
Entropy production is used as

- a discontinuity detector, and
- a local error indicator

in a multi-time framework.
Thus, one can

- keep the same order of accuracy and,
- save the computational cost.

Conclusion and works in progress

Efficient adaptive numerical scheme for conservation laws based on numerical entropy production have been proposed.
Entropy production is used as

- a discontinuity detector, and
- a local error indicator

in a multi-time framework.
Thus, one can

- keep the same order of accuracy and,
- save the computational cost.

Analysis of the scheme with others solvers such as Godunov, Rusanov, HLL, kinetic, VFRoe, ...) for hyperbolic systems with/without source terms

- convergence
- stability with respect to mesh refinement α_{max} and unrefinement α_{max},

in particular, properties such as

- steady states
- entropy inequalites

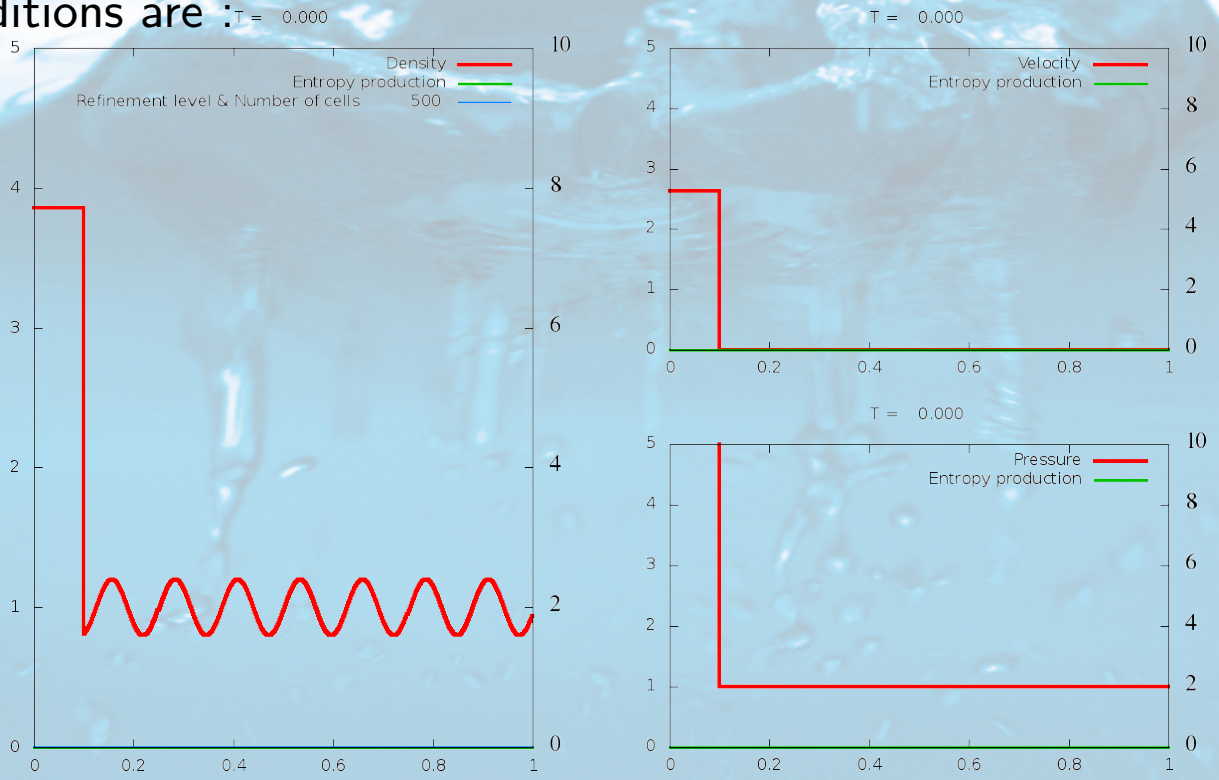
and 2D/3D algorithm


Thank you
for your
attention

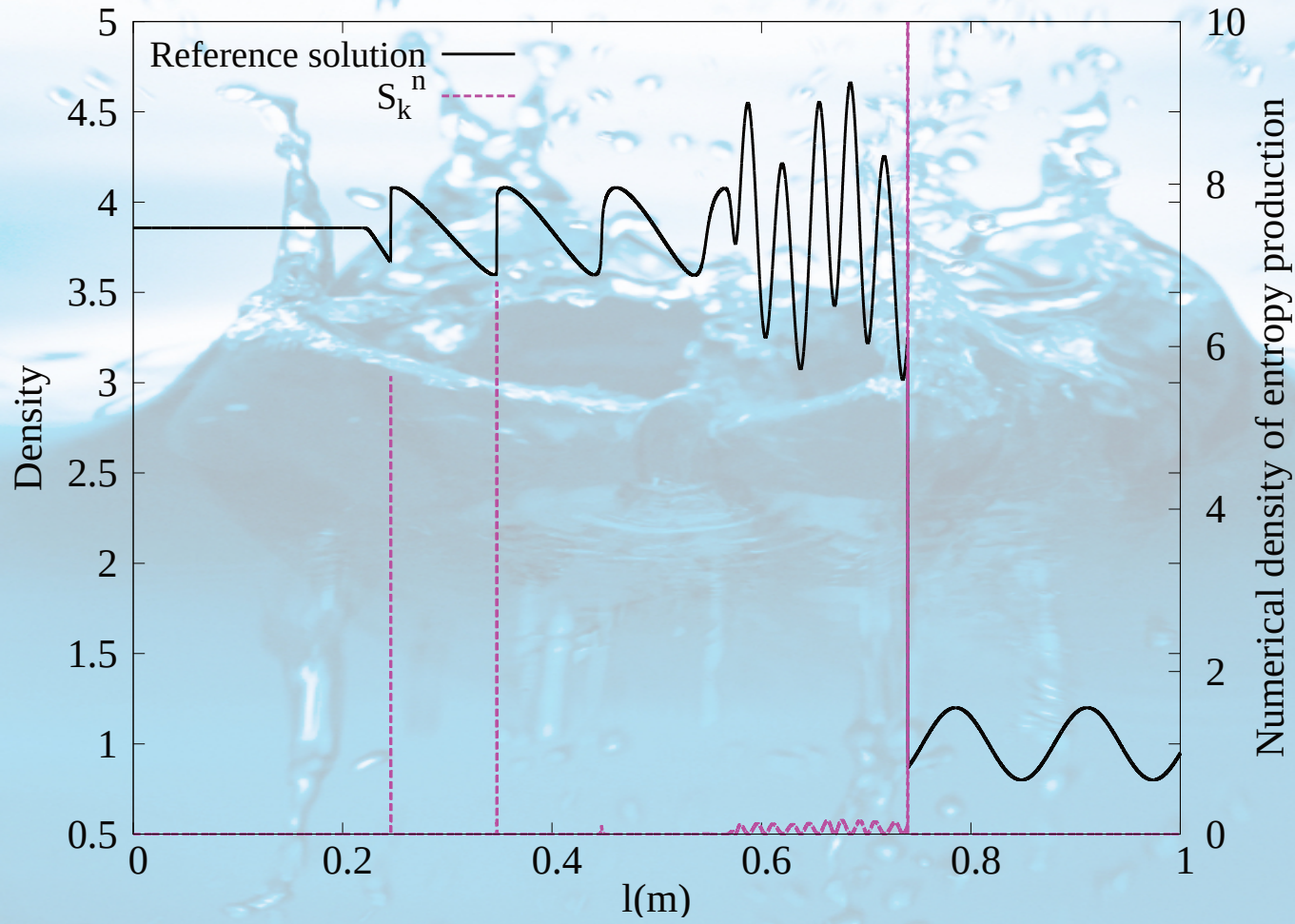
Parameters :

CFL	:	0.219,
Simulation time (s)	:	0.18,
Initial number of cells	:	500,
maximum level of refinement	:	$\mathcal{L}_{max} = 4$,
mesh refinement parameter α_{max}	:	0.001,
mesh unrefinement parameter α_{min}	:	0.05,
mesh criterion $\bar{S}$	:	provided by formula $\bar{S} = \frac{1}{\|\Omega\|} \sum_{k_b} S^n_{k_b}$.

The initial conditions are :



Reference solution : RK2+MUSCL on a fixed grid with $N = 20000$



multi-time step AB1 : $AB1$,
constant-time step AB1 : $AB1_c$,
multi-time step AB1 and MUSCL reconstruction : $AB1_{muscl}$,
constant-time step AB1 and MUSCL reconstruction : $AB1_{c,muscl}$,
multi-time step AB2 and MUSCL reconstruction : $AB2$,
constant-time step AB2 and MUSCL reconstruction : $AB2_c$,
constant-time step RK2 and MUSCL reconstruction : $RK2$.

	$\mathcal{P}$	$\|\rho - \rho_{ref}\|_2$	CPU	N meshpoints
$AB1$	0.287	$7.77\,10^{-3}$	104.35	2015
$AB1_c$	0.288	$7.54\,10^{-3}$	166.52	2032
$AB1_{muscl}$	0.285	$4.19\,10^{-3}$	89.45	1765
$AB1_{c,muscl}$	0.287	$2.32\,10^{-3}$	157.07	1874
$AB2$	0.284	$3.91\,10^{-3}$	96.15	1815
$AB2_c$	0.287	$3.34\,10^{-3}$	158.40	1890
$RK2$	0.286	$2.32\,10^{-3}$	277.10	1874

multi-time step AB1 : $AB1$,
constant-time step AB1 : $AB1_c$,
multi-time step AB1 and MUSCL reconstruction : $AB1_{muscl}$,
constant-time step AB1 and MUSCL reconstruction : $AB1_{c,muscl}$,
multi-time step AB2 and MUSCL reconstruction : $AB2$,
constant-time step AB2 and MUSCL reconstruction : $AB2_c$,
constant-time step RK2 and MUSCL reconstruction : $RK2$.

	$\mathcal{P}$	$\|\rho - \rho_{ref}\|_2$	CPU	N meshpoints
$AB1$	0.287	$7.77\,10^{-3}$	104.35	2015
$AB1_c$	0.288	$7.54\,10^{-3}$	166.52	2032
$AB1_{muscl}$	0.285	$4.19\,10^{-3}$	89.45	1765
$AB1_{c,muscl}$	0.287	$2.32\,10^{-3}$	157.07	1874
$AB2$	0.284	$3.91\,10^{-3}$	96.15	1815
$AB2_c$	0.287	$3.34\,10^{-3}$	158.40	1890
$RK2$	0.286	$2.32\,10^{-3}$	277.10	1874

foreach $i \in \{1, 2^N\}$ **do**

Let j be the biggest integer such that 2^j divides i

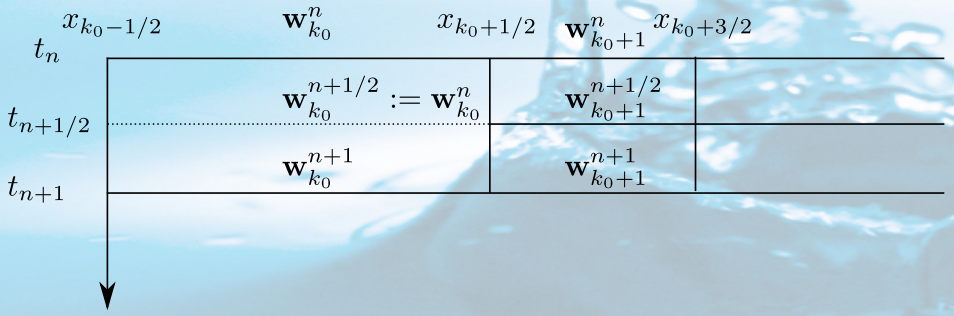
foreach *interface* $x_{k+1/2}$ *such that* $\mathcal{L}_{k+1/2} \geqslant N - j$ **do**

1. compute the integral of $\boldsymbol{F}_{k+1/2}(t)$ on the time interval $2^{N-\mathcal{L}_{k+1/2}}\delta t_n$,
2. distribute $\boldsymbol{F}_{k+1/2}(t_n)$ to the two adjacent cells,
3. update only the cells of level greater than $N - j$.

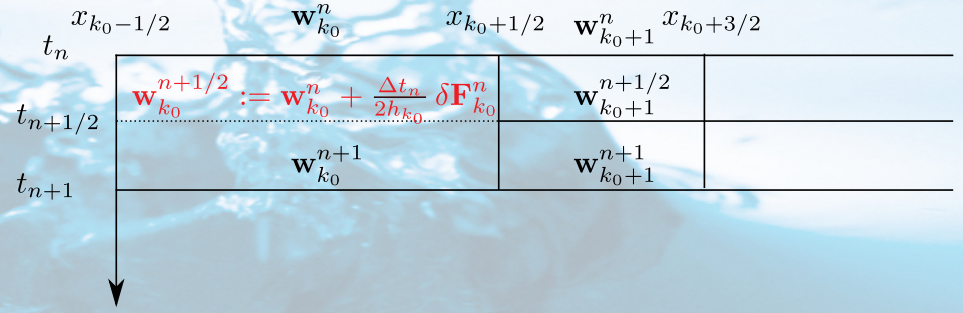
end

end

Assume that for every $k \leqslant k_0$, all cells C_k are of level $\mathcal{L}_{k_0}$ and for every $k > k_0$, all cells C_k are of level $\mathcal{L}_{k_0} + 1$.



(j) Osher and Sanders projection.



(k) Tang and Warnecke projection.

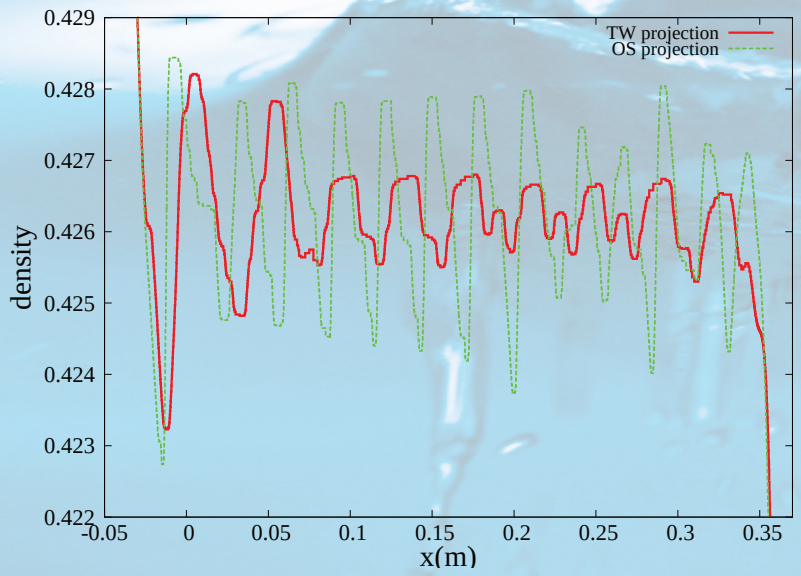
TW : locally consistant but not globally conservative between two adjacent cells of different levels (increase the CPU time)
OS : locally non consistant and not conservative (simple projection)

Indeed, we have without smoothing effect :

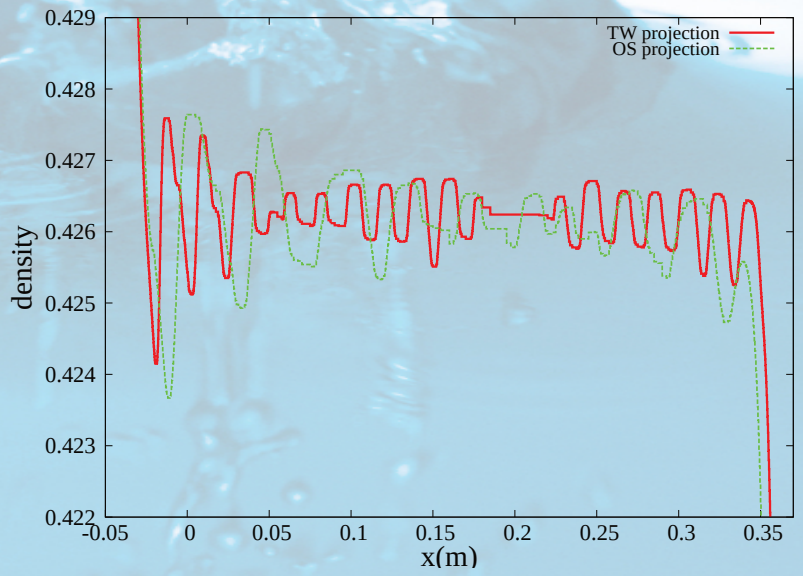
$$\begin{aligned} \text{cpu-time for TW projection} &= 330.27\ (s), \\ \text{cpu-time for OS projection} &= 85.02\ (s), \end{aligned}$$

and with smoothing effect :

$$\begin{aligned} \text{cpu-time for TW projection} &= 288.21\ (s), \\ \text{cpu-time for OS projection} &= 80.04\ (s). \end{aligned}$$



(l) Without smoothing effect.



(m) With smoothing effect.

Computation of the flux for the second order MUSCL reconstruction :
Refinement :

$$\begin{cases} \boldsymbol{w}_{k_b 0}^n = \boldsymbol{w}_{k_b}^n - \dfrac{h_k}{4} \dfrac{\partial \boldsymbol{w}_{k_b}^n}{\partial x} , \\ \boldsymbol{w}_{k_b 1}^n = \boldsymbol{w}_{k_b}^n + \dfrac{h_k}{4} \dfrac{\partial \boldsymbol{w}_{k_b}^n}{\partial x} . \end{cases}$$