A 3D vertex centered CENO scheme for advection

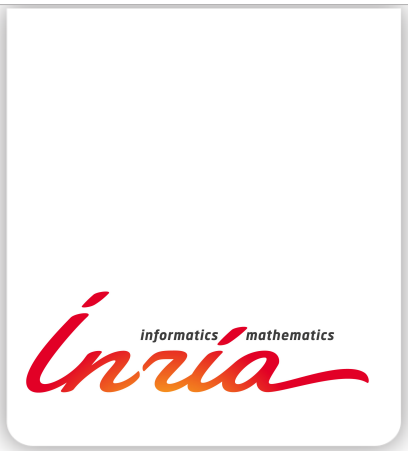
Matthieu Gschwend

Un schéma CENO 3D centré sommet pour l'advection

Matthieu Gschwend

RESEARCH REPORT
N° 9415
2021
Project-Team Ecuador

ISSN 0249-6399 ISRN INRIA/RR--9415--FR+ENG



Un schéma CENO 3D centré sommet pour l'advection

Matthieu Gschwend *

Équipe-Projet Ecuador

Rapport de recherche n° 9415 — 2021 — 37 pages

Résumé : Nous présentons un schéma volumes-finis avec une reconstruction quadratique de la solution pour une équation de type hyperbolique 3D sur des maillages non-structurés formés de tétraèdres. Ce schéma est une adaptation au contexte centré-sommet du schéma CENO. Ce travail a été financé par l'Agence Nationale de la Recherche dans le cadre du projet NORMA, contrat ANR-19-CE40-0020-01.

Mots-clés : Volumes finis, Ordre élevé

* Université Côte d'Azur,INRIA-Ecuador,B.P.93,06902 Sophia-Antipolis, FRANCE,matthieu.gschwend@inria.fr

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

A 3D vertex centered CENO scheme for advection

Abstract: We present a finite-volume scheme with a quadratic reconstruction of the solution for a 3D hyperbolic equation on structured unstructured meshes formed of tetrahedra. This scheme is an adaptation of the CENO scheme to the vertex-centered context.This work was supported by the ANR project NORMA under grant ANR-19-CE40-0020-01.

Key-words: Finite Volume, higher order

Table des matières

1 Introduction

We are interested in the derivation of dissipative higher-order methods for CFD. By dissipative, we mean that we shall address advective models for which the option of no numerical dissipation can cause an important lack of robustness. Due to this, most CFD, particularly compressible CFD, approximations will involve dissipation devices, mainly based on the introduction of Riemann solvers as initiated by Godunov.

Most high-order approximation schemes like Discontinous Galerkin [4, 7, 8, 15], ENO [3, 9, 11, 13] or distributive schemes [2] use kth-order interpolation or reconstruction and are k-exact. Interpolation and reconstruction are two approximation mappings, the errors of which need to be analysed. Most analyses are inspired by the Bramble-Hilbert principle, saying that an approximation which is exact for kth-order polynomial is a $(k+1)$th-order accurate approximation. Demonstrations can be found in the fundamental paper [6]. Later, when considering reconstruction-based schemes, see [10], the authors refered to the Taylor series. A re-visitation in [1] establishes the link with [6].

The goal of the presented research is to study a 3D version of a third-order accurate and a fourth-order accurate Central Essentially Non-Oscillatory central (CENO) approximation, based on a quadratic polynomial reconstruction (then CENO2). These scheme are inspired by the CENO proposition of Groth and coworkers, see [12]. We prealably transpose them to a vertex formulation, as in [5]. The extension to 3D is described of an advection model. In order to save CPU, we examine a new reconstruction molecule for this scheme.

The efficiency of a high order scheme does not depend solely of its order of convergence. In the case of finite-volume based approximations, we have to evaluate for a given mesh the number of degrees of freedom, the reconstruction/interpolation effort, and the number of finite volume fluxes to be evaluated. By comparison with second-order schemes, e.g. MUSCL, vertex CENO needs reconstructions, and, of most cost, high-order integrated fluxes at finite volume interfaces. By comparison with a discontinuous Galerkin of degree three (for fourth order), an DG3 element contains 20 d.o.f. and need four high-order integrated fluxes. The DG3 element can be split in 27 DG1 or P_1 elements. Counting 6 elements for a vertex, this shows that a vertex CENO of degree 3 will have 27/6=4.44 d.o.f. on the same DG3 mesh, but with 14*27/6=62 fluxes to evaluate. We shall see that this last figure is still increased due to the special geometry of vertex interfaces. Then it appears that the Achille neel of vertex CENO is the computational cost of the fluxes. The smaller number of d.o.f. can be advantageous in case of implicit time advancing. The treatment of discontinuity can also be easier that for DG. And for the user, a given mesh of n vertices will not lead to much more than n d.o.f.

We start this report with a presentation of the basic scheme, a discussion of reconstruction molecules, of quadratic and cubic polynomial reconstruction, of fluc evaluation, of time advancing, and the report is completed with a few numerical experiments.

2 Basic scheme

2.1 Continuous advection model

We consider the following advection model :

$$\frac{\partial u}{\partial t}(x,y,z,t) + \nabla \cdot \mathbf{f}(u(x,y,z,t)) = 0 \tag{1}$$

where initial and boundary conditions are :

$$\begin{cases} u(x,y,z,0) = u_0(x,y,z) \\ u(x,y,z,t) = \Phi(x,y,z,t) \quad \text{for } (x,y,z) \in \partial\Omega \end{cases} \tag{2}$$

where $(x,y,z) \in \Omega$ with Ω an open subset of $\mathbb{R}^3$. For $t \geq 0, u : \Omega \times \mathbb{R}_+^* \longrightarrow \mathbb{R}$. The flux $\mathbf{f}(u(x,t)) = (f_1(u(x,t)), f_2(u(x,t)), f_3(u(x,t)))^t$ with $f_i \in \mathbb{C}^1(\mathbb{R},\mathbb{R})$.

In this report, we restrict to *constant-velocity advection* :

$$\mathbf{f}(u(x,t))) = (V_1(u(x,t)), V_2(u(x,t), V_3(u(x,t)))^t \text{ with } (V_1,V_2,V_3) \in \mathbb{R}^3. \tag{3}$$

The velocity $\mathbf{V} = (V_1,V_2,V_3)^t$ being given. Then the boundary conditions for well-posedness are :

$$u(x,y,z,t) = \Phi(x,y,z,t) \quad \text{for } \mathbf{V}\cdot\mathbf{n} < 0 \in \partial\Omega \tag{4}$$

où $\mathbf{n}$ est la normale extérieure locale de la cellule C_i.

The computational domain Ω is partitioned in dual vertex-centered finite-volume cells C_i around vertices i, limites by triangular facets IgG, connecting mid-edge, centroid of face and centroid of tetrahedra, Figure 1. As a result, the cell is made of a certain number of minitets

FIGURE 1 – Dual cells from tetrahedras. Left, view of the boundary of cell C_i (i : highest vertex in the figure) with a tetrahedron $ijkl$. The cell C_i around vertex i is made of all subtetrahedras $igGI$ where I is the middle of a neiboring edge, g the centroid of a tetrahedron face ijk containing iI, G the centroid of a tetrahedron containing the same face ijk. Right, intersection of the boundaries of cell i and cell j, on the facet of which is integrated the flux between cell i and cell j.

included in the neigboring tetrahedras, each tetrahedron of the mesh being split into 24 of these minitets. For the finite volume formulation, we integrate the above equation 1 on cell C_i and apply the Green formula :

$$\frac{d}{dt}\int_{C_i} u(x,y,z,t)dxdt + \int_{\partial C_i} \mathbf{f}(u(x,y,z,t))\cdot\mathbf{n}ds = 0$$

which becomes :

$$\frac{d}{dt}\int_{C_i} u(x,y,z,t)dxdt + \sum_{k\in\mathcal{V}(i)} \int_{\partial C_i\cap\partial C_k} \mathbf{f}(u(x,y,z,t))\cdot\mathbf{n}ds = 0. \tag{5}$$

where $\mathcal{V}(i)$ is the set of vertices j directly neighboring i, *i.e.* sharing an edge ij.

2.2 k-exact finite volume

The field u can be projected in a spatially-discrete field $\bar{u} = (u_1,..,u_{ncell})$ defined by :

$$\bar{u}_i = \int_{C_i} u\mathrm{d}\Omega.$$

Let $\mathcal{P}_j(\bar{v})$ a set of polynomials defined on Ω, with index j possibly identical to i. A central assumption is the *k-exact reconstruction assumption* :

$$\textit{For any polynomial } v \textit{ of degree } k, \ \mathcal{P}_j \textit{ reconstructs exactly } v. \tag{6}$$

We have $\mathcal{P}_j(\bar{u}) = u$ for any j. Due to the k-exactness of the reconstruction we have :

$$\int_0^T \int_{C_i} \frac{\partial \mathcal{P}_j(\bar{u})}{\partial t} \mathrm{d}\Omega \mathrm{d}t + \int_0^T \int_{\partial C_i} \mathbf{f}(\mathcal{P}_j(\bar{u})) \cdot \mathbf{n} \mathrm{d}\sigma \mathrm{d}t = 0. \tag{7}$$

In order to build a spatially discrete problem with a spatially discrete unknown, we introduce a vector $v_i(t)$ such that :

$$\bar{v}_i(0) = \int_{C_i} u(\mathbf{x}, 0) \mathrm{d}\Omega \quad \forall\ i. \tag{8}$$

Vector $v_i(t)$ is the semi-discrete solution of the k-exact reconstruction finite volume :

$$\text{Find} \quad v_i(t) \text{ such that (8) holds together with :}$$

$$\int_0^T \int_{C_i} \frac{\partial \mathcal{P}_i(\bar{v})}{\partial t} \mathrm{d}\Omega \mathrm{d}t + \int_0^T \int_{\partial C_i} \mathbf{f}(\mathcal{P}_i(\bar{v})) \cdot \mathbf{n} \mathrm{d}\sigma \mathrm{d}t = 0.$$

If we add the following *conservation assumption* concerning the reconstruction $\mathcal{P}$:

$$\int_{C_i} \mathcal{P}_i(\bar{v}) \mathrm{d}\Omega = meas(C_i) \bar{v}_i \tag{9}$$

We get a finite-volume formulation :

$$\begin{aligned} &\text{Find} \quad v_i(t) \text{ such that (8) holds together with :} \\ &\int_0^T \int_{C_i} \frac{\partial \bar{v}_i}{\partial t} \mathrm{d}\Omega \mathrm{d}t + \int_0^T \int_{\partial C_i} \mathbf{f}(\mathcal{P}_i(\bar{v})) \cdot \mathbf{n} \mathrm{d}\sigma \mathrm{d}t = 0. \end{aligned} \tag{10}$$

This finite volume spatial discretization is k-exact, that is satisfies :

If u is a polynomial of degree k satisfying the continuous equation (1), then its mean on cells $\bar{u}$ satisfies the discrete equation (10) and is its solution if (10) has a unique solution.

3 Two-step reconstruction

We assume that we have build around any cell i a *reconstruction molecule*

$$\mathcal{N}(i) = \bigcup_j cell j$$

where the set of the j' is a sufficiently large set of (direct and non-direct) neighbors of i, to fic the ideas, $card(\mathcal{N}(i)) > 30$. Let us choose a reference point $\mathbf{c}_i$ inside the reconstruction molecule. We can directly reconstruct around cell C_i the polynomial $\mathcal{P}$

$$\mathcal{P}_i(\mathbf{x}) = \overline{u_{0,i}} + \sum_{|\alpha| \leq k} c_{i,\alpha} [(\mathbf{x} - \mathbf{c}_i)^\alpha - \overline{(\mathbf{x} - \mathbf{c}_i)^\alpha}]$$

by solving a constrained least square problem. It consists of minimizing the least squares deviation between the polynomial $\mathcal{P}$ and the means under the constraint of respecting the mean on cell C_i :

$$(c_{i,\alpha})_\alpha \ = \ Arg \min \textstyle\sum_{j \in \mathcal{N}(i)} (\overline{\mathcal{P}_{i,j}} - \overline{U_j})^2 \ \text{ subject to } \ \overline{\mathcal{P}_{i,i}} = \overline{U_i}$$

where $\overline{\mathcal{P}_{i,j}}$ holds for the mean of $\mathcal{P}_i$ on cell C_j. Many choices are possible for designing the least square fonctional, like adding weights for taking into account the distance between i and j. These options will influence the accuracy, but not the convergence order, as far as the k-exactness of the reconstruction is maintained :

Lemma : *we keep k-exactness if the constrained optimum is replaced by the following two-step reconstruction :*

$$\begin{array}{l} \text{Step 1 : } (c_{i,\alpha})_\alpha \ = \ Arg\min \sum_{j\in N(i)\cup\{i\}} (\overline{\mathcal{P}_{i,j}} - \overline{U_j})^2 \\ \text{Step 2 : } \mathcal{P}_i = \mathcal{P}_i - \overline{\mathcal{P}_{i,i}} + \overline{U_i}.\square \end{array} \tag{11}$$

Indeed, for a degree k polynomial U_1, the first step will exactly find this polynomial, then $-\overline{\mathcal{P}_{i,i}} + \overline{U_i} = 0$ and the second step will not change it.

The two-step reconstruction allows two different cells i_1 and i_2 to have the same reconstruction molecule containing both. Choosing a unique reference point c_i, we can apply Step 1 once for both. Then solely Step 2 is done for each cell i_1 and i_2. In practice, the same reconstruction molecule can be used for all subcells.

4 Defining the reconstruction molecule

4.1 Algorithmics

A first algorithm, the *algorithm for centered molecules*, gathers around any cell its reconstruction molecule :
Generating centered cluster of N cells around around each vertex :
-Identifying direct neaighbouring cells
-Adding these to the molecule
-Identifying direct neighbours of the above cells
-Continue untill the targeted number of cells is reached

With this algorithm, the number of reconstruction is equal to the total number N of cells.

The second algorithm, the *partition-based algorithm*, builds a partition into macromolecules and links to any cell the macromolecule containing it :

With this algorithm, the number of reconstruction will be the number of partition, which will be of the order of $N/30$ if each partition contains about 30 cells. Each element of the partition represents a molecule.
Unlike the Algo 1, in this case molecules are disjoint.
First step
Build a molecule from a randomly selected vertex :
-Adding successive layers of direct neighbours (excluding cells already in another molecule), until the targeted number of cells is reached.
-Stating a new molecule from a vertex that has not already been linked in to a molecule.
Second step
Remaining cells that have not been included in a molecule are added to the closest neighbouring molecule.

Algorithm 1 Generating centered molecules

```
OUTPUT : Listing of N neighbours for each cells
{Start by indentifying directs neighbours}
voisDir=VoisinsDirect(mesh)
repeat
  {Loop over vertices}
  for i = 0 to nbVertex do
    {nbVois[i] represents the number of cells in the molecule i}
    nb1 = nbVois[i]
    if nb1 < N then
      {tracking the cells whintin to the molecule }
      for j = 0 to nb1 do
        nb2 = voisDir[resultat[i][j]].size()
        {Loop on the cell's direct neighbours}
        for k to nb2 do
          if (nbVois[i]<N) and (voisDir[resultat[i][j]][k] ∉ resultat[i]) and
          (voisDir[resultat[i][j]][k]≠i) then
            {adding the direct neighbour if it is not already included in the molecule }
            resultat[i][nbVois[i]]=voisDir[resultat[i][j]][k]
            nbVois[i]=nbVois[i]+1
          end if
        end for
      end for
    end if
  end for
until nbVois[i] ≠ N, ∀ i
return result
```

Algorithm 2 Domain partition

```
{Start by indentifying directs neighbours}
voisDir=VoisinsDirect(mesh)
Step (1)
{Loop over vertices}
for i = 0 to nbVertex do
  {Creating a molecule from a cell that is not already included in a molecule}
  if appartenanceAMacro[i]=0 then
    macroIntermediaire[0] = i ;{adding the cell i to macro}
    appartenanceAMacro[i]=1{the added cell is flaged}
    while stopReccurence do
      {Loop over cells in the molecule}
      for j = 0 to macroIntermediaire.size() do
        VoisinsDirectIntermediaires = listeVoisinsDirectID[macroIntermediaire[j]] ;
        {Loop over cell's direct neighbours}
        for k = 0 to VoisinsDirectIntermediaires.size() do
          {Adding the neighbour to the molecule provided it is not already included in a
          molecule and if the targeted number of cells in a molecule is not reached}
          if (appartenanceAMacro[VoisinsDirectIntermediaires[k]]==0) and
          (macroIntermediaire.size()<nbCellParMol) and (VoisinsDirectIntermediaires[k] ∉
          macroIntermediaire) then
            macroIntermediaire.pushback(VoisinsDirectIntermediaires[k])
          end if
        end for
      end for
      {When the targeted number of cells in the molecule is reached, the molecule
      macroIntermediaire is added to the list of molecules}
      {Inportant to note : the targeted number of cells nbCellParMol can not always be
      reached, in such cases a molecule can not be created, meaning the cell i needs to be
      flaged back to 0 }
      if (macroIntermediaire.size() = nbCellParMol) or (macroIntermediaire.size()=nbCellParMol)
      then
        stopReccurence = false
      end if
    end while
    if (macroIntermediaire.size()=nbCellParMol) then
      listeDesMacro.pushback(macroIntermediaire) { création d'une nouvelle macro}
    end if
  end if
end for
Step (2) : Allocating renaming lone cells to adjacent molecule
while nbCellAglom < nbVertex do
  for i = 0 to nbVertex do
    if appartenanceAMacro[i] = 0 then
      {In case the cell i is not already included in a molecule it is added to the smallest
      neighbouring molecule}
      for j = 0 to listeVoisinsDirectID[i].size() do
        if appartenanceAMacro[idDuVoisin] ≠0 then
          if nbCellMacroLaPlusPetite > nbCellMacroVoisin then
            idMacroLaPlusPetite = idMacroVoisin {Identifying the smallest molecule}
          end if
        end if
      end for
      {Adding the cell to that molecule }
      listeDesMacro[idMacroLaPlusPetite -1].pushback(i) ;
      nbCellAglom = nbCellAglom +1 ;{Increasing the number of cells that have been
      allocated to molecules}
    end if
  end for
end while
```

4.2 Examples

The two approaches are illustrated in Figures 6 and 3.

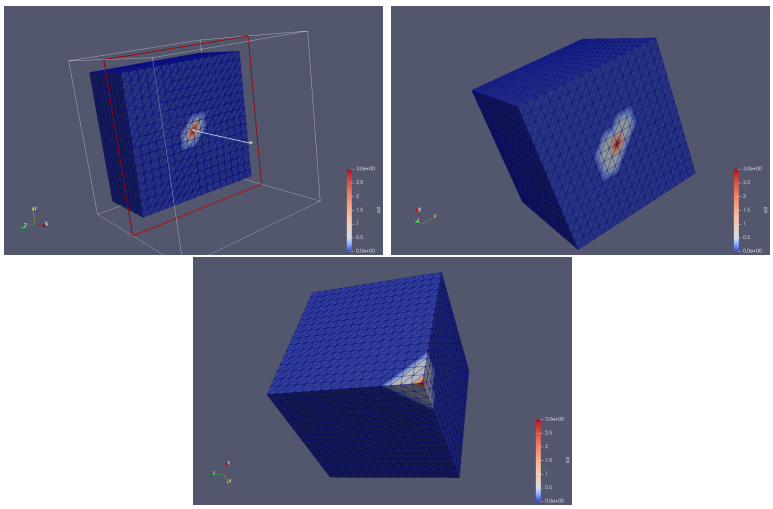


FIGURE 2 – Neighboring-based reconstruction. Top : a reconstruction molecule around its central cell. Bottom : two boundary cases.

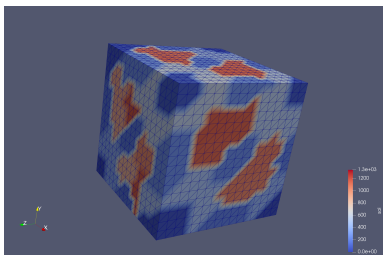


FIGURE 3 – Partition-based reconstruction. Examples of reconstruction molecules.

5 Polynomial reconstruction

5.1 Quadratic polynomial reconstruction

On each control cell C_i and at each time step, we try to approximate the solution $u\left(x,y,z,t^n\right) = u(x,y,z)^n$ by constructing a quadratic polynomial P_i^n.

It is necessary that the mean values of the polynomial P_i^n (which we write $\bar{P}_i^{i,n}$) and the mean values of the solution (which we write $\bar{u}^{i,n}$) on cell C_i are equal.

This condition is written : $\bar{P}_i^{i,n} = \bar{u}^{i,n}$, with :

$$\begin{cases} \bar{P}_i^{i,n} = \dfrac{1}{Vol\,(C_i)} \displaystyle\int_{C_i} P_i^n(x,y,z)dxdydz \\ \bar{u}^{i,n} = \dfrac{1}{Vol\,(C_i)} \displaystyle\int_{C_i} u^n(x,y,z)dxdydz \end{cases} \tag{12}$$

The polynomial P_i^n to be reconstructed on the dual cell C_i is defined by the following relation :

$$P_i^n = \bar{u}^{i,n} + \sum_{\alpha \in I} c_{i,\alpha}^n \left[(X - X_{o,i})^\alpha - \overline{(X - X_{o,i})}^{i,\alpha} \right] \tag{13}$$

where :

$$\begin{cases} \overline{(X - X_{o,i})}^{i,\alpha} = \frac{1}{Vol(C_i)} \int_{C_i} (X - X_{o,i})^\alpha \, dxdydz, \\ I \text{ is the set of multi-indices : } I = \{\alpha = (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{N}^3, \text{ s.t. } |\alpha| = \alpha_1 + \alpha_2 + \alpha_3 \in [1,2]\}, \\ (X - X_{o,i})^\alpha = (x - x_{o,i})^{\alpha_1} (y - y_{o,i})^{\alpha_2} (z - z_{o,i})^{\alpha_3}, \text{ where } (x_{o,i}, y_{o,i}, z_{o,i}) \text{ is the centroid of } C_i. \end{cases}$$

We order the multi-indices as follows :

$$\begin{cases} \alpha = 1 \to (\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 0) \\ \alpha = 2 \to (\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 0) \\ \alpha = 3 \to (\alpha_1 = 0, \alpha_2 = 0, \alpha_3 = 1) \\ \alpha = 4 \to (\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 0) \\ \alpha = 5 \to (\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 1) \\ \alpha = 6 \to (\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 1) \\ \alpha = 7 \to (\alpha_1 = 2, \alpha_2 = 0, \alpha_3 = 0) \\ \alpha = 8 \to (\alpha_1 = 0, \alpha_2 = 2, \alpha_3 = 0) \\ \alpha = 9 \to (\alpha_1 = 0, \alpha_2 = 0, \alpha_3 = 2) \end{cases}$$

To build the polynomial P_i^n on the cell C_i, we need to define the molecule M_i formed from the neighboring cells of C_i, so as to take enough values of the solution around i to reconstruct the coefficients of the polynomial. In the present case we need to calculate the 9 coefficients of the polynomial, so our molecule M_i must contain at least 9 cells (in addition to C_i).

The 9 unknown coefficients $c_{i,\alpha}^n$ of the polynomial reconstruction are calculated by the Least Squares method, i.e. such a way that the distance L_2 between the means on cells C_k, of the polynomial $\bar{P}_i^{i,n}$ associated with the cell C_i, and of the solution $\bar{u}^{k,n}$, be minimal for cells C_k included in the molecule M_i.

$$H_i = \sum_{C_{k \neq i} \subset M_i} \left(\bar{P}_i^{k,n} - \bar{u}^{k,n} \right)^2 \tag{14}$$

where $\bar{P}_i^{k,n}$ represents the mean of polynomial P_i^n in cell C_k :

$$\bar{P}_i^{k,n} = \frac{1}{Vol\,(C_k)} \int_{C_k} P_i^n(x,y)dxdydz. \tag{15}$$

Through the construction of C_k, we get :

$$H_i = \sum_{C_{k \neq i} \subset M_i} \left(\frac{1}{Vol\,(C_k)} \sum_{T \in C_k} \int_T P_i^n(x,y)dxdydz - \bar{u}^{k,n} \right)^2 \tag{16}$$

where T is a subtetrahedron.

$$H_i = \sum_{C_{k\neq i}\subset M_i} \Big[-\bar{u}^{k,n} + \frac{1}{Vol(C_k)} \sum_{T\in C_k} \int_T \left(\bar{u}^{i,n} + \sum_{\alpha\in I} c_{i,\alpha}^n \left[(X-X_{o,i})^\alpha - \frac{1}{Vol(C_i)} \sum_{T\in C_i} \int_T (X-X_{o,i})^\alpha \, dxdydz\right]\right)\Big]^2$$

The least square minimisation of H_i with respect to $c_{i,\alpha}^n(\alpha \in I)$, will define the polynomial P_i^n. It writes :

$$\frac{\delta H_i}{\delta c_{i,\alpha}^n} = 0, \alpha \in I. \tag{17}$$

From this condition we deduce the following system :

$$\begin{cases} 2 * \sum_{k\in V(i)} D_{i,k,1} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,2} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,3} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,4} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,5} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,6} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,7} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,8} * K_\alpha = 0 \\ 2 * \sum_{k\in V(i)} D_{i,k,9} * K_\alpha = 0 \end{cases}$$

where

$$D_{i,k,p} = \frac{1}{aire\,(C_k)} \sum_{T\in C_k} \int_T (X-X_{o,i})^p \, dxdydz - \frac{1}{aire\,(C_i)} \sum_{T\in C_i} \int_T (X-X_{o,i})^p \, dxdydz$$

for $p \in [1,2,3,4,5,6,7,8,9]$, and :

$$K_\alpha = \frac{1}{Vol\,(C_k)} \sum_{T\in C_k} \int_T \left(\bar{u}^{i,n} + \sum_{\alpha\in I} c_{i,\alpha}^n \left[(X-X_{o,i})^\alpha - \frac{1}{Vol\,(C_i)} \sum_{T\in C_i} \int_T (X-X_{o,i})^\alpha \, dxdydz\right]\right) - \bar{u}^{k,n}$$

We simplify K_α by using the following relation :

$$\frac{1}{Vol\,(C_k)} \int_{C_k} \left[\frac{1}{Vol\,(C_i)} \int_{C_i} u^n(x,y)\right] = \bar{u}^{i,n}(x,y).$$

We then obtain :

$$K_\alpha = \bar{u}^{i,n} + \frac{1}{Vol\,(C_k)} \sum_{T\in C_k} \int_T \sum_{\alpha\in I} c_{i,\alpha}^n \,(X-X_{o,i})^\alpha - \frac{1}{Vol\,(C_i)} \sum_{T\in C_i} \int_T \sum_{\alpha\in I} c_{i,\alpha}^n \,(X-X_{o,i})^\alpha \, dxdydz - \bar{u}^{k,n}$$

Taking, for example the computation of $\sum_{k\in V(i)} D_{i,k,1} * K_\alpha = 0$, we get :

$$\sum_{C_{k\neq i}\subset M_i} D_{i,k,1} \left[\bar{u}^{i,n}\right] + \sum_{C_{k\neq i}\subset M_i} D_{i,k,1} \left[\frac{1}{Vol\,(C_k)} \sum_{T\in C_k} \int_T \sum_{\alpha\in I} c_{i,\alpha}^n \,(X-X_{o,i})^\alpha\right]$$

$$- \sum_{C_{k\neq i}\subset M_i} D_{i,k,1} \left[\frac{1}{Vol\,(C_i)} \sum_{T\in C_i} \int_T \sum_{\alpha\in I} c_{i,\alpha}^n \,(X-X_{o,i})^\alpha \, dxdydz\right] - \sum_{C_{k\neq i}\subset M_i} D_{i,k,1} \left[\bar{u}^{k,n}\right] = 0$$

or in a more compact and readable manner :

$$\sum_{\alpha\in I}\sum_{C_{k\neq i}\subset M_i} D_{i,k,1}\left[\frac{1}{Vol\,(C_k)}\int_{C_k}(X-X_{o,i})^{\alpha}-\frac{1}{Vol\,(C_i)}\int_{C_i}(X-X_{o,i})^{\alpha}\,dxdydz\right]c_{i,\alpha}^{n}$$

$$\begin{cases}
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,1}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,1}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,2}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,2}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,3}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,3}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,4}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,4}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,5}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,5}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,6}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,6}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,7}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,7}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,8}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,8}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]\\
\sum\limits_{q\in[1,..,9]}\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,9}*D_{i,k,q}=\sum\limits_{C_{k\neq i}\subset M_i} D_{i,k,9}\left[\bar{u}^{k,n}-\bar{u}^{i,n}\right]
\end{cases}$$

We can now re-write the least square system to solve : under a compact form :

$$A_i c_i^n = b_i^n.$$

The integrals $\int_T (X-X_{o,i})^{\alpha}\,dxdydz$ forming the matrix are numerically computed with Gauss quadrature.

5.2 Cubic polynomial reconstruction

5.2.1 General case

On any cell C_i and any time phase we have to approximate the solution $u(x,y,z,t^n) = u(x,y,z)^n$ by reconstructing it with a cunic polynomial P_i^n.

We impose that the mean of the polynomial P_i^n (also written $\bar{P}_i^{i,n}$) in cell C_i is exactly equal to the mean of the current solution, $\bar{u}^{i,n}$) on cell C_i.

This condition is written : $\bar{P}_i^{i,n} = \bar{u}^{i,n}$, with :

$$\begin{cases} \bar{P}_i^{i,n} = \dfrac{1}{aire(C_i)} \displaystyle\int_{C_i} P_i^n(x,y)dxdy \\ \\ \bar{u}^{i,n} = \dfrac{1}{aire(C_i)} \displaystyle\int_{C_i} u^n(x,y)dxdy \end{cases}$$

The polynomial P_i^n to be found on cell C_i is defined by ;

$$P_i^n = \bar{u}^{i,n} + \sum_{\alpha \in I} c_{i,\alpha}^n \big[(X - X_{o,i})^\alpha - \overline{(X - X_{o,i})}^{i,\alpha} \big]$$

with :

$$\begin{cases} \overline{(X - X_{o,i})}^{i,\alpha} = \dfrac{1}{aire(C_i)} \displaystyle\int_{C_i} (X - X_{o,i})^\alpha dxdy \\ \\ I = \alpha = (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{N} \times \mathbb{N}, |\alpha| = \alpha_1 + \alpha_2 + \alpha_3 \in [1,3] \quad \text{is the set of multi-indices} \\ \\ (X - X_{o,i})^\alpha = (x - x_{o,i})^{\alpha_1} (y - y_{o,i})^{\alpha_2} (z - z_{o,i})^{\alpha_3} \end{cases}$$

where $(x_{o,i}, y_{o,i}, z_{o,i})$ will be taken equal to the centroid of cell i.

For a degree three polynomial the coefficients $c_{i,\alpha}$ are 19 (constant term not included).

To construct P_i^n on cell C_i, we need to define a stencil S_i made of a *sufficiently large number of cells* neighboring C_i.
Of course the stencil assembly is influenced by the number of direct neighbors. While all direct neighbors are put in S_i, we need extra cells (see figure 4), and the cardinal of S_i can be as high as 100 in order to have an accurate reconstruction.

FIGURE 4 – Construction de la molécule S_i centrée au nœud i. À gauche : cas où i a au plus cinq voisins, à droite : cas où i a moins de cinq voisins.

The 19 unknown coefficients $c_{i,\alpha}^n$ are again computed through a least square fitting, minimising the $l2$ distance between polynomial means and solution means :

$$H_i(c^n) = \sum_{k \in V(i)} \big(\bar{P}_i^{k,n} - \bar{u}^{k,n} \big)^2,$$

where $\bar{P}_i^{k,n}$ is the mean of P_i^n on C_k :

$$\bar{P}_i^{k,n} = \frac{1}{aire(C_k)} \int_{C_k} P_i^n(x,y,dz)dxdydz.$$

Remark : In order to make easier the fitting, we can add a diagonal term as follows :

$$H_i(c^n) = \sum_{k\in V(i)} \left(\bar{P}_i^{k,n} - \bar{u}^{k,n}\right)^2 \; + \; \varepsilon \sum_{\alpha} c_{i,\alpha}^2. \;\; \square \tag{18}$$

By construction of C_k, we can write :

$$H_i = \sum_{k\in V(i)} \left[\frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T P_i^n(x,y,z)dxdydz - \bar{u}^{k,n}\right]^2,$$

where T are the *subtetrahedra of cell* C_k. Equivalently we have :

$$H_i = \sum_{k\in V(i)} \left[H_{i,k}\right]^2 \qquad ; \qquad H_{i,k} = \frac{1}{aire(C_k)} \sum_{T\in C_k}$$
$$\int_T \Big(- \bar{u}^{k,n} + \bar{u}^{i,n} + \sum_{\alpha\in I} c_{i,\alpha}^n \left[(X - X_{o,i})^\alpha - \frac{1}{aire(C_i)} \sum_{T'\in C_i} \int_{T'} (X - X_{o,i})^\alpha dx'dy'dz'\right]\Big) dxdydz$$

Thus :

$$H_i = \sum_{k\in V(i)} \left[H_{i,k}^1 + H_{i,k}^2 + H_{i,k}^3 + H_{i,k}^4\right]^2$$
$$H_{i,k}^1 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T (-\bar{u}^{k,n})dxdydz$$
$$H_{i,k}^2 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T (+\bar{u}^{i,n})dxdydz$$
$$H_{i,k}^3 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T \left[\sum_{\alpha\in I} c_{i,\alpha}^n (X - X_{o,i})^\alpha\right] dxdydz$$
$$H_{i,k}^4 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T \left[- \sum_{\alpha\in I} c_{i,\alpha}^n \frac{1}{aire(C_i)} \sum_{T'\in C_i} \int_{T'} (X - X_{o,i})^\alpha dx'dy'dz'\right] dxdydz.$$

We observe that the integral on subtetraha=edra commute with the uniform functions :

$$H_{i,k}^1 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T (-\bar{u}^{k,n})dxdydz = -\bar{u}^{k,n}$$

$$H_{i,k}^2 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_T \bar{u}^{i,n} dxdydz = \bar{u}^{i,n}$$

We have also a constant quantity :

$$H_{i,k}^4 = \frac{1}{aire(C_k)} \sum_{T\in C_k} \int_{T'} \sum_{\alpha\in I} -c_{i,\alpha}^n \left[\frac{1}{aire(C_i)} \sum_{T'\in C_i} \int_T (X - X_{o,i})^\alpha dx'dy'dz'\right] dxdydz =$$
$$\sum_{\alpha\in I} -c_{i,\alpha}^n \left[\frac{1}{aire(C_i)} \sum_{T'\in C_i} \int_{T'} (X - X_{o,i})^\alpha dx'dy'dz'\right] \quad = \quad \sum_{\alpha\in I} -c_{i,\alpha}^n \left[\frac{D_{i,\alpha}}{aire(C_i)}\right]$$

whre we have introduced the notation :

$$D_{i,\alpha} = \sum_{T \in C_i} \int_T (X - X_{o,i})^\alpha dxdydz.$$

Implementation :

```
           DO  igauss = 1,4
               DO  m = 1,3
                        cg(m) = (alp1(igauss)-1.0)*c(m,i)
     1                        + alp2(igauss)*cIij(m)
     1                        + alp3(igauss)*cgijk(m)
     1                        + alp4(igauss)*cGijkl(m)
               ENDDO ! DO  m = 1,3

c Contributions to integrals of 1, x,x*x,x*y,x*z
c Contributions to integrals of 1, y,y*x,y*y,y*z
c Contributions to integrals of 1, z,z*x,z*y,z*z
               sum0      = sum0     +       wgtval(igauss)*volminitet
               sum(is)   = sum(is)  +       wgtval(igauss)*volminitet

               sumx(is)  = sumx(is) + cg(1)*wgtval(igauss)*volminitet
               sumy(is)  = sumy(is) + cg(2)*wgtval(igauss)*volminitet
               sumz(is)  = sumz(is) + cg(3)*wgtval(igauss)*volminitet

               sumxx(is) = sumxx(is) + cg(1)*cg(1)*wgtval(igauss)*volminitet
               sumyx(is) = sumyx(is) + cg(2)*cg(1)*wgtval(igauss)*volminitet
               sumzx(is) = sumzx(is) + cg(3)*cg(1)*wgtval(igauss)*volminitet

               sumyy(is) = sumyy(is) + cg(2)*cg(2)*wgtval(igauss)*volminitet
               sumzy(is) = sumzy(is) + cg(3)*cg(2)*wgtval(igauss)*volminitet

               sumzz(is) = sumzz(is) + cg(3)*cg(3)*wgtval(igauss)*volminitet

            ENDDO ! DO  igauss = 1,4
```

We also introduce the following notations :

$$\bar{D}_{i,k,\alpha} = \frac{1}{aire(C_k)} \sum_{T \in C_k} \int_T (X - X_{o,i})^\alpha dxdydz$$

and :

$$D_{i,k,\alpha} = \bar{D}_{i,k,\alpha} - \frac{D_{i,\alpha}}{aire(C_i)}.$$

Finally we have shown that :

$$H_{i,k} = H^1_{i,k} + H^2_{i,k} + H^3_{i,k} + H^4_{i,k}$$

with :

$$H^1_{i,k} = -\bar{u}^{k,n}$$

$$H^2_{i,k} = \bar{u}^{i,n}$$

$$H^3_{i,k} = \frac{1}{aire(C_k)} \sum_{T \in C_k} \int_T \left[\sum_{\alpha \in I} c^n_{i,\alpha} (X - X_{o,i})^\alpha \right] dxdydz = \sum_{\alpha \in I} c^n_{i,\alpha} \bar{D}_{i,k,\alpha}$$

$$H^4_{i,k} = \sum_{\alpha \in I} -c^n_{i,\alpha} \left[\frac{D_{i,\alpha}}{aire(C_i)} \right].$$

We deduce a simplified writting for $H_{i,k}$:

$$H_{i,k} = -\bar{u}^{k,n} + \bar{u}^{i,n} + \sum_{\alpha \in I} c^n_{i,\alpha} \left[+ \bar{D}_{i,k,\alpha} - \frac{D_{i,\alpha}}{aire(C_i)} \right] \quad = \quad -\bar{u}^{k,n} + \bar{u}^{i,n} + \sum_{\alpha \in I} c^n_{i,\alpha} D_{i,k,\alpha}.$$

We transform now the numbering of monomials in a more tractable form :

$$1 \leq |\alpha_p| \leq 3 \Leftrightarrow \alpha_p \in \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9, \alpha_{10}, \alpha_{10}, \alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}, \alpha_{16}, \alpha_{17}, \alpha_{18}, \alpha_{19}, \} \tag{19}$$

$$\begin{array}{ccccc} & & D_{\alpha_1} = D_x, & D_{\alpha_2} = D_y, & D_{\alpha_3} = D_z, \\ & & D_{\alpha_4} = D_{xx}, , & D_{\alpha_5} = D_{yy}, & D_{\alpha_6} = D_{zz}, \\ & & D_{\alpha_7} = D_{xy}, & D_{\alpha_8} = D_{xz}, & D_{\alpha_9} = D_{yz}, \\ D_{\alpha_{10}} = D_{xxx}, & D_{\alpha_{11}} = D_{yyy}, & D_{\alpha_{12}} = D_{zzz}, & D_{\alpha_{13}} = D_{xxy}, & D_{\alpha_{14}} = D_{xxz}, \\ D_{\alpha_{15}} = D_{xyy}, & D_{\alpha_{16}} = D_{yyz}, & D_{\alpha_{17}} = D_{xzz}, & D_{\alpha_{18}} = D_{yzz}, & D_{\alpha_{19}} = D_{xyz}, \end{array} \tag{20}$$

with the adapted notation :

$$D_{i,p} = D_{i,\alpha_p} \quad ; \quad \bar{D}_{i,k,p} = \bar{D}_{i,k,\alpha_p} \quad ; \quad D_{i,k,p} = D_{i,k,\alpha_p}$$

Let us transformthe terms $\sum_{T \in C_k} \int_T (X - X_{o,i})^\alpha \; dxdydz$ présents dans $\bar{D}_{i,k,\alpha}$. We remark that (for $\alpha_1 = (1,0,0)$)

$$\sum_{T \in C_k} \int_T (x - x(i)) dxdydz = \sum_{T \in C_k} \int_T (x - x(k)) dxdydz + aire(C_k)(x(k) - x(i)) \tag{21}$$

which implies that

$$\begin{aligned} D_{i,k,1} &= \frac{D_{k,1}}{aire(C_k)} + (x(k) - x(i)) - \frac{D_{i,1}}{aire(C_i)} \\ D_{i,k,2} &= \frac{D_{k,2}}{aire(C_k)} + (y(k) - y(i)) - \frac{D_{i,2}}{aire(C_i)} \\ D_{i,k,3} &= \frac{D_{k,3}}{aire(C_k)} + (z(k) - z(i)) - \frac{D_{i,3}}{aire(C_i)}. \end{aligned} \tag{22}$$

```
x        = coco(1,is)
y        = coco(2,is)
z        = coco(3,is)

xsv      = coco(1,isv)
ysv      = coco(2,isv)
zsv      = coco(3,isv)

dxsv     = x - xsv
dysv     = y - ysv
dzsv     = z - zsv

DD(ineig,1) = D(1,2)/vols(isv) - dxsv - D(1,1)/vols(is)
DD(ineig,2) = D(2,2)/vols(isv) - dysv - D(2,1)/vols(is)
DD(ineig,3) = D(3,2)/vols(isv) - dzsv - D(3,1)/vols(is)
```

Remarking that :

$$\begin{aligned}(x-x(i))^2 &= x^2 - 2x(i)x + x(i)^2 \\ &= x^2 - 2x(k)x + x(k)^2 - 2x(i)x + 2x(k)x + x(i)^2 - x(k)^2 \\ &= x^2 - 2x(k)x + x(k)^2 - 2(x(i)-x(k))x + (x(i)^2 - x(k)^2)\end{aligned}$$

$$\begin{aligned}(x-x(i))^2 &= (x-x(k))^2 - 2(x(i)-x(k))(x-x(k)) - 2(x(i)-x(k))x(k) + (x(i)^2 - x(k)^2) \\ &= (x-x(k))^2 - 2(x(i)-x(k))(x-x(k)) - 2(x(i)-x(k))x(k) + (x(i)^2 - x(k)^2) \\ &= (x-x(k))^2 - 2(x(i)-x(k))(x-x(k)) + x(i)^2 - 2x(i)x(k) + x(k)^2 \\ &= (x-x(k))^2 - 2(x(i)-x(k))(x-x(k)) + x(i)^2 - 2x(i)x(k) + x(k)^2 \\ &= (x-x(k))^2 - 2(x(i)-x(k))(x-x(k)) + (x(i)-x(k))^2\end{aligned}$$

We deduce :

$$\begin{aligned}\sum_{T\in C_k}\int_T (x-x(i))^2 dxdydz = &\sum_{T\in C_k}\int_T (x-x(k))^2 dxdydz \\ &-2(x(i)-x(k))\sum_{T\in C_k}\int_T (x-x(k))dxdydz \\ &+aire(C_k)(x(i)-x(k))^2\end{aligned} \tag{23}$$

which also writes $(\alpha_4 = (2,0,0,), \alpha_1 = (1,0,0,)$:

$$\sum_{T\in C_k}\int_T (x-x(i))^2 dxdydz = D_{k,4} - 2(x(i)-x(k))D_{k,1} + aire(C_k)(x(i)-x(k))^2. \tag{24}$$

De même en y :

$$\sum_{T\in C_k}\int_T (y-y(i))^2 dxdydz = D_{k,5} - 2(y(i)-y(k))D_{k,2} + aire(C_k)(y(i)-y(k))^2. \tag{25}$$

Similarly in z :

$$\sum_{T\in C_k}\int_T (z-z(i))^2 dxdydz = D_{k,6} - 2(z(i)-z(k))D_{k,3} + aire(C_k)(z(i)-z(k))^2. \tag{26}$$

Which also writes :

$$\begin{aligned}
D_{i,k,4} &= \frac{D_{k,4}-2(x(i)-x(k))D_{k,1}}{aire(C_k)} + (x(i)-x(k))^2 - \frac{D_{i,4}}{aire(C_i)}.\\
D_{i,k,5} &= \frac{D_{k,5}-2(y(i)-y(k))D_{k,2}}{aire(C_k)} + (y(i)-y(k))^2 - \frac{D_{i,5}}{aire(C_i)}.\\
D_{i,k,6} &= \frac{D_{k,6}-2(z(i)-z(k))D_{k,3}}{aire(C_k)} + (z(i)-z(k))^2 - \frac{D_{i,6}}{aire(C_i)}.
\end{aligned} \tag{27}$$

```
DD(ineig,4) = ( D(4,2)-2*dxsv*D(1,2) )/vols(isv) + dxsv**2 - D(4,1)/vols(is)
DD(ineig,5) = ( D(5,2)-2*dysv*D(2,2) )/vols(isv) + dysv**2 - D(5,1)/vols(is)
DD(ineig,6) = ( D(6,2)-2*dzsv*D(3,2) )/vols(isv) + dzsv**2 - D(6,1)/vols(is)
```

Remarking that :

$$\begin{aligned}
(x-x(i))(y-y(i)) &= (x-x(k))(y-y(k)) + (x-x(i))(y-y(i)) - (x-x(k))(y-y(k))\\
&= (x-x(k))(y-y(k)) - (x(i)-x(k))y\\
&\quad -(y(i)-y(k))x + x(i)y(i) - x(k)y(k)\\
&= (x-x(k))(y-y(k))\\
&\quad -(x(i)-x(k))(y-y(k)) - (x(i)-x(k)y(k)\\
&\quad -(y(i)-y(k))(x-x(k)) - (y(i)-y(k))x(k)\\
&\quad +x(i)y(i) - x(k)y(k)\\
&= (x-x(k))(y-y(k)) - (x(i)-x(k))(y-y(k)) - (y(i)-y(k))(x-x(k))\\
&\quad -x(i)y(k) + x(k)y(k) - y(i)x(k) + y(k))x(k) + x(i)y(i) - x(k)y(k)\\
&= (x-x(k))(y-y(k)) - (x(i)-x(k))(y-y(k)) - (y(i)-y(k))(x-x(k))\\
&\quad -x(i)y(k) + x(k)y(k) - y(i)x(k) + x(i)y(i)
\end{aligned}$$

We derive :

$$\begin{aligned}
D_{i,k,7} &= \frac{D_{k,7} - (x(i) - x(k))D_{k,2} - (y(i) - y(k))D_{k,1}}{aire(C_k)} \\
&\quad + x(i)y(i) + x(k)y(k) - x(i)y(k) - x(k)y(i) - \frac{D_{i,7}}{aire(C_i)} \\
D_{i,k,8} &= \frac{D_{k,8} - (x(i) - x(k))D_{k,3} - (z(i) - z(k))D_{k,1}}{aire(C_k)} \\
&\quad + x(i)z(i) + x(k)z(k) - x(i)z(k) - x(k)z(i) - \frac{D_{i,8}}{aire(C_i)} \\
D_{i,k,9} &= \frac{D_{k,9} - (y(i) - y(k))D_{k,3} - (z(i) - z(k))D_{k,2}}{aire(C_k)} \\
&\quad + y(i)z(i) + y(k)z(k) - y(i)z(k) - y(k)z(i) - \frac{D_{i,9}}{aire(C_i)}.
\end{aligned} \tag{28}$$

```
DD(ineig,7) = ( D(7,2)-dxsv*D(2,2) - dysv*D(1,2) )/vols(isv)
DD(ineig,8) = ( D(8,2)-dxsv*D(3,2) - dzsv*D(1,2) )/vols(isv)
DD(ineig,9) = ( D(9,2)-dysv*D(3,2) - dzsv*D(2,2) )/vols(isv)

DD(ineig,7) = DD(ineig,7) + (y*y-xsv*ysv-x*ysv-xsv*y) - D(7,1) /vols(is)
DD(ineig,8) = DD(ineig,8) + (x*z-xsv*zsv-x*zsv-xsv*z) - D(8,1) /vols(is)
DD(ineig,9) = DD(ineig,9) + (y*z-ysv*zsv-y*zsv-ysv*z) - D(9,1) /vols(is)
```

Remarking that :

$$(x - x(i))^3 = (x - x(k) - x(i) + x(k))^3$$

Putting $a = (x - x(k))$, $b = (x(i) - x(k))$ and using the fact that : $(a - b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$, we have :$(x - x(i))^3 = (x - x(k))^3 - 3(x - x(k))^2(x(i) - x(k)) + 3(x - x(k))(x(i) - x(k))^2 - (x(i) - x(k))^3$

Putting $(x(i) - x(k)) = dx, (y(i) - y(k)) = dy, (z(i) - z(k)) = dz$ we deduce :

$$\begin{aligned}
D_{i,k,10} &= \frac{3dx^2 D_{k,1} - 3dx D_{k,4} + D_{k,10}}{aire(C_k)} - dx^3 - \frac{D_{i,10}}{aire(C_i)} \\
D_{i,k,11} &= \frac{3dy^2 D_{k,2} - 3dy D_{k,5} + D_{k,11}}{aire(C_k)} - dy^3 - \frac{D_{i,11}}{aire(C_i)} \\
D_{i,k,12} &= \frac{3dz^2 D_{k,3} - 3dz D_{k,6} + D_{k,12}}{aire(C_k)} - dz^3 - \frac{D_{i,12}}{aire(C_i)}
\end{aligned} \tag{29}$$

Remarking that :

$$(x - x(i))(y - y(i))^2 = (x - x(k) - x(i) + x(k))(y - y(k) - y(i) + y(k))^2$$

putting $a = (x - x(k))$, $b = (x(i) - x(k))$, $c = (y - y(k))$, $d = (y(i) - y(k))$ and using the fact that : $(a - b)(c - d)^2 = ac^2 - 2acd + ad^2 - bc^2 + 2bcd - bd^2$, we have :$(x - x(i))(y - y(i))^2 = (x - x(k))(y - y(k))^2 - 2(x - x(k))(y - y(k))(y(i) - y(k))$

$+ (x - x(k))(y(i) - y(k))^2 - (x(i) - x(k))(y - y(k))^2$
$+ 2(x(i) - x(k))(y - y(k))(y(i) - y(k)) - (x(i) - x(k)(y(i) - y(k))^2$

Putting $(x(i) - x(k)) = dx, (y(i) - y(k)) = dy, (z(i) - z(k)) = dz$ we deduce :

$$\begin{aligned}
D_{i,k,13} &= \frac{dx^2 D_{k,2} + 2dxdyD_{k,1} - 2dxD_{k,7} - dyD_{k,4} + D_{k,13}}{aire(C_k)} - dx^2 dy - \frac{D_{i,13}}{aire(C_i)} \\
D_{i,k,14} &= \frac{dx^2 D_{k,3} + 2dxdzD_{k,1} - 2dxD_{k,8} - dzD_{k,4} + D_{k,14}}{aire(C_k)} - dx^2 dz - \frac{D_{i,14}}{aire(C_i)} \\
D_{i,k,15} &= \frac{dy^2 D_{k,1} + 2dxdyD_{k,2} - 2dyD_{k,7} - dxD_{k,5} + D_{k,15}}{aire(C_k)} - dy^2 dx - \frac{D_{i,15}}{aire(C_i)} \\
D_{i,k,16} &= \frac{dy^2 D_{k,3} + 2dzdyD_{k,2} - 2dyD_{k,9} - dzD_{k,5} + D_{k,16}}{aire(C_k)} - dy^2 dz - \frac{D_{i,16}}{aire(C_i)} \\
D_{i,k,17} &= \frac{dz^2 D_{k,1} + 2dxdzD_{k,3} - 2dzD_{k,8} - dxD_{k,6} + D_{k,17}}{aire(C_k)} - dz^2 dx - \frac{D_{i,17}}{aire(C_i)} \\
D_{i,k,18} &= \frac{dz^2 D_{k,2} + 2dydzD_{k,3} - 2dzD_{k,9} - dyD_{k,6} + D_{k,18}}{aire(C_k)} - dz^2 dy - \frac{D_{i,18}}{aire(C_i)}
\end{aligned} \tag{30}$$

Remarking that :

$$(x-x(i))(y-y(i))(z-z(i)) = (x-x(k)-x(i)+x(k))(y-y(k)-y(i)+y(k))(z-z(k)-z(i)+z(k))$$

putting $a = (x - x(k))$, $b = (x(i) - x(k))$, $c = (y - y(k))$, $d = (y(i) - y(k))$,$e = (z - z(k))$,$f = (z(i) - z(k))$ and using that : $(a - b)(c - d)(e - f) =$
$eac - ead - ebc + ebd - fac + fad + fbc - fbd$, we get : $(x - x(i))(y - y(i))(z - z(i)) = (z - z(k))(x - x(k))(y - y(k))$
$- (z - z(k))(x - x(k))(y(i) - y(k))$
$- (z - z(k))(x(i) - x(k))(y - y(k))$
$+ (z - z(k))(x(i) - x(k))(y(i) - y(k))$
$- (z(i) - z(k))(x - x(k))(y - y(k))$
$+ (z(i) - z(k))(x - x(k))(y(i) - y(k))$
$+ (z(i) - z(k))(x(i) - x(k))(y - y(k))$
$- (z(i) - z(k))(x(i) - x(k))(y(i) - y(k))$

Putting $(x(i) - x(k)) = dx, (y(i) - y(k)) = dy, (z(i) - z(k)) = dz$ we deduce :

$$\begin{aligned}
D_{i,k,19} &= \frac{-dxD_{k,9} - dyD_{k,8} - dzD_{k,7} + dydzD_{k,1} + dxdzD_{k,2} + dxdyD_{k,3} + D_{k,19}}{aire(C_k)} \\
&\quad -dxdydz - \frac{D_{i,19}}{aire(C_i)}
\end{aligned} \tag{31}$$

5.2.2 Partition case

Over each molecule we reconstruct only one polynomial P_{M_i} using the previous technique over the most centered cell in the molecule. Defined by

$$C_{rep} = argmin_{C_i \in Mi} \, |cg(C_i) - cg(M_i)| \tag{32}$$

Where cg() is the center of gravity.

Once the polynomials coefficients are obtained we have to add a constant therm P_k^0 in order to obtain :

$$\frac{1}{C_k}\int_{C_k} P_{M_i} + P_k^0 = \bar{u}^{k,n}, \forall C_k \in M_i \tag{33}$$

Using the polynomial definition we have :

$$\frac{1}{C_k}\int_{C_k} P_{M_i} = \frac{1}{C_k}\int_{C_k} \big(\bar{u}^{rep,n} + \sum_{\alpha\in I} c_{rep,\alpha}^n \big[(X - X_{o,rep})^\alpha - \overline{(X - X_{o,rep})}^{rep,\alpha}\big]\big) \tag{34}$$

$$= \bar{u}^{rep,n} + \sum_{\alpha\in I} c_{rep,\alpha}^n \big[\frac{1}{C_k}\int_{C_k} (X - X_{o,rep})^\alpha - \overline{(X - X_{o,rep})}^{rep,\alpha}\big] \tag{35}$$

$$= \bar{u}^{rep,n} + \sum_{\alpha\in I} c_{rep,\alpha}^n \big[\bar{D}_{rep,k,\alpha} - \frac{1}{aire(C_{rep)}} D_{rep,\alpha}\big] \tag{36}$$

$$= \bar{u}^{rep,n} + \sum_{\alpha\in I} c_{rep,\alpha}^n \big[D_{rep,k,\alpha}\big] \tag{37}$$

Then the constants terms are defined as :

$$P_k^0 = \bar{u}^{k,n} - \bar{u}^{rep,n} - \sum_{\alpha\in I} c_{rep,\alpha}^n \big[D_{rep,k,\alpha}\big] \tag{38}$$

6 Flux evaluation

6.1 Flux evaluation for third-order accuracy

The next objective is to evaluate the flow at time t on the interfaces between the cell C_i and its neighbors C_k. We use a 3-point Gaussian quadrature on the external facets of the sub-tetrahedra. Recall that the flow on the border of a cell can be written :

$$\int_{\partial C_i} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds = \sum_{k \ inV(i)} \int_{\partial C_i \cap \partial C_k} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds$$

The intersection between two cells $\partial C_i \cap \partial C_k$ can be broken down into several triangles as shown in Figure 5. We arbitrarily number these triangles $T_{\{i,k,r\}}$, with $1 \leq r \leq R_{i,k}$, where $R_{i,k}$ represents the total number of triangle making up the interface $\partial C_i \cap \partial C_k$.

$$\int_{\partial C_i} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds = \sum_{k\in V(i)} \sum_{1\leq r\leq R_{i,k},} \int_{T_{\{i,k,r\}}} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds.$$

On the whole cell C_i, the solution $u(x,y,z)$ is approximated by the polynomial P_i^n defined in previous section.

$$\int_{\partial C_i} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds = \sum_{k\in V(i)} \sum_{1\leq r\leq R_{i,k},} \int_{T_{\{i,k,r\}}} \mathbf{f}\left(P_i(x,y,z,t)\right) \cdot \mathbf{n} ds.$$

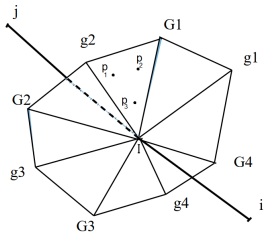


FIGURE 5 – Numerical quadrature for flux integration.

We use a Gauss quadrature (Figure 5) with three integration nodes for the evaluation of $\int_{T_{\{i,k,r\}}} \mathbf{f}\left(P_i(x,y,z,t)\right) \cdot \mathbf{n} ds$. For a triangle $T_{\{i,k,r\}}$ we denote the Gauss nodes $p^l_{\{i,k,r\}} = (x^l_{\{i,k,r\}}, y^l_{\{i,k,r\}}, z^l_{\{i,k,r\}})$ with $l \in [1,2,3]$ and their weights $\omega = \frac{1}{3}$.

$$\int_{\partial C_i} \mathbf{f}(u(x,y,t)) \cdot \mathbf{n} ds = \sum_{k \in V(i)} \sum_{1 \le r \le R_{i,k}} \sum_{l \in [1,2,3]} \omega \mathbf{f}\left(P_i(p^l_{\{i,k,r\}}, t)\right) \cdot \mathbf{n}_{\{i,k,r\}} ds$$

with $\mathbf{n}_{\{i,k,r\}} = \int_{T_{\{i,k,r\}}} \mathbf{n}(x,y,z) ds$.

However, since two distinct polynomials P_i and P_k were constructed on either side of the interface between cells C_i and C_k, the reconstructed solution on each Gaussian point takes two *a priori* different values.

Therefore we approximate the flux on the $T_{\{i,k,r\}}$ interface by a numerical Φ flux function :

$$\mathbf{f}\left(P_i\left(p^l_{\{i,k,r\}}, t\right)\right) \cdot \mathbf{n} = \Phi\left(P_i\left(p^l_{\{i,k,r\}}, t\right), P_k\left(p^l_{\{i,k,r\}}, t\right), \mathbf{n}_{\{i,k,r\}}\right)$$

We need to choose the numerical flux Φ such that the resulting scheme is stable. We choose the Donor Cell upwind solver, defined by the following relation :

$$\Phi\left(u_1, u_2, \mathbf{v}\right) = \frac{\mathbf{f}\left(u_1\right) + \mathbf{f}\left(u_2\right)}{2} \cdot \mathbf{v} - \frac{\gamma}{2} \left| \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \left(\frac{u_1 + u_2}{2} \right) \cdot \mathbf{v} \right| \left(u_2 - u_1\right) \tag{39}$$

where the regular option is to take $\gamma = 1$..

6.2 Flux evaluation for fourth-order accuracy

We follow the recommandation in [14] who propose the integration described in Table 1 .

Point location	Weight
Triangles	
$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$-\frac{9}{16}$
$(\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$	$\frac{25}{48}$
$(\frac{1}{5}, \frac{3}{5}, \frac{1}{5})$	$\frac{25}{48}$
$(\frac{3}{5}, \frac{1}{5}, \frac{1}{5})$	$\frac{25}{48}$
Tetrahedra	
$(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	$-\frac{4}{5}$
$(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{2})$	$\frac{9}{20}$
$(\frac{1}{6}, \frac{1}{6}, \frac{1}{2}, \frac{1}{6})$	$\frac{9}{20}$
$(\frac{1}{6}, \frac{1}{2}, \frac{1}{6}, \frac{1}{6})$	$\frac{9}{20}$
$(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$	$\frac{9}{20}$

TABLE 1 – Quadrature for fourth-order CENO in tetrahedra.

7 Time advancing

7.1 Explicit time advancing

We use the usual fourth order Runge-Kutta time advancing :

$$\begin{array}{l} u^{n+1} = u^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \text{ with} \\ k_1 = F(t^n, u^n), \\ k_2 = F(t^n + \frac{h}{2}, u^n + \frac{h}{2}k_1), \\ k_3 = F(t^n + \frac{h}{2}, u^n + \frac{h}{2}k_2), \\ k_4 = F(t^n + h, u^n + hk_3). \end{array} \tag{40}$$

7.2 Implicit time advancing

The third-order and fourth-order spatial CENO schemes can be advanced in time with a first backward differencing formulart (BDF1) which is linearized with a simplified spatially first order Jacobian.

8 Numerical validation of an advection kernel

The CENO schemes have be implemented in a C++ research kernel computing solely advection (with uniform velocity).

8.1 Third-order validations

We examine the calculation of a translation of a Gaussian distribution in a square computational domain. The height of the Gaussian distribution is 1.. The advection velocity is $(1., 0., 0.)$. Figure 6 gives an idea of the solution evolution. The numerical error is measured before the solution

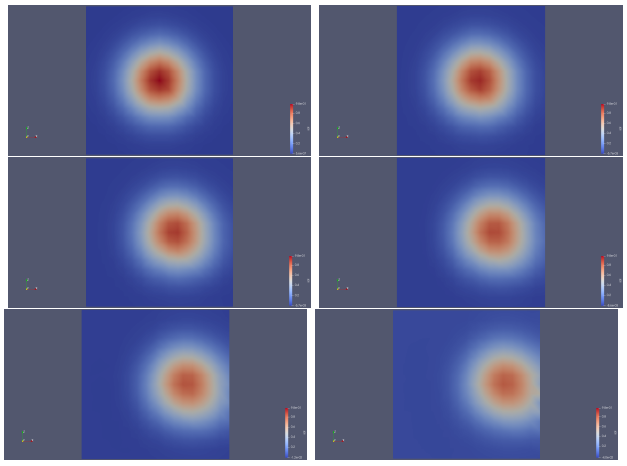


FIGURE 6 – Time advancing of a Gaussian.

interacts with the boundary. The CENO with quadratic reconstruction (CENO2) is theoretically third-order accurate independantly of the level of dissipation brought by the upwinding. But if no dissipation is added, error components of high frequency may destroy the convergence to

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00954	-	0.013869	-
4913	24576	0.00237	2.0091	0.00356255	1.96088
35937	196608	0.00039	2.603341	0.000790648	2.17180
274625	1572864	0.0000554	2.81551	0.000137616	2.52239
2146689	12582912	6.93317e-06	2.96158	2.00248e-05	2.78079

TABLE 2 – **Propagation of a Gaussian distribution.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

continuous solution. Conversely, the CENO2 scheme with an upwinding with full dissipation is not necessarily the most accurate. Following [5], we propose now a study of the accuracy for various values of γ.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00912991	-	0.0133377	-
4913	24576	0.00220518	2.049	0.00347692	1,93963
35937	196608	0.000367078	2.5867	0.000778426	2,15918
274625	1572864	0.0000498079	2.88164	0.000134288	2,53523

TABLE 3 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.9$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.0082845	-	0.0122967	-
4913	24576	0.00187837	2.14093	0.00335846	1.8742
35937	196608	0.000311715	259118	0.000770148	2.12459
274625	1572864	0.0000424097	2.87776	0.000129721	2.56972

TABLE 4 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.7$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00745004	-	0.0112601	-
4913	24576	0.0015774	2.2397	0.0033149	1,76418
35937	196608	0.000267363	2.56068	0.00079893	2,05282
274625	1572864	3.79196e-05	2.8178	0.00013209	2,59655
2146689	12582912	4.95399e-06	2,93628	1.84155e-05	2,84253

TABLE 5 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.5$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00663919	-	0.0103138	-
4913	24576	0.00131967	2.33083	0.00338815	1.60601
35937	196608	0.000246373	2.42126	0.000898604	1.91474
274625	1572864	0.0000411102	2.58328	0.00015429	2.54204

TABLE 6 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.3$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00598064	-	0.00959042	-
4913	24576	0.00115091	2.37752	0.00365705	1.39091
35937	196608	0.00027314	2.07506	0.00112703	1,69815
274625	1572864	0.0000633621	2.10795	0.0002328	2.27536

TABLE 7 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.1$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

Vertices N_e	Elements N_v	Error(C)	Order(C)	Error(P)	Order(P)
729	3072	0.00572187	-	0.00937782	-
4913	24576	0.00112617	2.34506	0.00388892	1.26988
35937	196608	0.000311919	1.85218	0.00131462	1.56472
274625	1572864	0.0000886359	1.81521	0.000320326	2.03703

TABLE 8 – **Propagation of a Gaussian distribution using the Donor Cell upwind solver $\gamma = 0.$.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm. Error(P) and convergence order(P) correspond to the error on solution with macromolecules built with the partition algorithm.

8.2 Fourth-order validation

A preliminary result is presented in Table 9. We observe that in similar conditions, the fourth-order schem is slightly less accurate than the third-order one for the coarsest mesh. On the contrary, for the second coarse mesh the error is 50% smaller and convergence order is already higher. For the finer mesh the fourth order is well obtained.

Vertices N_e	Elements N_v	Error(C)	Order(C)
729	3072	0.00832135	-
4913	24576	0.00107831	2.94805
35937	196608	8.20648e-05	3.71586
274625	1572864	5.03349e-06	4.02713

TABLE 9 – **Propagation of a Gaussian distribution using the fourth-order cubic reconstruction, and a Donor Cell upwind solver** $\gamma = 0.5$**.** Error(C) and convergence order(C) correspond to the error on solution with macromolecules built with the centered algorithm.

8.3 Computational cost for the third-order version

We give now a few informations concerning the computational cost of the different phases of the computation. The computational time has been measured on a DELL PRECISION Mobile 7550 equipped with an Intel Core i7 10875H (8 cores, 2.3Ghz).

We focus on the mesh of 35937 vertices.

8.3.1 With centered molecules

During the preprocessing, we have :
(a)- Computation of the volume of cells : 2.8 sec.
(b)- Centroids of cells : 10.2sec.
(c)- Building of macromolecules : 0.71sec.
(d)- Computations of coefficients $D_{i,\alpha}$: 66.9 sec.
(e)- Computations of coefficients $D_{i,k,\alpha}$: 0.67sec.
(f)- Inversion of the matrices : 4.3sec.
(g)- Computation of cell means for the initial solution : 6.5 sec.

During time advancing :
(h)- RHS of reconstruction : 0.17 sec.
(i)- Coefficients of polynomials : 0.19 sec.
(j)- Internal fluxes : 4.9 sec.
(k)- Boundary fluxes : 0.37 sec.

Total time for one RK4 time step : 23.9 sec.

8.3.2 With partition

The number of molecules is 871

During the preprocessing, we have :
(a)- Computation of the volume of cells : 2.9 sec.
(b)- Centroids of cells : 10.33 sec.
(c)- Building of macromolecules : 0.114sec.
(d)- Computations of coefficients $D_{i,\alpha}$: 69.16 sec.
(e)- Computations of coefficients $D_{i,k,\alpha}$: 0.0008sec.
(f)- Inversion of the matrices : 0.117sec.
(g)- Computation of cell means for the initial solution : 6.5 sec.

During time advancing :
(h)- RHS of reconstruction : 0.005 sec.
(i)- Coefficients of polynomials : 0.013 sec.
(ibis)- const.coef. : 8.5 sec.
(j)- Internal fluxes : 4.78 sec.
(k)- Boundary fluxes : 0.37 sec.

Total time for one RK4 time step : 57.11 sec. if the constant coefficient adjustment is performed but only 19.4 if the integral of monomials are computed onece for all.

Globally, the most costly step is the computation of fluxes, independantly of the way the molecules are constructed.

9 Numerical validation in a CFD code

The CENO third-order and fourth-order schemes have been implemented in a beta-version of the NiceFlow platform of Lemma. NiceFlow is equipped with two spatial approximations, the TV4 approximation is a second-order MUSCL approximation (third-order accurate on Cartesian meshes if no limiter is used) and the TV6 is a second-order MUSCL approximation (fifth-order accurate on Cartesian meshes if no limiter is used).

9.1 SOD tube test case

The purpose of this test case is to check that the Euler fluxes are consistently computed. We use a $500 * 5 * 5$ Cartesian mesh with $dx = 0.002$, split into 48000 tetrahedra. In order to keep a computation sufficiently stable we use inside the Roe approximate Riemann solver the projection of left and right densities (resp. total energy) into the interval $[min(\rho_L, \rho_R), max(\rho_L, \rho_R)]$ (resp. $[min(\rho_L E_L, \rho_R E_R), max(\rho_L E_L, \rho_R E_R)]$. Euler calculations produce results which are still oscillating, and we test also the Sod problem with a physical viscosity of 0.001. We advance explicitly in time with a CFL pf 0.4. We compare in Figures 7, 8, 9, 10, 11, and 12 the results obtained by the different options.

We observe that the partition-based molecules produce slightly more oscillating results.

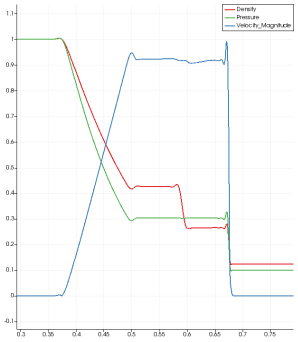


FIGURE 7 – CENO O3 at physical time 0.1 sec. CPU = 7 min 56 sec., 461 iterations

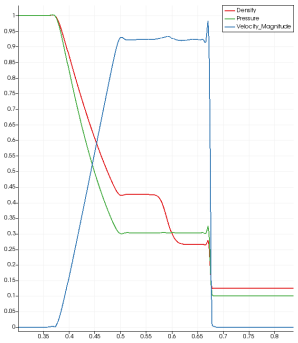


FIGURE 8 – CENO O3 at physical time 0.1 sec. Solving Navier-Stokes Equations with viscosity = 0.001

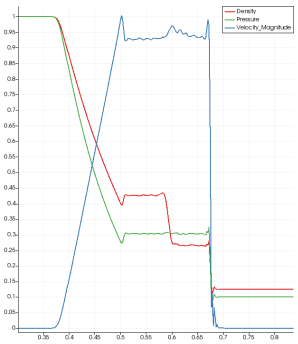


FIGURE 9 – CENO O3 with partition-based molecules at physical time 0.1 sec., CPU = 6 min 25 sec 461

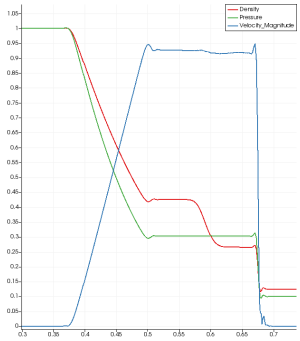


FIGURE 10 – CENO O4 at physical time 0.1 sec., CPU = 8 min 31 sec, 643 iterations. Solving Navier-Stokes Equatiosn with viscosity = 0.001

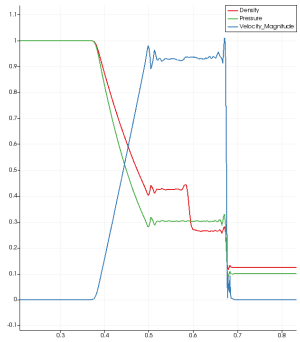


FIGURE 11 – CENO O4 with partition-based molecules at physical time 0.1 sec

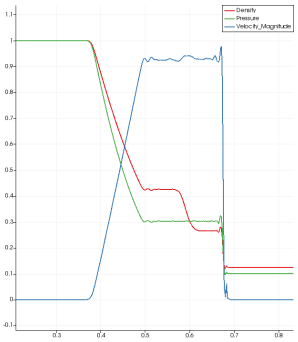


FIGURE 12 – CENO O4 with partition-based molecules at physical time 0.1 sec. Navier-Stokes Equations with viscosity = 0.001

9.2 Half cylinder flow

The flow past a cylinder at Mach number 0.3 is also computed. In order to keep a computation sufficiently stable we also use as entries of the Roe approximate Riemann solver the projection of left and right densities (resp. total energy) into the interval $[min(\rho_L, \rho_R), max(\rho_L, \rho_R)]$ (resp. $[min(\rho_L E_L, \rho_R E_R), max(\rho_L E_L, \rho_R E_R)]$. The number of vertices of the 3D mesh is 52662 and of tetrahedra 260383. the number of plan in span is 10. For all approximations, explicit time advancing with CFL=0.5 and implicit time advancing with CFL=2. produced the same steady solution, starting from a uniform flow (basic TV4 and TV6 schemes of NiceFlow with limiters) of, for CENO, from steady solutions obtained with TV4.

9.2.1 Mach number analysis

We compare in Figures 13, 14, and 15 the results obtained by the different spatial schemes. The results are very similar, with a slightly larger maximum of the CENO O4 ($0.24 > 0.23$).

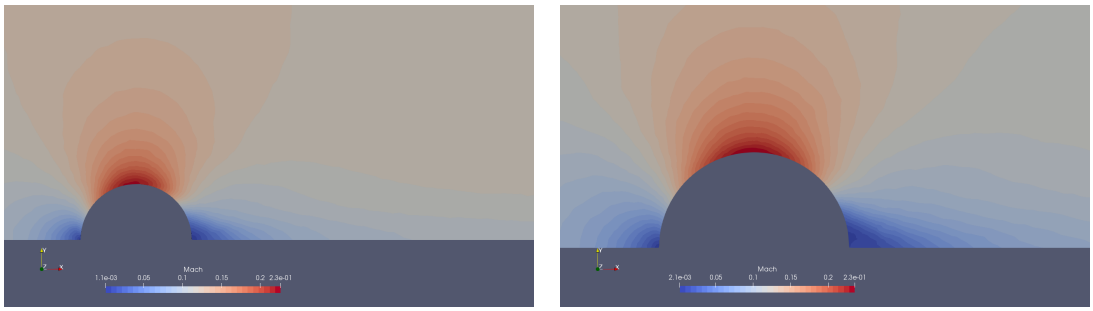


FIGURE 13 – Cylinder flow. Mach number : left, CENO O3, right, CENO O3 with partition

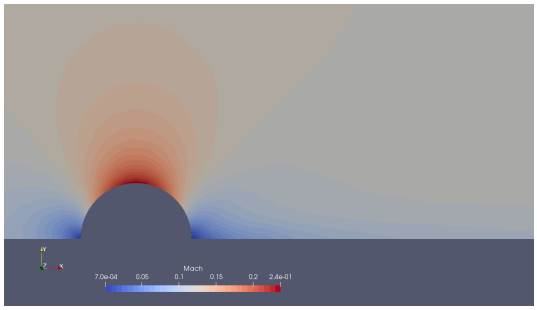


FIGURE 14 – Cylinder flow. Mach number : CENO O4.

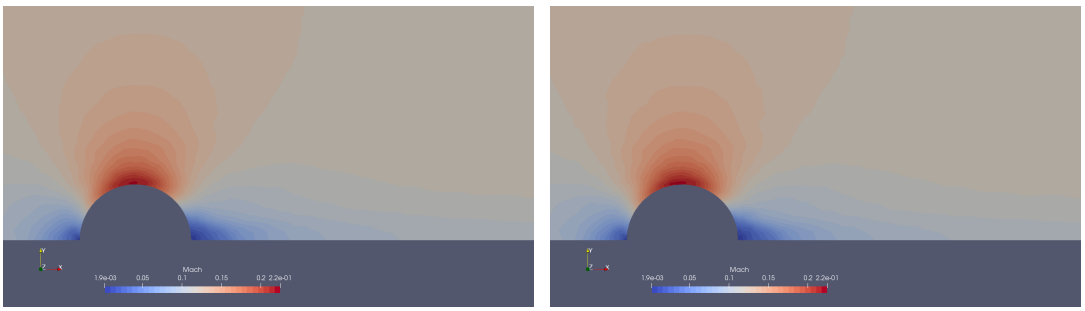


FIGURE 15 – Cylinder flow. Mach number : left, TV4, right, TV6.

9.2.2 Pressure analysis

Pressure contours show some difference. The accuracy appears as the ability in producing left-right symmetric pressure contours. While TV4 and TV6 give a poor pressure in rear part, due to a still coarse mesh, CENO O3 give a neat improvement and CENO O4 show a nearly perfect symmetry.

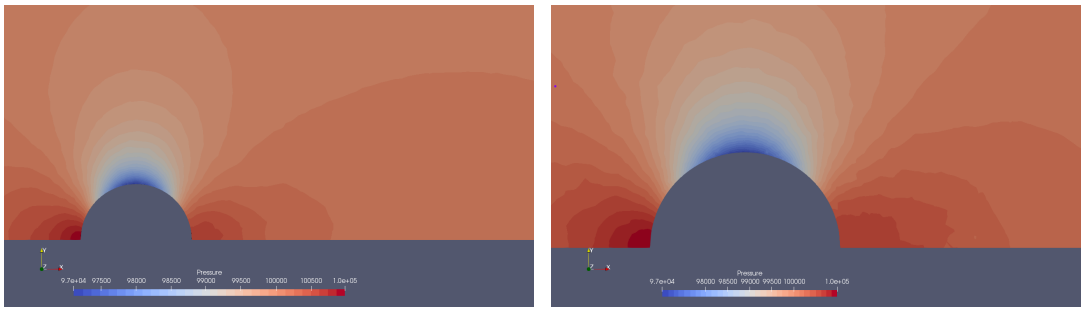


FIGURE 16 – Cylinder flow. Pressure : left, CENO O3, right, CENO O3 with Partition

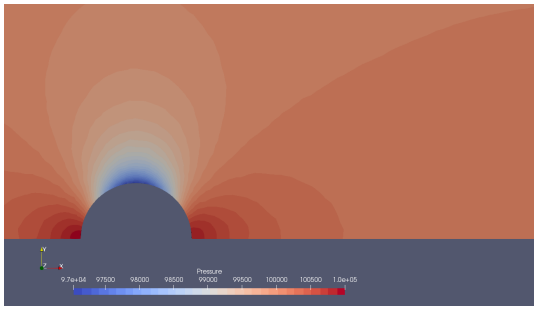


FIGURE 17 – Cylinder flow. Pressure : CENO O4

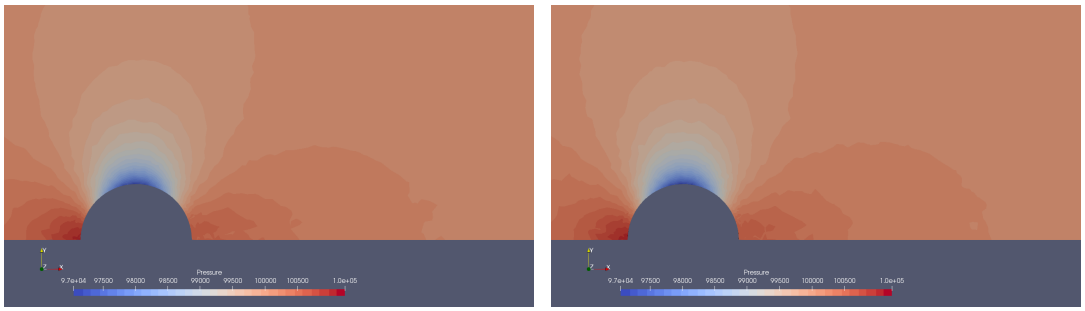


FIGURE 18 – Cylinder flow. Pressure : left, TV4, right, TV6.

9.2.3 Entropy analysis

The entropy deviation $(s - s_{inflow})$ is zero for the exact solution and also measures dissipation and arising of spurious rotational. We observe that the partition option again produces slightly oscillatory contours for this entropy. The CENO O4 gives smaller entropy errors ay front part of the cylinder, but of both positive and negative signs, due to a lack of monotony.

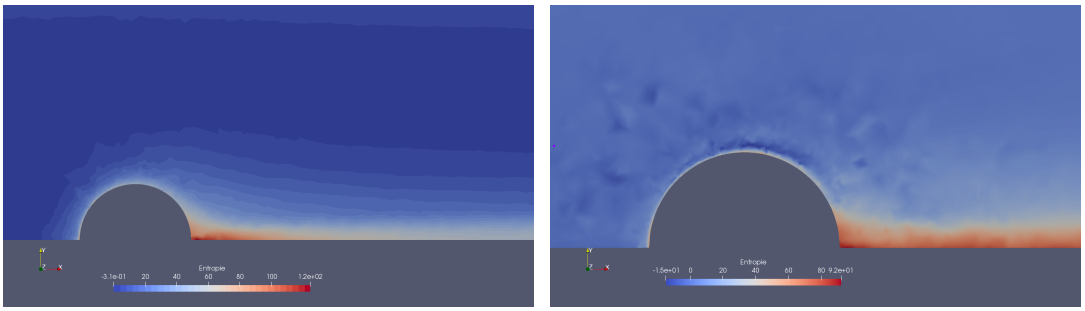


FIGURE 19 – Cylinder flow. Entropie : left, CENO O3, right, CENO O3 with Partition.

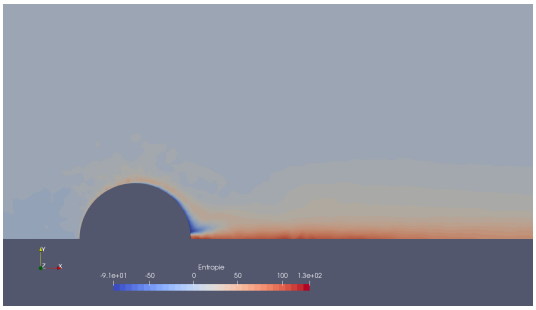


FIGURE 20 – Cylinder flow. Entropie CENO O4 at convergence state

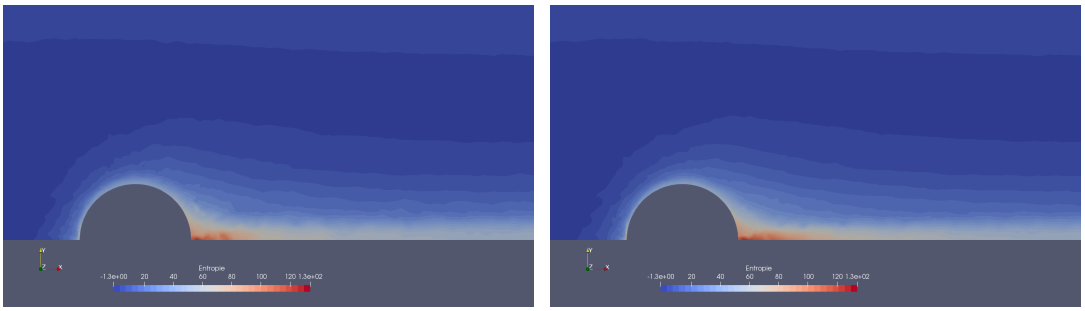


FIGURE 21 – Cylinder flow. Entropie : left, TV4, right, TV6

9.3 Cylinder flow CPU analysis

For the moment, we can propose only a very unfair comparison between the TV4/TV6 approximations of NiceFlow, which are optimized, and the first implementation of CENO. The number of processors is four.

Scheme	Pre-processing	Explicit time step CFL 0.5	Implicit time step CFL 2
Ceno O3	14 sec	6.55	5.1
Ceno O3 partition	20.2 sec	5.64	4.0
Ceno O4	21 sec	6.45	4.2
TV4	—	0.1	n.a.
TV6	—	0.123	n.a.

10 Concluding remarks

The vertex-centered CENO method has been revisited, with two ways in generating reconstruction molecules. Numerical tests with an advection model show the validity of both approaches. Although the asymptotic accuracy appears later (in terms of number of cells) for the partition macromolecules than for the centered macromolecules, third order accuracy is well approached for both and the partition macromolecules imply a much smaller computational effort for reconstruction. Numerical experiments with the Euler equation in NiceFlow show that the higher-order schemes give much better solutions than the second-order approximations of NiceFlow. The internal flux remain the most costly part, and its optimization will be addressed in a future work.

11 Acknowledgements

This work is done in the ANR project NORMA which is supported by the french ministery of Research under contract ANR-19-CE40-0020-01.

The integration in NiceFlow was partly supported by Lemma. The author gratefully thanks Didier Chargy (Lemma) for his strong help in this integration.

Références

[1] R. Abgrall. Design of an essentially non-oscillatory reconstruction procedure on Finite-Element type meshes. *Rapport technique 1584, INRIA*, 1992.

[2] R. Abgrall. Residual distribution schemes : current status and future trends. *Computers and Fluids*, 35 :641–669, 2006.

[3] T.J. Barth and P.O. Frederickson. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. *AIAA-90-0013*, 1990.

[4] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2D Euler equations. *Journal of Computational Physics*, 138(2) :251–285, 1997.

[5] A. Carabias. *Analyse et adaptation de maillage pour des schémas non-oscillatoires d'ordre élevé*. PhD thesis, Université de Nice - Sophia Antipolis, Nice, France, 2013.

[6] P.G. Ciarlet and P.A. Raviart. General Lagrange and Hermite interpolation in R^n with applications to Finite Element methods. *Archive for Rational Mechanics and Analysis*, 46 :177–199, 1972.

[7] B. Cockburn. Devising Discontinuous Galerkin methods for non-linear hyperbolic conservation laws. *Journal of Computational and Applied Mathematics*, 128(1-2) :187–204, 2001.

[8] B. Cockburn, G. Karniadakis, and C.-W. Shu. *Discontinuous Galerkin methods : theory, computation and application.* Lecture Notes in Computational Science and Engineering. Springer Verlag, Berlin, 2000.

[9] S. Engquist, B. Harten, A. Osher, and S.R. Chakravarthy. Some results on uniformly high-order accurate essentially non oscillatory schemes. *Appl. Numer. Math. 2(3-5) :347-377*, 1986.

[10] A. Harten et S. Chakravarthy. Multi-dimensional ENO schemes for general geometries. *ICASE report 91-76*, 1991.

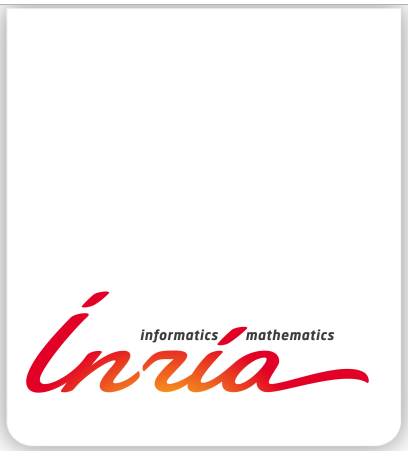
[11] C.P.T. Groth and L. Ivan. High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition AIAA 2011-367 4 - 7 January 2011, Orlando, Florida*, 2011.

[12] L. Ivan and C.P.T. Groth. High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. *J. Comp. Phys.*, 257 :830–862, 2014.

[13] F.C. Lafon and R. Abgrall. ENO schemes on unstructured meshes. *Rapport INRIA 2099*, 1993.

[14] C. Ollivier-Gooch, A. Nejat, and K. Michalak. Obtaining and verifying high-order unstructured finite volume solutions to the Euler equations. *AIAA Journal*, 47(9) :2105–2120, 2009.

[15] C.W. Shu and B. Cockburn. Runge-Kutta Discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3) :173–261, 2001.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399