Efficient high-order finite elements for Helmholtz equation and time-harmonic elastodynamics on hybrid meshes

M. Duruflé

IMB, Bacchus

13th December 2010

Bibliography

- S. Fauqueux, mixed spectral elements for wave and elastic equations (hexahedra)
- S. Pernet, Discontinuous Galerkin methods for Maxwell's equations (hexahedra)
- G.E. Karniadakis, S. Sherwin, T. Warburton, continuous and discontinuous finite elements on tetrahedra/prisms/pyramids by considering "degenerated" cube
- Bedrosian, Early work on pyramids, nodal basis functions for order 1 and 2
- Nigam, Philips, Recent work on finite element spaces for pyramids, infinite pyramid is the reference element

Motivation of Morgane Bergot's thesis

- Automatic generation of high-quality hexahedral meshes is difficult
- "Solution of split tetrahedra" is not interesting
- Some mesh tools are able to produce meshes with a high ratio of hexahedra and some remaining pyramids/tets/prisms.
- Pyramids elements not as well known as other elements.

Model equation

$$-\rho\,\omega^2\,u - \mathrm{Div}(\mu\,\nabla u) = f \quad \in \Omega$$

Model equation

$$-\rho\,\omega^2\,u \;-\; \mathrm{Div}(\mu\,\nabla u) \;=\; f \quad \in \Omega$$

Use of finite element method leads to the following linear system :

$$(-\omega^2 D_h \;+\; K_h)\, U_h \;=\; F_h$$

Model equation

$$-\rho\,\omega^2\,u - \mathrm{Div}(\mu\,\nabla u) = f \quad \in \Omega$$

Use of finite element method leads to the following linear system :

$$(-\omega^2 D_h + K_h)\,U_h = F_h$$

Our aim is to develop an efficient iterative solver for an high order of approximation r. Therefore, we need a fast matrix-vector product $(-\omega^2 D_h + K_h)\,U_h$

Model equation

$$-\rho\,\omega^2\,u - \mathrm{Div}(\mu\,\nabla u) = f \quad \in \Omega$$

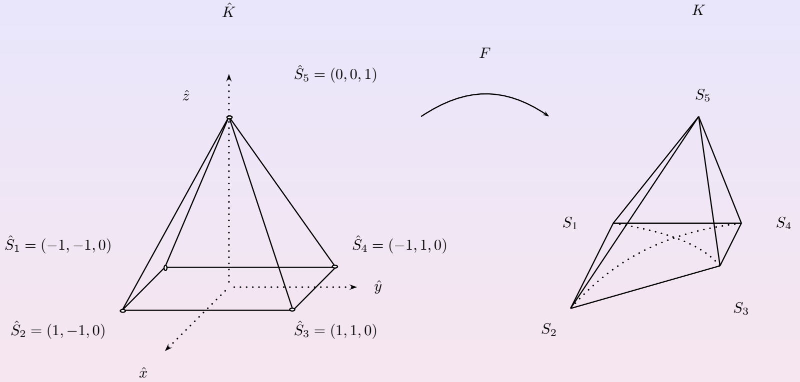
Use of finite element method leads to the following linear system :

$$(-\omega^2 D_h + K_h)\,U_h = F_h$$

Our aim is to develop an efficient iterative solver for an high order of approximation r. Therefore, we need a fast matrix-vector product $(-\omega^2 D_h + K_h)\,U_h$

u scalar $\Rightarrow$ Helmholtz equation
u vectorial $\Rightarrow$ time-harmonic elastodynamics

Finite element on pyramids



Simplest expression of F_i (Bedrosian) :

$$F_i(\hat{x},\hat{y},\hat{z}) \;=\; A + B\hat{x} + C\hat{y} + D\hat{z} + \frac{\hat{x}\,\hat{y}}{4(1-\hat{z})}\,(S_1 + S_3 - S_2 - S_4)$$

Finite element on pyramids

- Use of rational fractions to define F_i
  - Early work of Bedrosian with explicit first and second order basis functions
  - Work of Sherwin, Karniadakis, Warburton : h-p Basis functions obtained by considering a degenerated cube (coincidence with Bedrosian functions for r = 1)
  - Recent work of Nigam, Phillips with a reference infinite pyramid (same basis functions as Bedrosian for r = 1)

- Use of piecewise polynomial to define F_i (polynomial on each sub-tetrahedron)
  - Work of Wieners, with first and second order basis functions
  - Work of Knabner and Summ, with an analysis of this transformation
  - Work of Bluck and Walker, with a proposition of high order basis functions

Condition of optimality

We define the finite element space with real element K_i :

$$V_h = \{u \in H^1(\Omega) \text{ such that } u|_{K_i} \in V_F^r\}$$

V_F^r : finite element space for the real element

We define the finite element space with reference element $\hat{K}$:

$$V_h = \{u \in H^1(\Omega) \text{ such that } u|_{K_i} \circ F_i \in \hat{V}^r\}$$

$\hat{V}^r$: finite element space for the reference element

Condition of optimality :

$$V_F^r \supset P_r$$

For hexahedra, we can prove :

$$V_F^r \supset P_r \Leftrightarrow \hat{V}^r \supset Q_r$$

Optimal finite element space

Same approach than for hexahedra : We consider a monomial of P_r :

$$x^m, \qquad m \leq r$$

$$(a + b\hat{x} + c\hat{y} + d\hat{z} + \alpha(\frac{\hat{x}\hat{y}}{1-\hat{z}}))^m$$

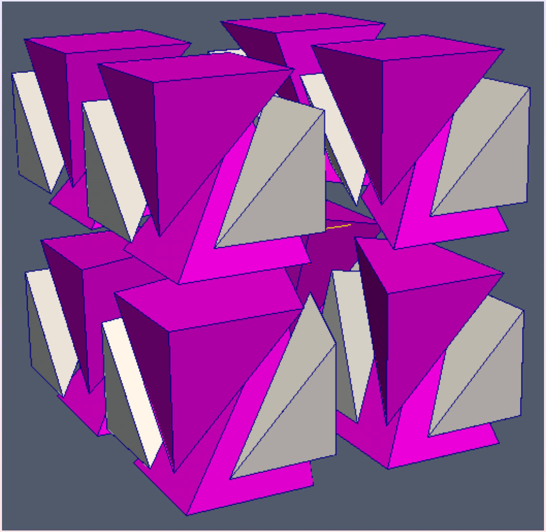
$$\sum_k C_m^k (a + b\hat{x} + c\hat{y})^k (d\hat{z})^k \alpha^{m-k} (\frac{\hat{x}\hat{y}}{1-\hat{z}})^{m-k}$$

After some calculations, you can show that the optimal finite element space is

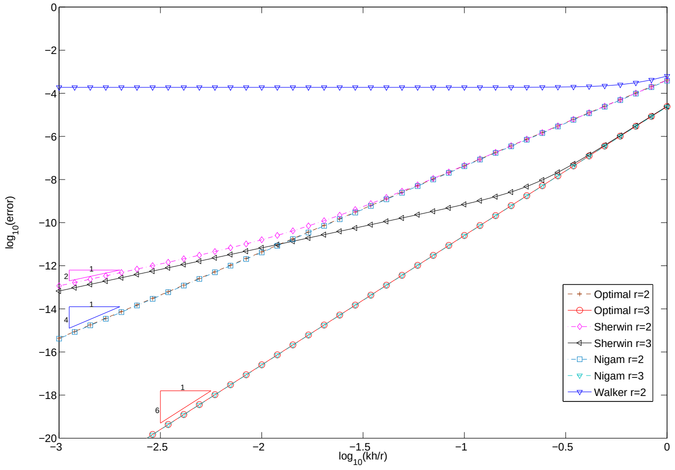
$$\hat{V}^r = P_r \oplus \sum_{k=0}^{r-1} (\frac{\hat{x}\hat{y}}{1-\hat{z}})^{r-k} P_k(\hat{x}, \hat{y})$$

Numerical comparison between different methods

We perform a dispersion analysis on the following hybrid mesh :



Numerical comparison between different methods



Numerical comparison between different methods

- We obtained same finite element space as Demkowicz/Zaglmayr
- We obtained a smaller finite element space than Nigam/Phillips
- We proposed modifications of basis functions of Sherwin/Karniadakis/Warburton so that they span the optimal finite element space
- Alternative approach using piecewise polynomial (by splitting pyramid in two or four tets) is not consistent for non-affine pyramids and order greater than 2

Numerical comparison between different methods

- We obtained same finite element space as Demkowicz/Zaglmayr
- We obtained a smaller finite element space than Nigam/Phillips
- We proposed modifications of basis functions of Sherwin/Karniadakis/Warburton so that they span the optimal finite element space
- Alternative approach using piecewise polynomial (by splitting pyramid in two or four tets) is not consistent for non-affine pyramids and order greater than 2

- Optimal finite element space constructed in Morgane Bergot's thesis for edge elements , different from Nigam/Phillips and Demkowicz/Zaglmayr

Nodal Basis functions

Orthogonal basis of pyramidal finite element space

$$\psi_{i,j,k} \;=\; P_i^{0,0}\left(\frac{\hat{x}}{1-\hat{z}}\right) P_j^{0,0}\left(\frac{\hat{y}}{1-\hat{z}}\right) P_k^{2\max(i,j)+2,0}(2\hat{z}-1)(1-z)^{\max(i,j)}$$

where $P_i^{\alpha,\beta}$ are Jacobi polynomials orthogonal with respect to $(1-x)^{\alpha}(1+x)^{\beta}$

Nodal Basis functions

Orthogonal basis of pyramidal finite element space

$$\psi_{i,j,k} = P_i^{0,0}\left(\frac{\hat{x}}{1-\hat{z}}\right) P_j^{0,0}\left(\frac{\hat{y}}{1-\hat{z}}\right) P_k^{2\max(i,j)+2,0}(2\hat{z}-1)(1-z)^{\max(i,j)}$$

where $P_i^{\alpha,\beta}$ are Jacobi polynomials orthogonal with respect to $(1-x)^\alpha (1+x)^\beta$

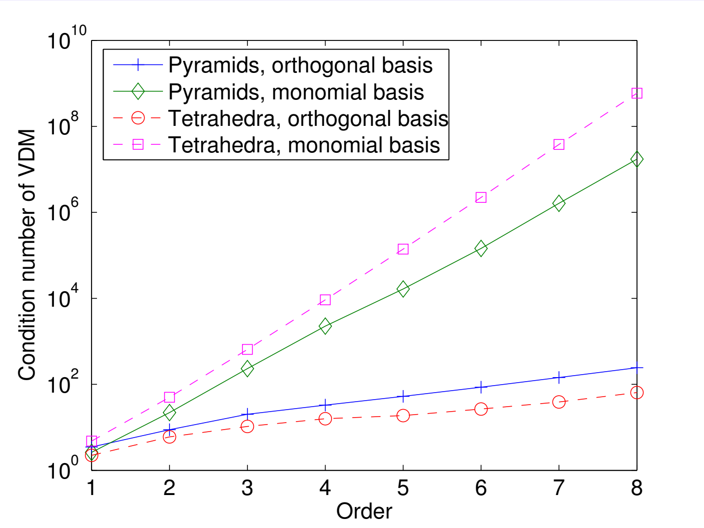
M_i : interpolation points on the reference pyramid
Vandermonde matrix :

$$VDM_{i,j} = \psi_i(M_j)$$

Nodal basis functions :

$$\varphi_i = \sum_j (VDM^{-1})_{i,j} \, \psi_j$$

Nodal Basis functions



Condition number of Vandermonde matrix

Hierarchical Basis functions

- Same basis functions as Sherwin, Karniadakis, Warburton for hexahedra, prisms, tetrahedra, but different ones for pyramids

Hierarchical Basis functions

- Same basis functions as Sherwin, Karniadakis, Warburton for hexahedra, prisms, tetrahedra, but different ones for pyramids
- Vertex :

$$N_1 = \frac{(1-\hat{x}-\hat{z})(1-\hat{y}-\hat{z})}{4\,(1-\hat{z})}$$

- Apex : $\hat{z}$
- Horizontal edge :

$$N_1 \, \frac{(1+\hat{x}-\hat{z})}{2} \, (1-\hat{z})^{i-1} \, P_{i-1}^{1,1}\left(\frac{\hat{x}}{1-\hat{z}}\right)$$

- Vertical edge :

$$N_1 \, \hat{z} \, P_{i-1}^{1,1}\left(\hat{z} + \frac{\hat{x}+\hat{y}}{2}\right)$$

- Triangular face :

$$N_1 \, \frac{(1+\hat{x}-\hat{z})}{2} \, \hat{z} \, (1-\hat{z})^{i-1} \, P_{i-1}^{1,1}\left(\frac{\hat{x}}{1-\hat{z}}\right) \, P_{j-1}^{2i+1,1}(2\hat{z}-1)$$

Hierarchical Basis functions

(Differences with Sherwin, Karniadakis, Warburton denoted in red)

- Base :

$$N_1 \, N_3 \, (1-\hat{z})^{\max(i,j)-1} \, P_{i-1}^{1,1}\left(\frac{\hat{x}}{1-\hat{z}}\right) P_{j-1}^{1,1}\left(\frac{\hat{y}}{1-\hat{z}}\right)$$

- Interior :

$$N_1 \, N_3 \, \hat{z} \, (1-\hat{z})^{\max(i,j)-1} \, P_{i-1}^{1,1}\left(\frac{\hat{x}}{1-\hat{z}}\right) P_{j-1}^{1,1}\left(\frac{\hat{y}}{1-\hat{z}}\right) P_{k-1}^{2\max(i,j)+2,1}(2\hat{z}-1$$

Fast matrix-vector product

Semi-tensorization of basis functions ⇒ fast matrix-vector product

$$\varphi_j = \varphi_{j_1}(\hat{x})\,\varphi_{j_2}^{j_1}(\hat{y})\varphi_{j_3}^{j_1,j_2}(\hat{z})$$

Fast matrix-vector product

$$(D_h)_{i,j} \;=\; \int_{\hat{K}} \rho J_i \, \hat{\varphi}_i \, \hat{\varphi}_j \, d\hat{x}$$

Use of quadrature formulas (ω_m, ξ_m) on the reference element

Fast matrix-vector product

$$(D_h)_{i,j} \;=\; \int_{\hat{K}} \rho J_i \, \hat{\varphi}_i \, \hat{\varphi}_j \, d\hat{x}$$

Use of quadrature formulas (ω_m, ξ_m) on the reference element

$$(D_h)_{i,j} \;=\; \sum_m \omega_m \, \rho J_i \, \hat{\varphi}_i(\xi_m) \, \hat{\varphi}_j(\xi_m)$$

Matrix-vector product $D_h U$ can be split into three steps :

$$v_m \;=\; \sum_j \hat{\varphi}_j(\xi_m) u_j$$

$$w_m \;=\; \omega_m \rho J_i(\xi_m) v_m$$

$$y_i \;=\; \sum_m \hat{\varphi}_i(\xi_m) w_m$$

Fast matrix-vector product

$$(D_h)_{i,j} \;=\; \int_{\hat{K}} \rho J_i \, \hat{\varphi}_i \, \hat{\varphi}_j \, d\hat{x}$$

Use of quadrature formulas (ω_m, ξ_m) on the reference element

$$(D_h)_{i,j} \;=\; \sum_m \omega_m \, \rho J_i \, \hat{\varphi}_i(\xi_m) \, \hat{\varphi}_j(\xi_m)$$

Underlying factorization

$$\hat{C}_{i,j} \;=\; \hat{\varphi}_i(\xi_j)$$

$$(A_h)_m = \omega_m \rho J_i(\xi_m)$$

$$D_h \;=\; \hat{C} A_h \hat{C}^*$$

$\Rightarrow$ only storage of $\omega_m \rho J_i(\xi_m)$

Fast matrix-vector product

$$(D_h)_{i,j} \; = \; \int_{\hat{K}} \rho J_i \, \hat{\varphi}_i \, \hat{\varphi}_j \, d\hat{x}$$

Use of quadrature formulas (ω_m, ξ_m) on the reference element

Product $Y = \hat{C} U$ is split into three steps :

$$v_{j_1, j_2, i_3} \; = \; \sum_{j_3} \hat{\varphi}_{j_3}^{j_1, j_2}(\xi_{i_3}) u_{j_1, j_2, j_3}$$

$$w_{j_1, i_2, i_3} \; = \; \sum_{j_2} \hat{\varphi}_{j_2}^{j_1}(\xi_{i_2}) v_{j_1, j_2, i_3}$$

$$y_{i_1, i_2, i_3} \; = \; \sum_{j_1} \hat{\varphi}_{j_1}(\xi_{i_1}) w_{j_1, i_2, i_3}$$

Stiffness terms

$$(K_h)_{i,j} \; = \; \int_{\hat{K}} J_i \, DF_i^{-1} \mu DF_i^{*-1}(\xi_m) \, \hat{\nabla} \hat{\varphi}_j \cdot \hat{\nabla} \hat{\varphi}_i \, d\hat{x}$$

Stiffness terms

$$(K_h)_{i,j} = \int_{\hat{K}} J_i \, DF_i^{-1} \mu DF_i^{*-1}(\xi_m) \, \hat{\nabla}\hat{\varphi}_j \cdot \hat{\nabla}\hat{\varphi}_i \, d\hat{x}$$

Matrix-vector product $K_h U$ can be split into three steps

$$v_m = \sum_j \hat{\nabla}\hat{\varphi}_j(\xi_m) u_j$$

$$w_m = \omega_m J_i \, DF_i^{-1} \mu DF_i^{*-1} v_m$$

$$y_i = \sum_q \hat{\nabla}\hat{\varphi}_i(\xi_m) w_m$$

Stiffness terms

$$(K_h)_{i,j} = \int_{\hat{K}} J_i \, DF_i^{-1} \mu DF_i^{*-1}(\xi_m) \, \hat{\nabla}\hat{\varphi}_j \cdot \hat{\nabla}\hat{\varphi}_i \, d\hat{x}$$

Underlying factorization

$$\hat{S}_{i,j} = \hat{\nabla}\hat{\varphi}_i(\xi_j)$$

$$(B_h)_m = \omega_m J_i \, DF_i^{-1} \mu DF_i^{*-1}$$

$$K_h = \hat{S} B_h \hat{S}^*$$

$\Rightarrow$ only storage of $J_i \, DF_i^{-1} \mu DF_i^{*-1}$ for Helmholtz equation, and only J_i and DF_i^{-1} for elastodynamics

Stiffness terms

By using the matrices

$$\hat{C}_{i,j} = \hat{\varphi}_i(\xi_j)$$

$$\hat{S}_{i,j} = \hat{\nabla}\hat{\varphi}_i(\xi_j)$$

$$\hat{R}_{i,j} = \hat{\nabla}\hat{\psi}_i(\xi_j)$$

where ψ are basis functions associated with quadrature points, we have $\hat{S} = \hat{R}\hat{C}$

final matrix : $\hat{C}(-\omega^2 A_h + \hat{R} B_h \hat{R}^*)\hat{C}^*$

Comparison Nodal/Hp

Computational time for 100 iterations of COCG on a mesh containing one million dofs. Pyramids, Helmholtz equation

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	327s	499s	1021s	1918s	4345s
Hierarchic	285.6s	183s	183.7s	194s	238s
Stored matrix	26s	55s	113s	234s	359s
	0.27 Go	0.78 Go	1.68 Go	3.09 Go	5.13 Go

Hexahedra, Helmholtz equation

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	77s	49s	45s	42s	46s
Hierarchic	99s	64s	62s	77s	68s
Stored matrix	22s	45s	79s	120s	171s
	0.27 Go	0.64 Go	1.170 Go	1.85 Go	2.72 Go

Comparison Nodal/Hp

Computational time for 100 iterations of COCG on a mesh containing one million dofs. Pyramids, Helmholtz equation

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	327s	499s	1021s	1918s	4345s
Hierarchic	285.6s	183s	183.7s	194s	238s
Stored matrix	26s	55s	113s	234s	359s
	0.27 Go	0.78 Go	1.68 Go	3.09 Go	5.13 Go

Hexahedra, Helmholtz equation

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	77s	49s	45s	42s	46s
Hierarchic	99s	64s	62s	77s	68s
Stored matrix	22s	45s	79s	120s	171s
	0.27 Go	0.64 Go	1.170 Go	1.85 Go	2.72 Go

Comparison Nodal/Hp, Elastodynamics

Pyramids, Elastodynamics

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	675s	630s	999s	1 553s	3 418s
Hierarchic	723s	468s	482s	517s	670 s
Stored matrix	205s	498s	1 935s	4 163s	5 351s
	2.56 Go	7.36 Go	16.5 Go	30.3 Go	50.8 Go

Hexahedra, Elastodynamics

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	197s	120s	114s	107s	123s
Hierarchic	259s	179s	165s	178s	184s
Stored matrix	216s	410s	814s	3 029s	3 105s
	2.52 Go	5.69 Go	11.4 Go	18.3 Go	24.3 Go

Comparison Nodal/Hp, Elastodynamics

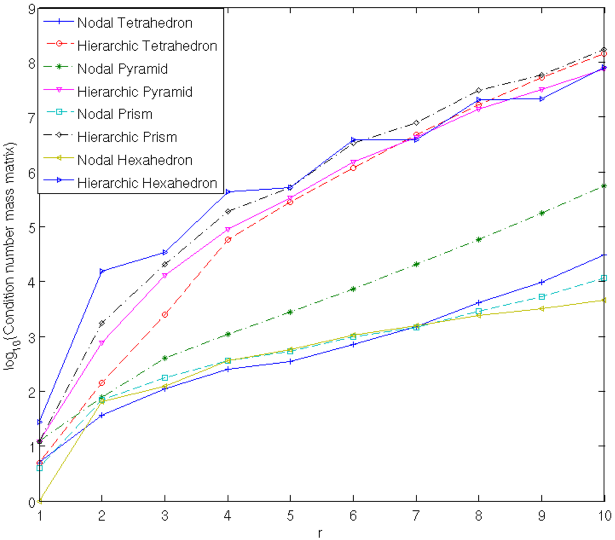
Pyramids, Elastodynamics

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	675s	630s	999s	1 553s	3 418s
Hierarchic	723s	468s	482s	517s	670 s
Stored matrix	205s	498s	1 935s	4 163s	5 351s
	2.56 Go	7.36 Go	16.5 Go	30.3 Go	50.8 Go

Hexahedra, Elastodynamics

Order	r = 2	r = 4	r = 6	r = 8	r = 10
Nodal	197s	120s	114s	107s	123s
Hierarchic	259s	179s	165s	178s	184s
Stored matrix	216s	410s	814s	3 029s	3 105s
	2.52 Go	5.69 Go	11.4 Go	18.3 Go	24.3 Go

Comparison Nodal/Hp, Condition number



Preconditioning techniques

- p-multigrid iteration on damped equation

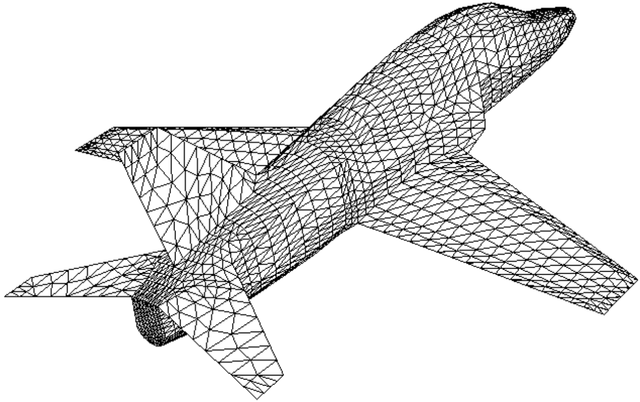
$$-\omega^2(\alpha + i\beta)u - \mathrm{Div}(\mu\nabla u) = 0$$

  - Jacobi smoother for hexahedral meshes
  - Gauss-Seidel smoother for hybrid meshes
- subdomain-preconditioning (additive Schwarz-like) :

$$M = \sum P_i A_i^{-1} P_i$$

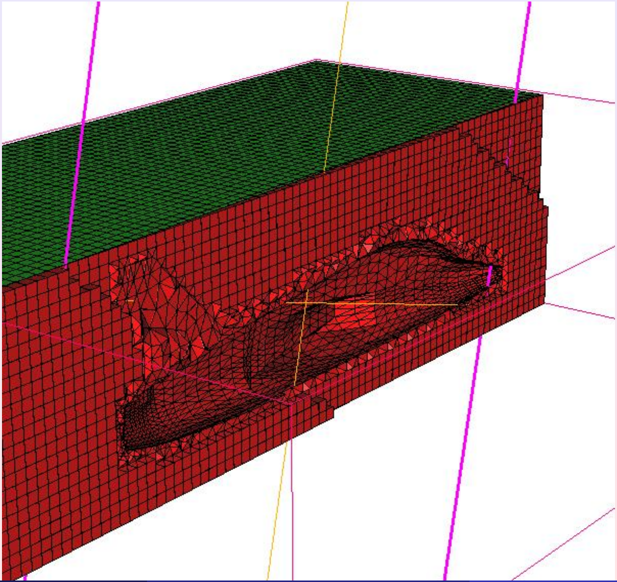
where A_i is the finite element matrix on subdomain Ω_i with absorbing boundary conditions
one processor = one domain

Scattering of an airplane

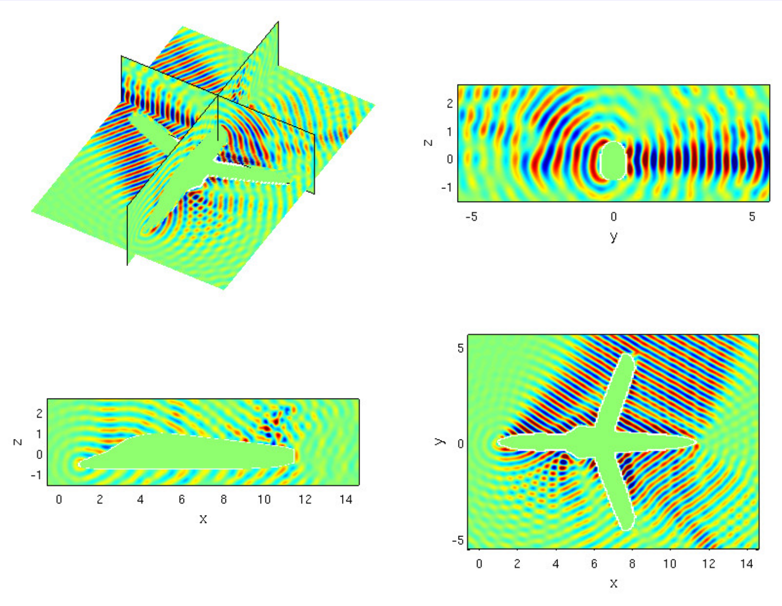


Scattering of an airplane

Hybrid mesh used :



Scattering of an airplane



Statistics for airplane

Without preconditioning :

Mesh	Hybrid	Split tetrahedra	Tetrahedra
Dofs	6.08 millions	13.2 millions	5.39 millions
L^2 error	1.05 %	0.89 %	1.14 %
Iterations	13 113	94 500	24 325
Time	24 253s	981 139s	80 274s

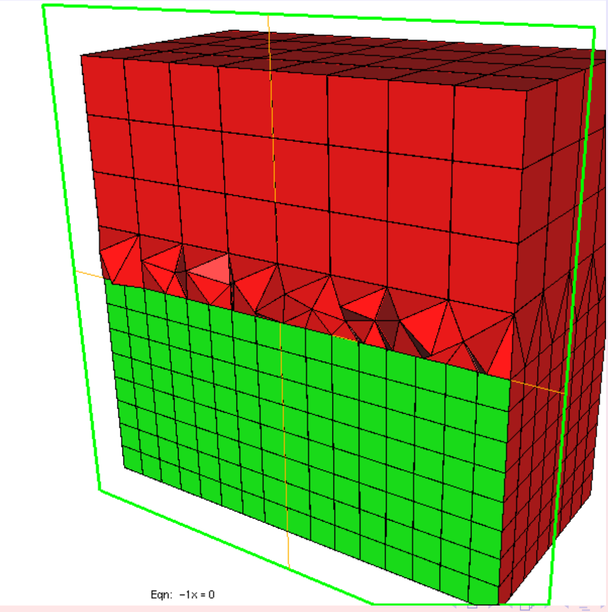
Multigrid preconditioning :

Iterations	193	781	268
Time	2 870s	68 354s	9 177s

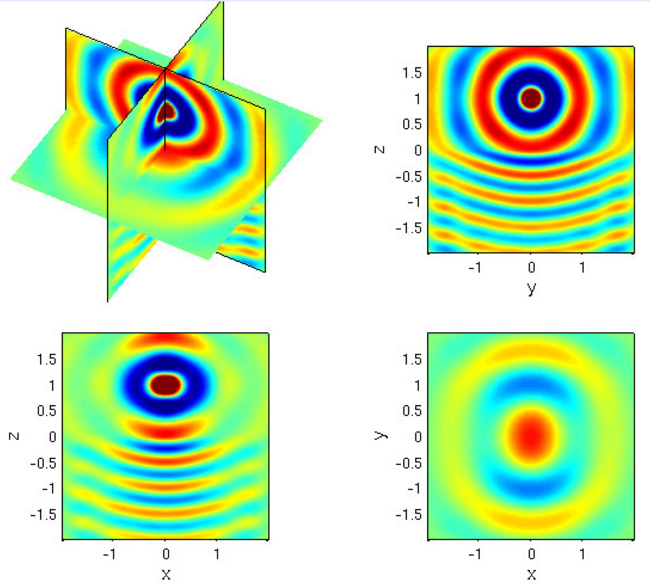
Subdomain preconditioning (128 domains) :

Iterations	545	579	481
Time	26 121s	39 500s	10 684s

Two-layer problem



Two-layer problem



Statistics of two-layer problem

Without preconditioning :

	Hexahedra	Hybrid
Dofs	274 625	189 669
Iterations	2808	10 530
Time	2 285s	7 788s

Subdomain preconditioning (32 subdomains) :

Iterations	263	505
Time	3 838s	19 644s

Two-grid preconditioning :

Iterations	59	117
Time	307s	346s