Discontinuous Galerkin methods for solving acoustic and elastodynamic wave equation

Caroline BALDASSARI

Equipe-projet INRIA Bordeaux Sud-Ouest Magique3D
Total

H. Barucq, H. Calandra, B. Denel and J. Diaz
December 15th 2010
Objectives

- Find a fast and accurate numerical method which can be applied for terrestrial migration;
- Write a 2D and 3D migration code;
- Optimize this code by reducing the computational cost.
Outline

- Introduction to seismic depth imaging
- Numerical methods for the wave equation
- Performance analysis of the Interior Penalty Discontinuous Galerkin method for the acoustic wave equation
- Migration results
- New time discretization method
- Centered flux Discontinuous Galerkin method for the elastodynamic wave equation
- Numerical results
- Perspectives
Seismic reflection basis
Seismic reflection basis
Take the **topography** effects into account
Seismic exploration

Recording echoes
Seismic exploration

Searching for reflectivity parameters

Velocity 1

Velocity 2

Velocity 3

P

Time
The Reverse time migration

1. Using same sources and receivers as for the acquisition campaign: propagate the sources into the velocity model
2. Retropropagate the recorded waves (reverse time)
3. Cross-correlation of the two fields: imaging condition (Claerbout)

Produce an image of the subsurface
The Reverse time migration

1. Using same sources and receivers as for the acquisition campaign: propagate the sources into the velocity model.

2. Retropropagate the recorded waves (reverse time).

3. Cross-correlation of the two fields: imaging condition (Claerbout).

   Produce an image of the subsurface.

Step 1 and 2 ⇒ solve the wave equation twice for each source.
Numerical methods for solving the full-wave equation

- Finite difference methods:
  - ✓ The representation of the solution is explicit;
  - ✓ numerical dispersion in strongly heterogeneous medium or with coarse grids;
  - ✓ not well adapted to take the topography effects into account.
Numerical methods for solving the full-wave equation

- **Finite difference methods**:
  - ✓ The representation of the solution is explicit;
  - ✓ numerical dispersion in strongly heterogeneous medium or with coarse grids;
  - ✓ not well adapted to take the topography effects into account.

- **Finite element methods**:
  - ✓ Efficient to take the topography effects into account;
  - ✓ optimized computational burden;
  - ✓ limitation of the dispersion effects;
  - ✓ the mass matrix is generally **not diagonal** without mass-lumping
    ⇒ the order of convergence of the method is penalized.
The Spectral Element method (Cohen, Joly, Roberts, Tordjman, Komatitsch,...)

✓ Gauss Lobatto quadrature rule: diagonal mass matrix, does not hamper the order of convergence;
✓ meshes made of quadrangles in 2D or hexahedra in 3D: may be difficult to compute, not always suitable for complex topographies.
Numerical methods for solving the full-wave equation

1. The Spectral Element method (Cohen, Joly, Roberts, Tordjman, Komatitsch, ...)
   - ✔ Gauss Lobatto quadrature rule: diagonal mass matrix, does not hamper the order of convergence;
   - ✔ meshes made of quadrangles in 2D or hexahedra in 3D: may be difficult to compute, not always suitable for complex topographies.

2. The Discontinuous Galerkin method (DG)
   - ✔ Representation of the solution is quasi-explicit because the mass matrix is block-diagonal without any approximation;
   - ✔ order of convergence is preserved;
   - ✔ polynomial velocities inside each element;
   - ✔ meshes made of triangles in 2D or tetrahedra in 3D. Thus the topography of the computational domain is easily discretized.
• To use discontinuous functions: we are then able to mix different orders of approximation

• To apply a stable numerical method

• The IPDG approach seems to be a good candidate
We consider the problem:

\[
\begin{aligned}
\frac{1}{\mu} \frac{\partial^2 u}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho} \nabla u \right) &= f \quad \text{in} \quad \Omega \times I \\
\nabla u \cdot \mathbf{n} &= 0 \quad \text{on} \quad \Gamma_N \\
\frac{1}{\sqrt{\mu}} \frac{\partial u}{\partial t} + \frac{1}{\sqrt{\rho}} \nabla u \cdot \mathbf{n} &= 0 \quad \text{on} \quad \Gamma_{abs} \\
u(x, 0) = \frac{\partial u}{\partial t}(x, 0) &= 0 \quad \text{in} \quad \Omega
\end{aligned}
\]
• $\Omega$: bounded domain;
• $I = ]0, T[$ is a time finite interval;
• the solution $u$ is a $C^0(]0, T[, H^1) \cap C^1(]0, T[, L^2)$ function;
• $f$ is a $C^0(]0, T[, L^2(\Omega))$ function;
• $\rho$ is the density and $\mu$ the compressibility modulus.
• $\mathcal{T}_h$: mesh of $\Omega$;
• $V_l^h := \{ \varphi \in L^2(\Omega): \varphi|_K \in P_l(K) \ \forall K \in \mathcal{T}_h \}$
• $P_l(K)$: space of polynomials of degree at most $l \geq 1$ in $K$;
• $(\varphi_k)_{k=1,...N_{fb}}$: Lagrange functions of degree $l$ in $P_l(K)$;
• $u_h = \sum_{j=1}^{N} u_{h,j} \varphi_j$; $N$: nb of d.o.f

$$\mathcal{M} \frac{d^2 U_h}{dt^2} + \mathcal{B} \frac{dU_h}{dt} + \mathcal{K} U_h = \mathcal{F}_h \quad (1)$$

• The coefficients of the vector $U_h$ are the components $u_{h,j}$;
• $\mathcal{F}_h$ is the source vector with coefficients $\mathcal{F}_i = \int_{\Omega} f \varphi_i \, dx.$
The matrix $\mathcal{M}$ is the block diagonal mass matrix;

$$\mathcal{M}_{ij} = \int_{\Omega} \frac{1}{\mu} \varphi_i \varphi_j \, dx;$$

the damping matrix $\mathcal{B}$ is null except for the element of $\Gamma_{abs}$;

$$\mathcal{B}_{ij} = \int_{\Gamma_{abs}} \frac{1}{\sqrt{\mu \rho}} \varphi_i \varphi_j \, ds;$$

the matrix $\mathcal{K}$ is the symmetrical stiffness matrix.

$$\mathcal{K}_{ij} = \int_{K} \frac{1}{\rho} \nabla \varphi_i \cdot \nabla \varphi_j \, dx - \sum_{F} \int_{F} [[ \varphi_j ]] \cdot \left\{ \left\{ \frac{1}{\rho} \nabla \varphi_i \right\} \right\} \, ds$$

$$- \sum_{F} \int_{F} [[ \varphi_i ]] \cdot \left\{ \left\{ \frac{1}{\rho} \nabla \varphi_j \right\} \right\} \, ds + \sum_{F} \int_{F} \gamma [[ \varphi_i ]] \cdot [[ \varphi_j ]] \, ds$$
Interior Penalty Discontinuous Galerkin Method (IPDG)

\[
[q] = q^+ n^+ + q^- n^- \quad \text{et} \quad \{q\} := \frac{(q^+ + q^-)}{2};
\]

• \( n^\pm \) normal vectors of \( K^\pm \);

• the function \( \gamma \) penalizes the jump of \( u_h \) and \( v_h \) on the edges (or faces) of \( T_h \). It is called interior penalty function. It is defined on each face by:

\[
\gamma|_F := \alpha c_{max} h_{min}^{-1},
\]

where \( \alpha \) is a positive parameter independent of the mesh and the coefficient \( c_{max} \).
Arnold, Brezzi, Cockburn and Marini : Unified analysis of discontinuous Galerkin methods for elliptic problems

Ainsworth, Monk and Muniz : Dispersive and Dissipative Properties of Discontinuous Galerkin Finite Element Methods for the second-Order Wave Equation

Grote, Schneebeli and Schötzau : Discontinuous Galerkin finite element method for the wave equation
• Time discretization: leap-frog scheme
• Time discretization: leap-frog scheme

\[
\mathcal{M} \frac{U_h^{n+1} - 2U_h^n + U_h^{n-1}}{\Delta t^2} + \mathcal{B} \frac{U_h^{n+1} - U_h^{n-1}}{2\Delta t} = -KU_h^n + \mathcal{F}_h
\]
• Time discretization: leap-frog scheme

\[ M \frac{U_{h}^{n+1} - 2U_{h}^{n} + U_{h}^{n-1}}{\Delta t^2} + B \frac{U_{h}^{n+1} - U_{h}^{n-1}}{2\Delta t} = -KU_{h}^{n} + F_{h} \]

This scheme is stable under a CFL condition which depends on the penalty parameter.

We compare IPDG with:

1. FD: most used in petroleum engineering;
2. SEM: applied with success to the wave propagation.
What are the good criteria to compare the methods?

- Compare the computational burden for a given accuracy.
What are the good criteria to compare the methods?

- Compare the computational burden for a given accuracy.
- Compare the accuracy for a given computational burden.
What are the good criteria to compare the methods?

- Compare the computational burden for a given accuracy.
- Compare the accuracy for a given computational burden.
  The computation cost of one iteration is directly related to the number of degrees of freedom.
What are the good criteria to compare the methods?

- Compare the computational burden for a given accuracy.
- Compare the accuracy for a given computational burden.

The computation cost of one iteration is directly related to the number of degrees of freedom.

⇒ compare the accuracy for a given number of degrees of freedom
Conclusion:

- Whatever the order of the method, SEM and IPDG are much more accurate than the FD;
- for low order element \((l = 1)\) SEM is better than IPDG;
- for high order element \((l \geq 2)\) SEM and IPDG perform similarly.
Performance of IPDG — 2D Tests

- \([0 \ 21600] \times [0 \ 24470]\)
- \(C_1=1600\text{m/s},\ C_2=2400\text{m/s}\)
- Source position: \((10800, 14470)\)
- First derivative of a Gaussian
- \(f = 2\text{Hz}\)
- Absorbing boundary condition
- Position of the receivers: \((10\times i, 14470)\)
IPDG/FD/SEM results — Comparison with the exact solution

Mesh : $14 \frac{pts}{\lambda}$

**Fig.**: Receiver at 5100m from the source
IPDG/FD/SEM results — Convergence

![Graph showing convergence of IPDG/FD/SEM results](image)
Performance of IPDG — 3D Tests

- $[0 \ 600] \times [0 \ 600] \times [0 \ 600]$
- $C_1=100\text{m/s}, \ C_2=200\text{m/s}$
- Source position : $(300,300,350)$
- First derivative of a Gaussian
- $f = 2\text{Hz}$
- Absorbing boundary condition
- 16 receivers
Performance of IPDG — 3D Tests

- $[0 \ 600] \times [0 \ 600] \times [0 \ 600]$ 
- $C_1=100\text{m/s}, \ C_2=200\text{m/s}$ 
- Source position: (300, 300, 350) 
- First derivative of a Gaussian 
- $f = 2\text{Hz}$ 
- Absorbing boundary condition 
- 16 receivers
Mesh: 412913 elements, $\approx 19 \, \text{pts}/\lambda$

**Fig.:** Comparaison between the exact solution and IPDG

**Fig.:** Difference between the exact solution and IPDG
2D Reverse Time Migration results — First experiment

- 100 sources on the top
- First derivative of a Gaussian
- $f = 10\text{Hz}$
- Dirichlet condition on the top, absorbing boundary condition elsewhere
- Receivers on the top
Result of the propagation
Result of the propagation
Result of the propagation
Result of the propagation
Result of the propagation
Result of the RTM
2D Reverse Time Migration results — Second experiment

Experiment with synthetic data obtained by another computational code

- 500 sources on the top
- First derivative of a Gaussian
- $f = 10\text{Hz}$
- Dirichlet condition on the top, absorbing boundary condition elsewhere
- Receivers on the top
Results of the RTM
Optimization of the code — Adaptation of the size of the computational domain

Fig.: Velocity model for the source localized at \( x = 750 \text{m} \)

Fig.: Velocity model for the source localized at \( x = 6000 \text{m} \)

Fig.: Velocity model for the source localized at \( x = 12375 \text{m} \)
Adaptation of the size of the computational domain — RTM result
Adaptation of the size of the computational domain — RTM result

- CPU time for the first mesh: 2h 30 min
- CPU time for the adapted mesh: 1h 27 min (42%)
Optimization of the code — Adaptation of the mesh

- $C_1 = 1500\text{m/s}, \ C_2 = 3000\text{m/s}$
- Source position : $(300,600)$
- First derivative of a Gaussian
- $f = 20\text{Hz}$
- Dirichlet condition on the top, absorbing boundary condition elsewhere
- First line of receivers at 600m
- Second line of receivers at 400m
Adaptation of the mesh: three different meshes

Maillage de 65107 mailles, 32926 noeuds et 98032 arêtes

Maillage de 21250 mailles, 10680 noeuds et 32109 arêtes

Maillage de 15202 mailles, 7790 noeuds et 22369 arêtes
Results for the first line of receivers

Fig.: Mesh n°1

Fig.: Mesh n°2

Fig.: Mesh n°3
Results for the second line of receivers

**Fig. 1**: Mesh n°1

**Fig. 2**: Mesh n°2

**Fig. 3**: Mesh n°3
Adaptation of the mesh

- CPU time for the mesh n°1 : 50 min
Adaptation of the mesh

- CPU time for the mesh n°1 : 50 min
- CPU time for the mesh n°2 : 8 min 20 s (83%)
Adaptation of the mesh

The number of points per $\lambda$ is function of

- the size of the elements;
- the velocity;
- the frequency of the source.

So, if the number of points per $\lambda$ is fixed ($\approx 8 - 10$) we can evaluate the size of the mesh in the different parts of the domain.
Adaptation of the mesh — RTM result
Adaptation of the mesh — RTM result

- CPU time for the first mesh: 2h 30 min
- CPU time for the adapted mesh: 1h 25 min (45%)
Mixing the two optimizations — RTM result
Mixing the two optimizations — RTM result

- CPU time for the first mesh: 2h 30 min
- CPU time for the adapted mesh: 50 min (66%)
Optimization of the code — Adaptation of the element order
Optimization of the code — Adaptation of the element order
Adaptation of the element order — Results for the first line of receivers

**Fig.:** $\mathcal{P}^3$-elements everywhere

**Fig.:** $\mathcal{P}^1$-elements on the top, $\mathcal{P}^3$-elements elsewhere
Adaptation of the element order — Results for the second line of receivers

**Fig.**: $\mathcal{P}^3$-elements everywhere

**Fig.**: $\mathcal{P}^1$-elements on the top, $\mathcal{P}^3$-elements elsewhere
Adaptation of the element order
$P^1$-elements on the top, $P^3$-elements elsewhere
Improvements of the method

- CFL not optimized: use a local time stepping strategy
- Use $P_1$ polynomials in the fine mesh and $P_3$ polynomials in the coarse mesh.
- Use a second order time scheme in the fine mesh and a fourth order time scheme in the coarse mesh.
Improvements of the method

- CFL not optimized: use a local time stepping strategy
Improvements of the method

- CFL not optimized: use a local time stepping strategy
- Use $P_1$ polynomials in the fine mesh and $P_3$ polynomials in the coarse mesh
Improvements of the method

- CFL not optimized: use a local time stepping strategy
- Use $P_1$ polynomials in the fine mesh and $P_3$ polynomials in the coarse mesh
- Use a second order time scheme in the fine mesh and a fourth order time scheme in the coarse mesh.
Local Time stepping - bibliography

- Bécache, Collino, Fouquet, Joly, Rodríguez
- Piperno
• Bécache, Collino, Fouquet, Joly, Rodríguez

• Piperno

• ADER Schemes, Käser, Dumbser et al
Local Time stepping - bibliography

- Bécache, Collino, Fouquet, Joly, Rodríguez
- Piperno
- ADER Schemes, Käser, Dumbser et al
  - Conservation of energy
  - Optimal stability condition
At each time step $n$ we define an auxiliary function

$$Q_n(\tau) = \frac{U(n\Delta t - \tau) + U(n\Delta t + \tau)}{2}$$

for $\tau \in [-\Delta t; \Delta t]$. 
At each time step \( n \) we define an auxiliary function

\[
Q_n(\tau) = \frac{U(n\Delta t - \tau) + U(n\Delta t + \tau)}{2}
\]

for \( \tau \in [-\Delta t; \Delta t] \).

This function is obviously even and satisfies:

\[
\begin{align*}
\frac{d^2Q_n}{d\tau^2}(\tau) &= -AQ_n(\tau), \\
Q_n(0) &= U(n\Delta t), \\
\frac{dQ_n}{d\tau}(0) &= 0,
\end{align*}
\]
At each time step $n$ we define an auxiliary function

$$Q_n(\tau) = \frac{U(n\Delta t - \tau) + U(n\Delta t + \tau)}{2}$$

for $\tau \in [-\Delta t; \Delta t]$.

This function is obviously even and satisfies:

$$\begin{cases} 
\frac{d^2Q_n}{d\tau^2}(\tau) = -AQ_n(\tau), \\
Q_n(0) = U(n\Delta t), \quad \frac{dQ_n}{d\tau}(0) = 0,
\end{cases}$$

After having solved this equation, $U((n + 1)\Delta t)$ can be computed using $U((n + 1)\Delta t) = -U((n - 1)\Delta t) + 2Q_n(\Delta t)$. 
Local Time-Stepping

\[ \frac{d^2 Q_n}{d\tau^2} + AQ_n = 0 \]
Local Time-Stepping

\[
\frac{d^2 Q_n}{d\tau^2} + AQ_n = 0
\]

Let us now split \( Q_n \) in two parts:

\[
Q_n = \begin{bmatrix} Q^g_n \\ Q^f_n \\ Q^n_n \end{bmatrix}
\]
Local Time-Stepping

\[ \frac{d^2 Q_n}{d\tau^2} + AQ_n = 0 \]

Let us now split \( Q_n \) in two parts:

\[
Q_n = \begin{bmatrix} Q_{n}^{g} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ Q_{n}^{f} \end{bmatrix}
\]
Local Time-Stepping

\[ \frac{d^2 Q_n}{d\tau^2} + AQ_n = 0 \]

Let us now split \( Q_n \) in two parts:

\[ Q_n = \begin{bmatrix} Q^g_n \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ Q^f_n \end{bmatrix} = (I - P)Q_n + PQ_n, \text{ with } P^2 = P \]
Local Time-Stepping

\[
\frac{d^2 Q_n}{d\tau^2} + A(I - P)Q_n + APQ_n = 0
\]

Let us now split \( Q_n \) in two parts:

\[
Q_n = \begin{bmatrix} Q^g_n \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ Q^f_n \end{bmatrix} = (I - P)Q_n + PQ_n, \text{ with } P^2 = P
\]
Local Time-Stepping

\[
\frac{d^2 Q_n}{d\tau^2} + A(I - P)Q_n + APQ_n = 0
\]

Idea:
Approximate \(A(I - P)Q_n(\tau)\) by

\[
A(I - P)Q_n(\tau) \approx A(I - P)Q_n(0) + \frac{\tau^2}{2} A(I - P)\frac{d^2 Q_n(0)}{d\tau^2}
\]
Local Time-Stepping

\[
\frac{d^2 Q_n}{d\tau^2} + A(I - P)Q_n + APQ_n = 0
\]

Idea:

Approximate \( A(I - P)Q_n(\tau) \) by

\[
A(I - P)Q_n(\tau) \approx A(I - P)U(n\Delta t) - \frac{\tau^2}{2} A(I - P)AU(n\Delta t)
\]
Local Time-Stepping

\[ \frac{d^2 Q_n}{d\tau^2} + A(I - P)Q_n + APQ_n = 0 \]

Idea:

Approximate \( A(I - P)Q_n(\tau) \) by

\[ A(I - P)Q_n(\tau) \approx A(I - P)U(n\Delta t) - \frac{\tau^2}{2} A(I - P)AU(n\Delta t) \]

So that \( Q_n \) is the solution to

\[ \frac{d^2 Q_n(\tau)}{d\tau^2} + A(I - P)U(n\Delta t) - \frac{\tau^2}{2} A(I - P)AU(n\Delta t) + APQ_n(\tau) = 0 \]

\[ Q_n(0) = U(t) \]

\[ Q'_n(0) = 0 \]
We solve

\[
\frac{d^2}{d\tau^2} Q_n(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P)AU^n + APQ_n(\tau) = 0
\]

\[Q_n(0) = U^n\]

\[Q'_n(0) = 0\]

from \(\tau = 0\) to \(\tau = \Delta t\), using a fourth order Modified Equation scheme with a time step \(\frac{\Delta t}{p}\).
We solve
\[
\frac{d^2}{d\tau^2} Q_n(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P)AU^n + APQ_n(\tau) = 0
\]
\[
Q_n(0) = U^n
\]
\[
Q'_n(0) = 0
\]

from $\tau = 0$ to $\tau = \Delta t$, using a fourth order Modified Equation scheme with a time step $\frac{\Delta t}{p}$.

- The scheme conserves an energy;
- the scheme is stable if $\Delta t < h_g$ and $\frac{\Delta t}{p} < h_f$. 
Algorithm of a 4/2 order local time-stepping scheme

New method: we consider the equation

$$\frac{d^2 Q_n}{d\tau^2}(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P)!AU + APQ_n(\tau) = 0$$
New method: we consider the equation

\[
\frac{d^2 Q_n}{d\tau^2}(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P)A U^n + APQ_n(\tau) = 0
\]

and we discretize

\[
(I - P) \frac{d^2}{d\tau^2} Q_n(\tau)
\]

by a fourth order Modified equation scheme
Algorithm of a 4/2 order local time-stepping scheme

New method: we consider the equation

$$\frac{d^2 Q_n}{d\tau^2}(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P) AU^n + APQ_n(\tau) = 0$$

and we discretize

$$(I - P)\frac{d^2}{d\tau^2} Q_n(\tau)$$

by a fourth order Modified equation scheme and

$$P\frac{d^2}{d\tau^2} Q_n(\tau)$$

by a second order Leap-Frog scheme
Algorithm of a 4/2 order local time-stepping scheme

New method: we consider the equation

\[
\frac{d^2 Q_n}{d\tau^2}(\tau) + A(I - P)U^n - \frac{\tau^2}{2} A(I - P)AU^n + APQ_n(\tau) = 0
\]

We have to solve

\[
\frac{Q_{n+1}^p - 2Q_i^p + Q_{i-1}^p}{\Delta\tau^2} = -A(I - P)U^n + \frac{\tau^2}{2} A(I - P)AU^n - APQ_i^p + \frac{\Delta\tau^2}{12} (I - P) A(I - P)AU^n
\]
Numerical experiment — 1D

\[ u_0(x) = (x - x_0) e^{-\frac{4(x-x_0)^2}{\pi^2 r_0^2}}, \quad u_1(x) = \left( \frac{8 (x - x_0)}{\pi^2 r_0^2} - 1 \right) e^{-\frac{4(x-x_0)^2}{\pi^2 r_0^2}} \]

- \( x_0 = 2, \; r_0 = 2 \);
- \( c = 1 \text{ms}^{-1} \);
- \( L = 0.1, \; h_f = 1/320 \) and \( h_g = [0.2, 0.1, 0.05, 0.025] \).
Numerical results — 1D

**Fig.** Convergence results for the scheme EM\_42 ($L = 0.1$ et $h_f = 1/320$).

**Fig.** Convergence results for the scheme Loc\_42\_42 ($L = 0.1$ et $h_f = 1/320$).
Numerical experiment — 2D
$\mathcal{P}^1$-elements on the top, $\mathcal{P}^3$-elements elsewhere
Elastodynamic

\[
\begin{aligned}
\rho(\vec{x}) \partial_t \vec{v}(\vec{x}, t) &= \nabla \cdot \sigma(\vec{x}, t) \quad \text{dans} \quad \Omega \times [0, T], \\
\partial_t \sigma(\vec{x}, t) &= C(\vec{x}) \varepsilon(\vec{v}(\vec{x}, t)) \quad \text{dans} \quad \Omega \times [0, T]
\end{aligned}
\]

- \( \vec{x} \in \mathbb{R}^3 \)
- \( \vec{v} = (v_x, v_y, v_z)^t \)
- \( \rho \) is the density.
- \( \varepsilon \) is defined by: \( \varepsilon_{ij} = \frac{1}{2} \left( \frac{\partial v_i}{\partial j} + \frac{\partial v_j}{\partial i} \right), \quad i, j = x, y, z \)
- \( \sigma \) is defined by, (isotropic case): \( \sigma_{ij} = \lambda \delta_{ij} \text{tr}(\varepsilon) + 2\mu \varepsilon_{ij}, \quad i, j = x, y, z. \)
- \( C \) is defined by the Lamé parameter \( \lambda \) and \( \mu \)
- Boundary conditions: \( \sigma \vec{n} = 0 \) on \( \partial \Omega \).
We consider:

- $\mathcal{T}_h$ a mesh of $\Omega$ composed by tetrahedra $K$
- finite element space

$$V^h := \{ v \in L^2(\Omega) : v|_K \in P(K), \forall K \in \mathcal{T}_h \}$$
Space discretization

We consider:

- $\mathcal{T}_h$ a mesh of $\Omega$ composed by tetrahedra $K$
- finite element space

$$V^h := \{ v \in L^2(\Omega) : v|_K \in P(K), \forall K \in \mathcal{T}_h \}$$

- We multiply the first equation by $\vec{w}$, and the second by $\xi$
Space discretization

We consider:

- $\mathcal{T}_h$ a mesh of $\Omega$ composed by tetrahedra $K$
- finite element space

$$V^h := \{ v \in L^2(\Omega) : v|_K \in P(K), \quad \forall K \in \mathcal{T}_h \}$$

- We multiply the first equation by $\vec{w}$, and the second by $\xi$
- We integrate by part over each element $K$
Space discretization

We consider:

- $\mathcal{T}_h$ a mesh of $\Omega$ composed by tetrahedra $K$
- finite element space

$$V^h := \{ v \in L^2(\Omega) : v|_K \in P(K), \forall K \in \mathcal{T}_h \}$$

- We multiply the first equation by $\vec{w}$, and the second by $\xi$
- We integrate by part over each element $K$
- We sum over all the elements $K$
Space discretization

- We denote by $[.]$ and respectively $\{.\}$ the jump and the mean over a face of $K$. 
Space discretization

- We denote by $[.]$ and respectively $\{.\}$ the jump and the mean over a face of $K$.
- We use the continuity of $\overline{\sigma \cdot n}$ and we have on each face:

$$\overline{[\sigma \cdot n \cdot w]} = \overline{[\sigma \cdot n]} \cdot \{\overline{w}\} + \{\overline{\sigma \cdot n}\} \cdot [\overline{w}] = \{\overline{\sigma \cdot n}\} \cdot [\overline{w}]$$
Space discretization

- We denote by \([\cdot]\) and respectively \(\{\cdot\}\) the jump and the mean over a face of \(K\).
- We use the continuity of \(\overline{\sigma n}\) and we have on each face:
  \[
  [\overline{\sigma n} \cdot \overrightarrow{w}] = [\overline{\sigma n}] \cdot \{\overrightarrow{w}\} + \{\overline{\sigma n}\} \cdot [\overrightarrow{w}] = \{\overline{\sigma n}\} \cdot [\overrightarrow{w}]
  \]
- In the same way, using the continuity of \(\overrightarrow{v}\), we have:
  \[
  [\overrightarrow{v} \cdot C_K \xi \overline{n}] = [\overrightarrow{v} \cdot C_K \xi] \cdot \{\overrightarrow{v}\}
  \]
Space discretization

- We denote by $[.]$ and respectively $\{.\}$ the jump and the mean over a face of $K$.

- We use the continuity of $\overrightarrow{\sigma n}$ and we have on each face:

$$[\overrightarrow{\sigma n} \cdot \overrightarrow{w}] = [\overrightarrow{\sigma n}] \cdot \overrightarrow{w} + \overrightarrow{\sigma n} \cdot [\overrightarrow{w}] = \overrightarrow{\sigma n} \cdot [\overrightarrow{w}]$$

- In the same way, using the continuity of $\overrightarrow{v}$, we have:

$$[\overrightarrow{v} \cdot C_{K} \xi \overrightarrow{n}] = [\overrightarrow{v} \cdot C_{K} \xi] \cdot \{\overrightarrow{v}\}$$

And finally we get:

$$\begin{cases} 
\int_{\Omega} \rho \partial_{t} \overrightarrow{v} \cdot \overrightarrow{w} + \int_{\Omega} \overrightarrow{\sigma} : \nabla \overrightarrow{w} - \int_{\Gamma^{0}} \{\overrightarrow{\sigma n}\} \cdot [\overrightarrow{w}] = 0, \\
\int_{\Omega} \partial_{t} \overrightarrow{\sigma} : \xi + \int_{\Omega} \overrightarrow{v} \cdot \nabla \cdot (C_{K} \xi) - \int_{\Gamma^{0}} [\overrightarrow{v} \cdot C_{K} \xi] \cdot \{\overrightarrow{v}\} = 0
\end{cases}$$
We obtain the linear system:

\[
\begin{align*}
\rho \mathcal{M} \partial_t \vec{V}_h + \mathcal{K}_\sigma \overline{S}_h &= 0, \\
\mathcal{M} \partial_t \overline{S}_h + \mathcal{K}_v \vec{V}_h &= 0
\end{align*}
\] (2)

- the matrix \( \mathcal{M} \) is the diagonal mass matrix;
- The matrices \( \mathcal{K}_\sigma \) and \( \mathcal{K}_v \) are the stiffness matrices.
Time discretization

We use a Leap-frog scheme:

\[
\begin{align*}
  \rho M \frac{\vec{V}_{h}^{n+1} - \vec{V}_{h}^{n}}{\Delta t} + K_{\sigma} \frac{S_{h}^{n+1/2}}{2} &= 0, \\
  \mathcal{M} \frac{S_{h}^{n+3/2} - S_{h}^{n+1/2}}{\Delta t} + K_{v} \vec{V}_{h}^{n+1} &= 0
\end{align*}
\]

- \( \Delta t \) is the time step;
- \( \vec{V}_{h}^{n} \) is the approximation of \( \vec{V}_{h} \) at time \( n \);
- \( S_{h}^{n+1/2} \) is the approximation of \( S_{h} \) at time \( n + 1/2 \).

This scheme is stable and conservative.

S. Delcourte, L. Fezouli and N. Glinsky-Olivier: A High-order discontinuous Galerkin method for the seismic wave propagation
Numerical results

- \([0, 20] \times [0, 20] \times [0, 20]\) domain;
- source at \((10, 10, 10)\), \(f = 3Hz\)
- receivers at \((y = 10, z = 10)\), \(x = i, i = 1, ..., 20\)
- \(v_p = 6ms^{-1}\)
- \(v_s = 0ms^{-1}\)
- \(\rho = 1kgm^{-3}\)
- 756047 \(P3\) elements
**Numerical results**

**Fig. 1:** Result of the propagation of $v_x$ at $T=0.6s$

**Fig. 2:** Result of the propagation of $v_x$ at $T=0.9s$
Numerical results

Fig.: Result of the propagation of $v_x$ at $T=1.3s$

Fig.: Result of the propagation of $v_x$ at $T=1.8s$
Fig.: Result of the propagation of $v_x$ at $T=2.5s$

Fig.: Result of the propagation of $v_x$ at $T=3s$
Numerical results

\textbf{Fig.}: Result for the receiver at 2m from the source

\textbf{Fig.}: Result for the receiver at 8m from the source
Perspectives

- Validation on more complex cases
- Implementation of C-PML
- Local time stepping