

# **H-MATRIX SOLVER APPLIED TO COUPLED FEM-BEM AEROACOUSTICS SIMULATIONS**

**AIRBUS**

Guillaume SYLVAND - Airbus Central R&T

JOSO - March 12th, 2019.



- Airbus Central R&T

- Airbus Central R&T
  - Departement "Virtual Product Engineering"



- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS
  - and numerous PhD:

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS
  - and numerous PhD:
    - Fabien Casenave (2013) FEM/BEM coupling

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS
  - and numerous PhD:
    - Fabien Casenave (2013) FEM/BEM coupling
    - Benoit Lizé (2014) H-matrix solver

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS
  - and numerous PhD:
    - Fabien Casenave (2013) FEM/BEM coupling
    - Benoit Lizé (2014) H-matrix solver
    - Antoine Bensalah (2018) extension to FEM/BEM formulation

- Airbus Central R&T
  - Departement "Virtual Product Engineering"
- Joint Work with
  - Applied maths team
  - IMACS
  - and numerous PhD:
    - Fabien Casenave (2013) FEM/BEM coupling
    - Benoit Lizé (2014) H-matrix solver
    - Antoine Bensalah (2018) extension to FEM/BEM formulation
    - Aurélien Falco (WIP) H-matrix for FEM/BEM system





- Sound propagation in a air flow

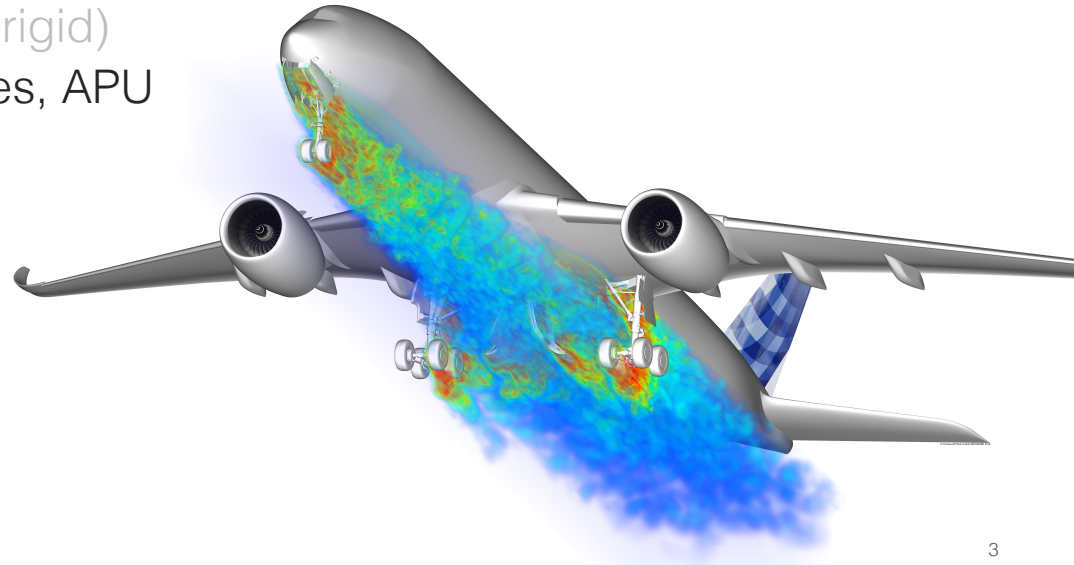
- Sound propagation in a air flow
- Context for Airbus

- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts

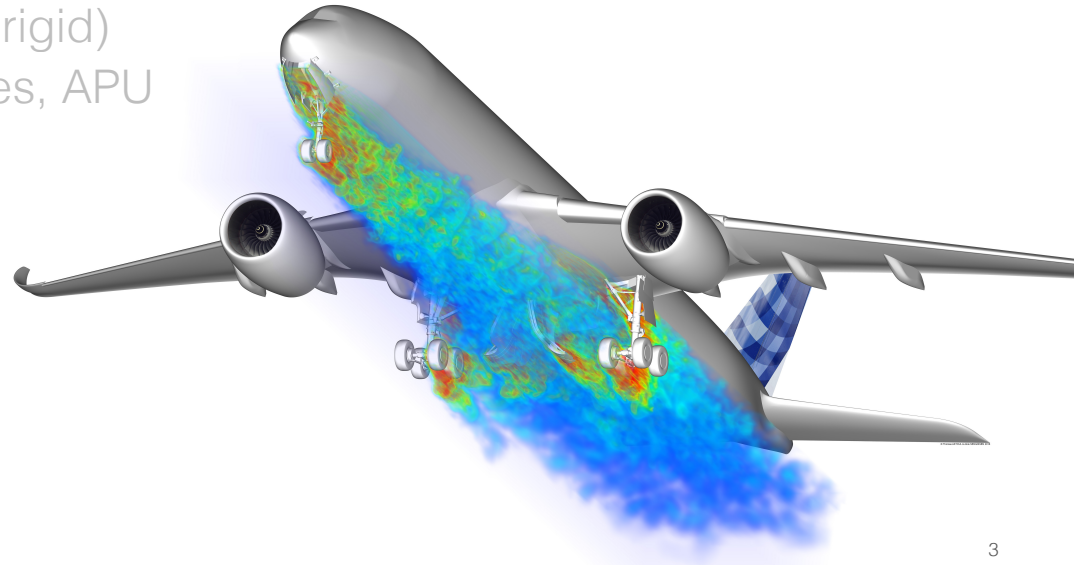
- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?

- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)

- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)
  - Sources: Landing gear, engines, APU

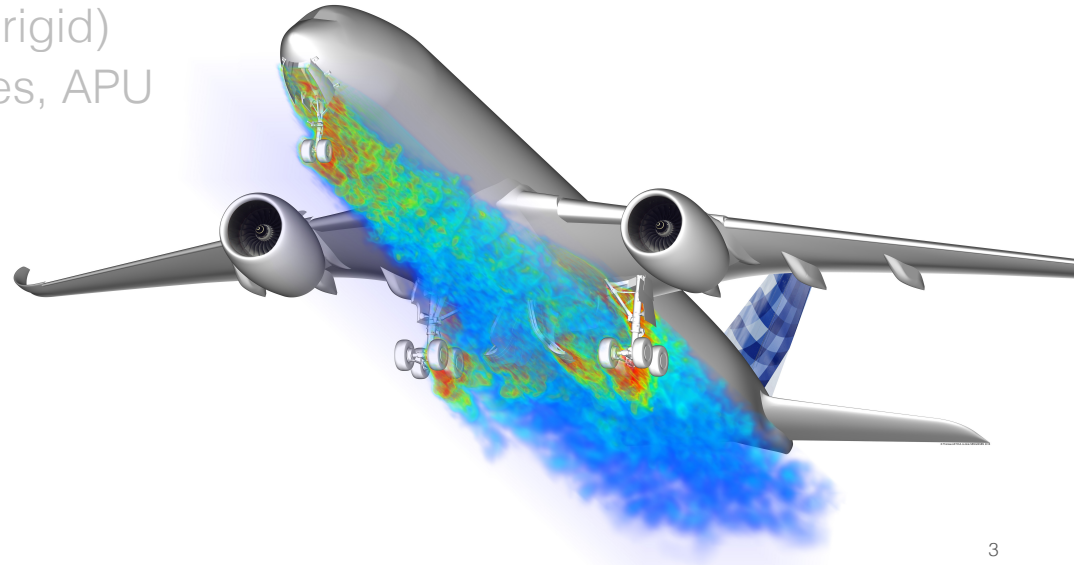


- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)
  - Sources: Landing gear, engines, APU
  - Flow: air flow, jet flow

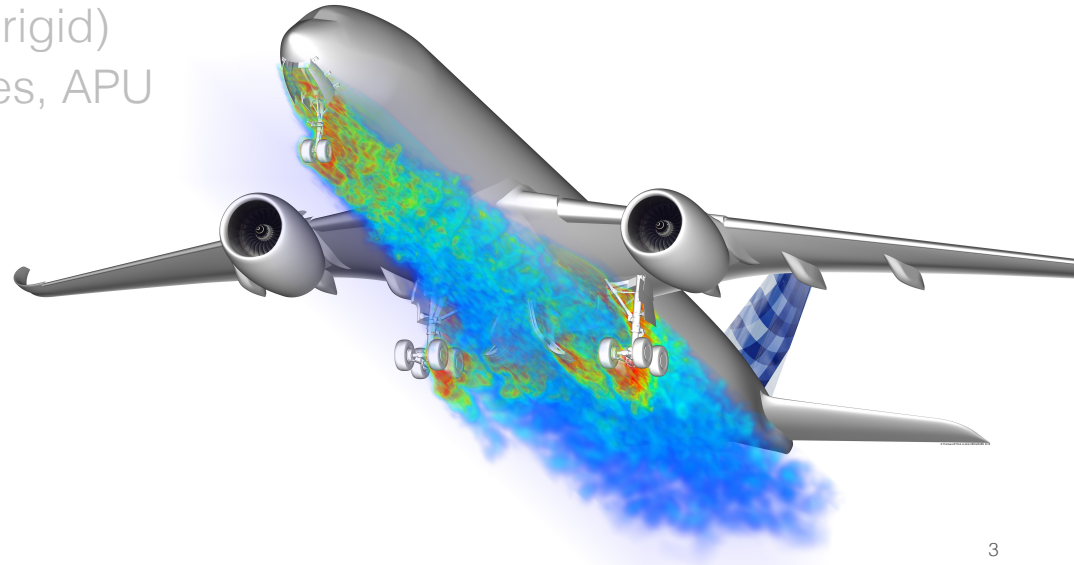




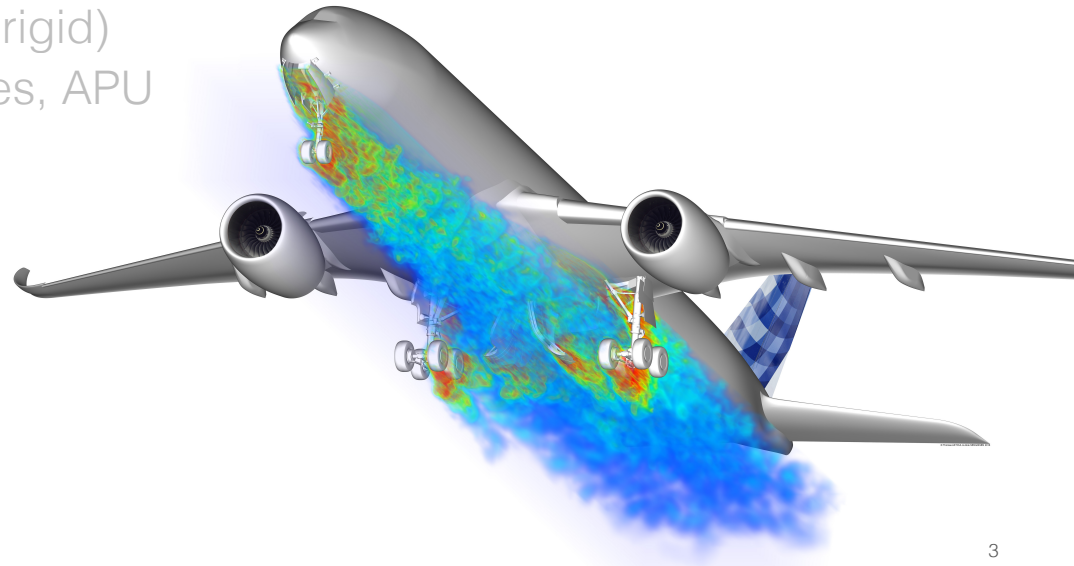
- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)
  - Sources: Landing gear, engines, APU
  - Flow: air flow, jet flow
- What not to modelize ?



- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)
  - Sources: Landing gear, engines, APU
  - Flow: air flow, jet flow
- What not to modelize ?
  - inside the engine



- Sound propagation in a air flow
- Context for Airbus
  - Numerical certification for commercial aircrafts
- What to modelize ?
  - Shapes and Materials (mostly rigid)
  - Sources: Landing gear, engines, APU
  - Flow: air flow, jet flow
- What not to modelize ?
  - inside the engine
  - retroaction sound  $\rightarrow$  flow



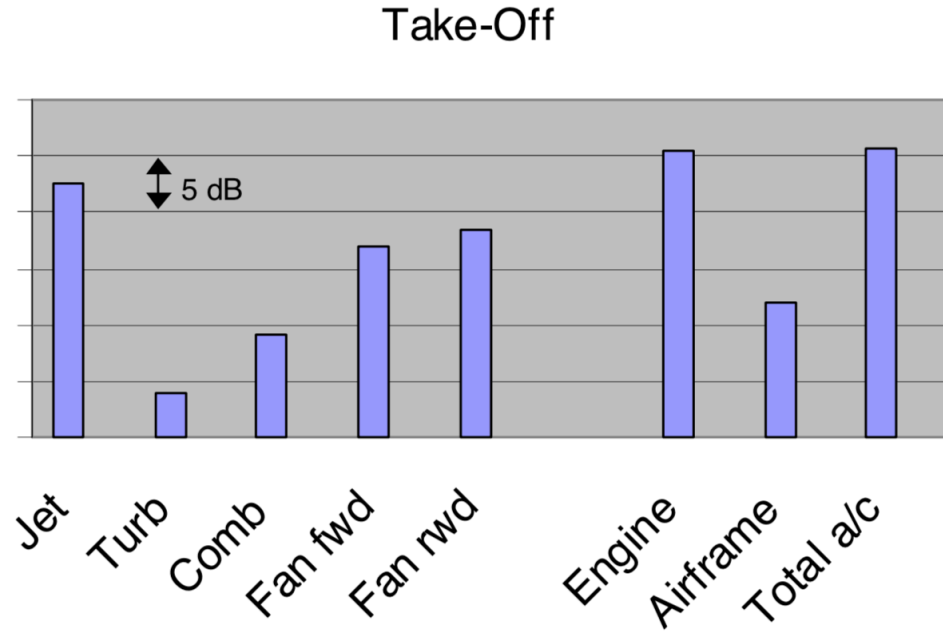


- Depends on the phase of the flight

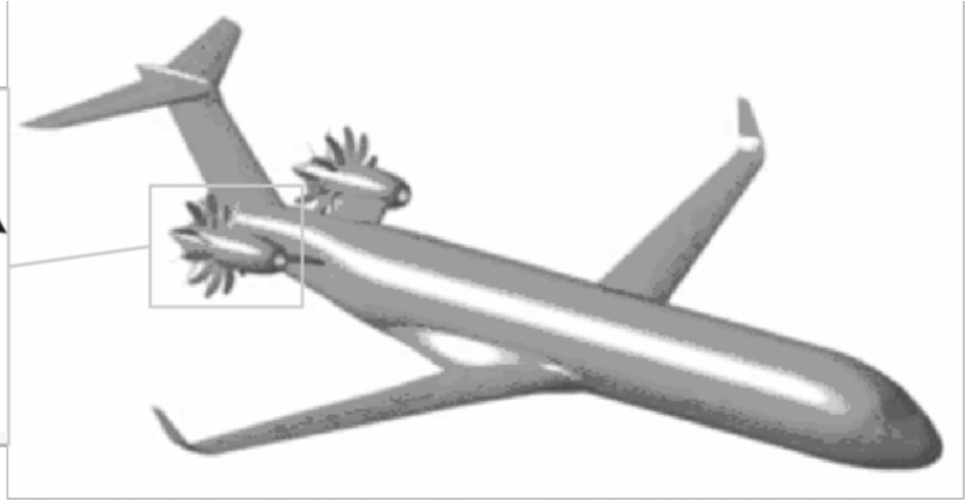
- Depends on the phase of the flight
  - Taxi: APU



- Depends on the phase of the flight
  - Taxi: APU
  - Take-off

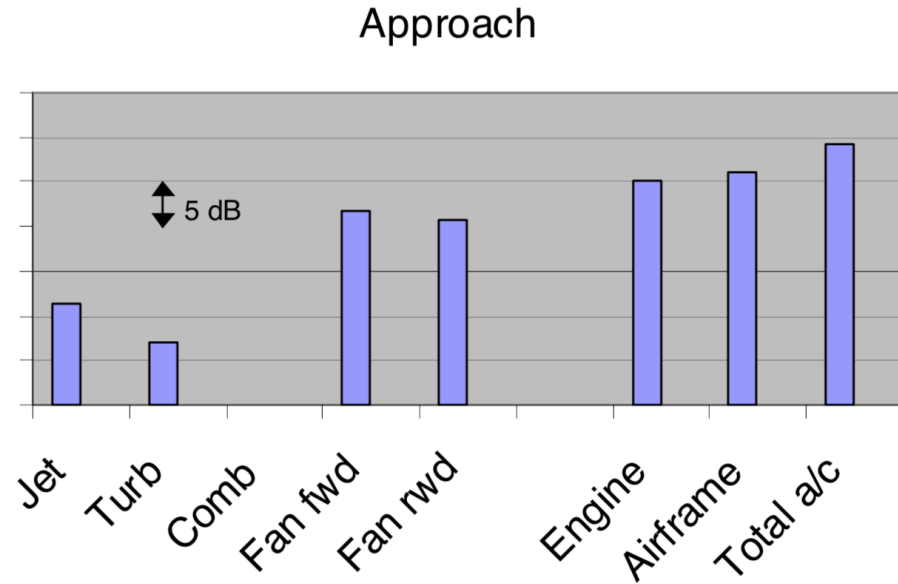


- Depends on the phase of the flight
  - Taxi: APU
  - Take-off
  - Cruise: CROR ?





- Depends on the phase of the flight
  - Taxi: APU
  - Take-off
  - Cruise: CROR ?
  - Approach/Landing





- Frequency domain, 3D to solve Helmholtz equation

$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

- Frequency domain, 3D to solve Helmholtz equation

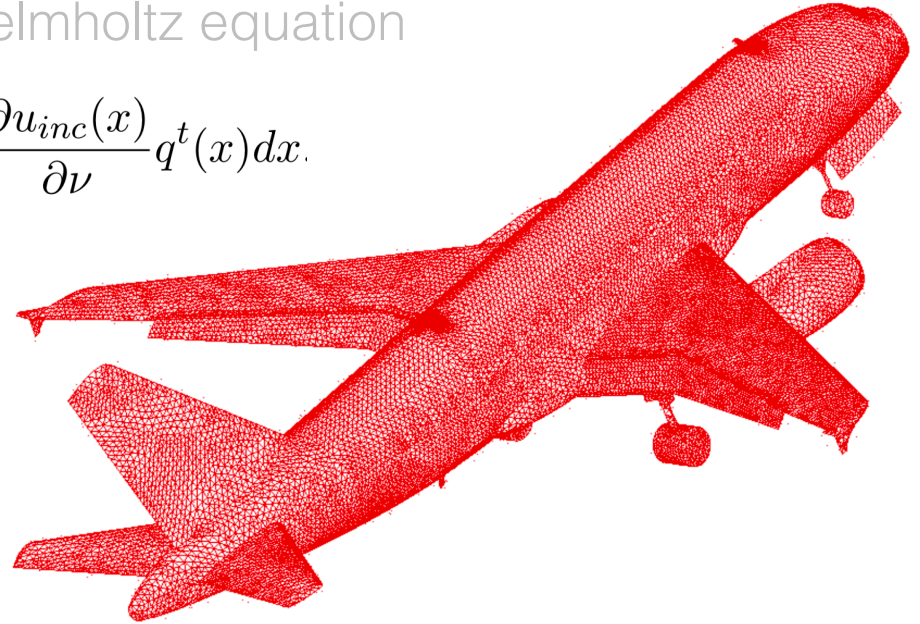
$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

- Interest:

- Frequency domain, 3D to solve Helmholtz equation

$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

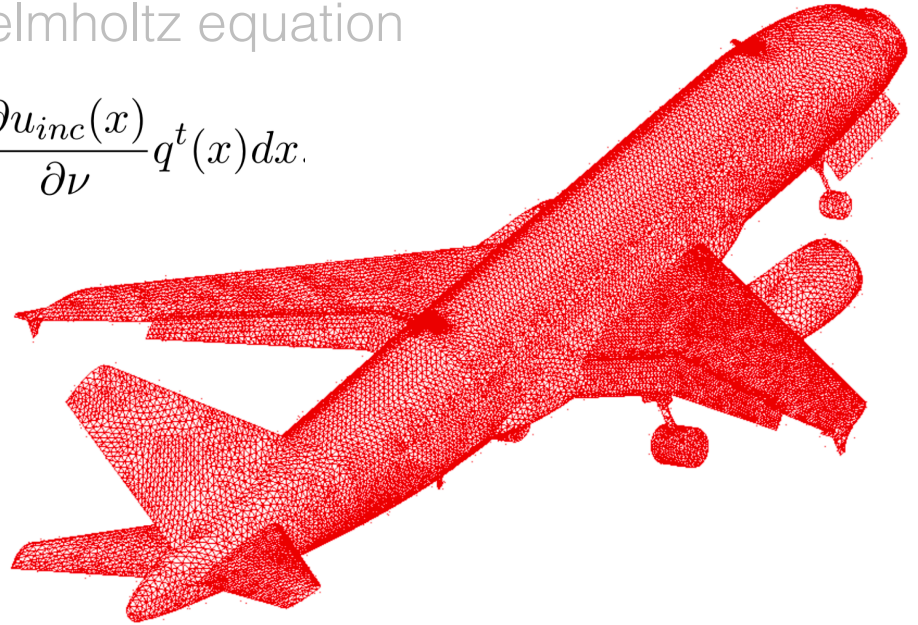
- Interest:
  - accuracy



- Frequency domain, 3D to solve Helmholtz equation

$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

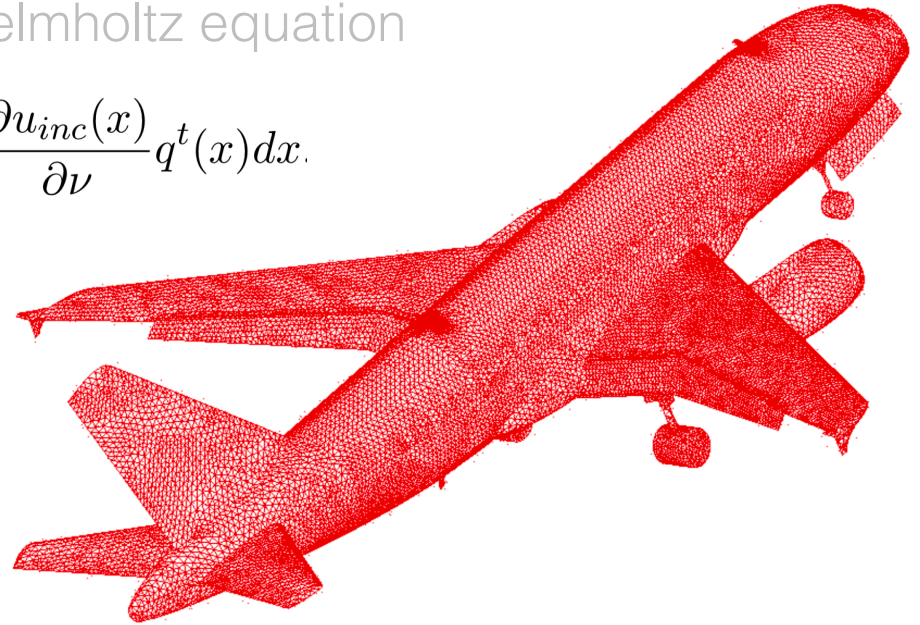
- Interest:
  - accuracy
  - smaller problem size



- Frequency domain, 3D to solve Helmholtz equation

$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

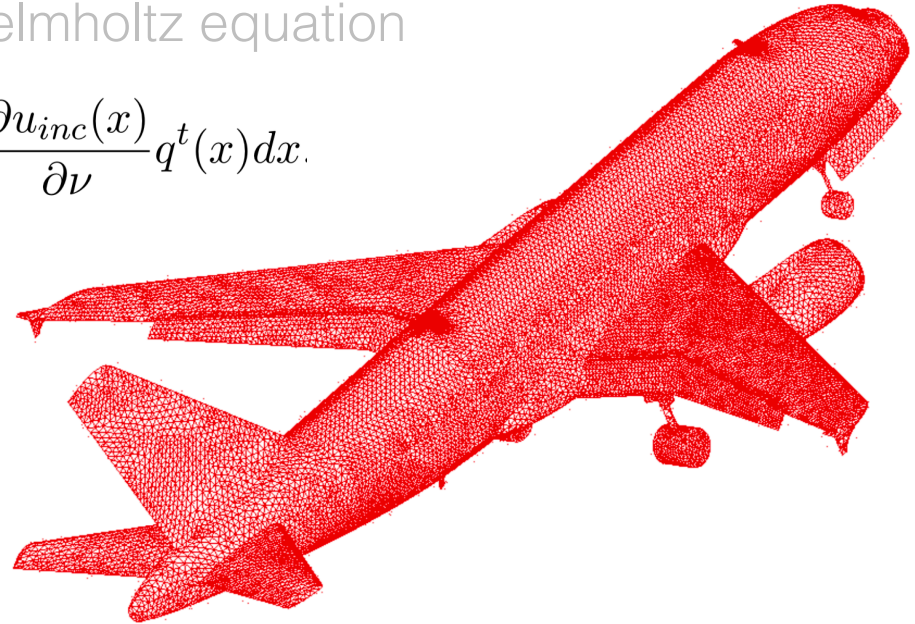
- Interest:
  - accuracy
  - smaller problem size
- Drawback:



- Frequency domain, 3D to solve Helmholtz equation

$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

- Interest:
  - accuracy
  - smaller problem size
- Drawback:
  - Homogeneous domain

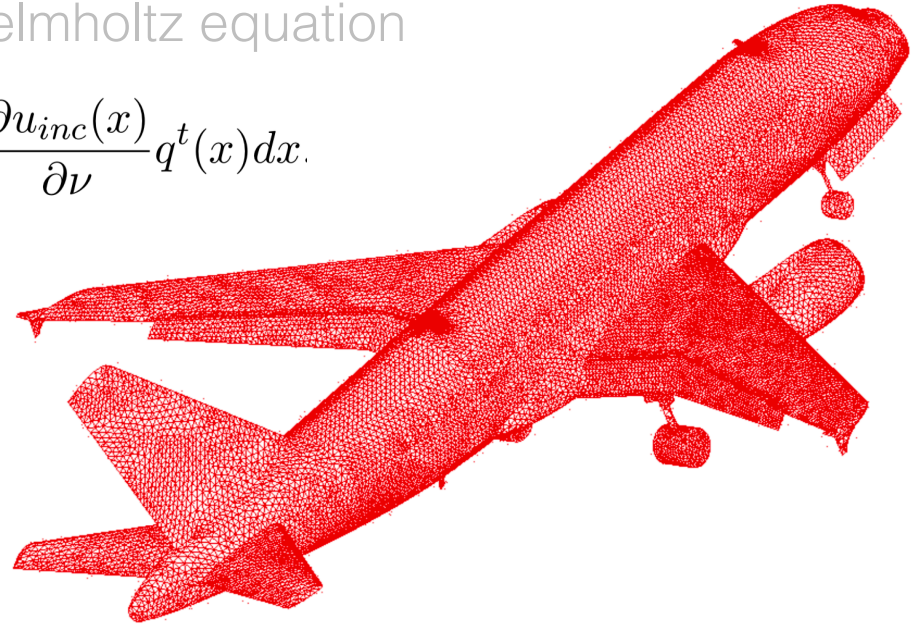




- Frequency domain, 3D to solve Helmholtz equation

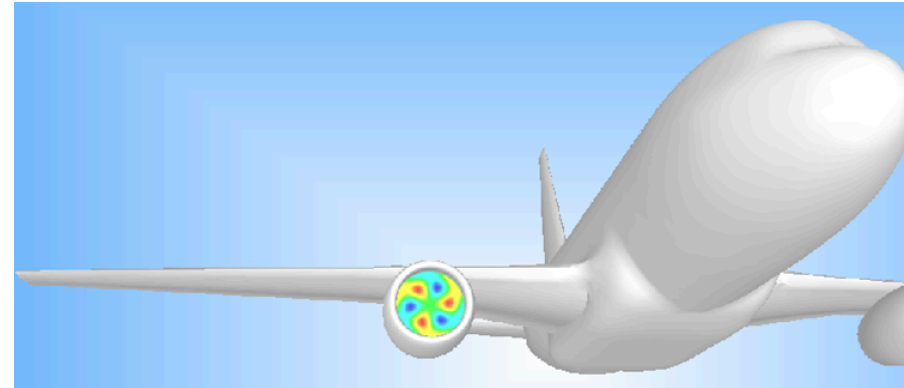
$$-\frac{1}{ik} \oint_{\Gamma \times \Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} q(y) q^t(x) dy dx = -\frac{1}{ik} \int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu} q^t(x) dx.$$

- Interest:
  - accuracy
  - smaller problem size
- Drawback:
  - Homogeneous domain
  - Dense linear system

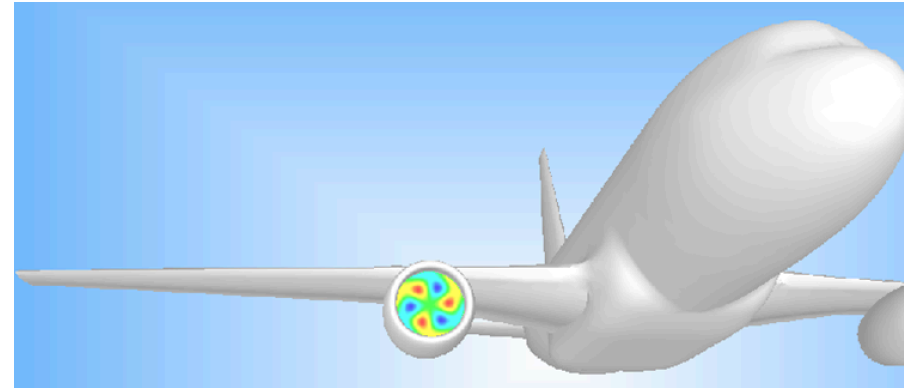




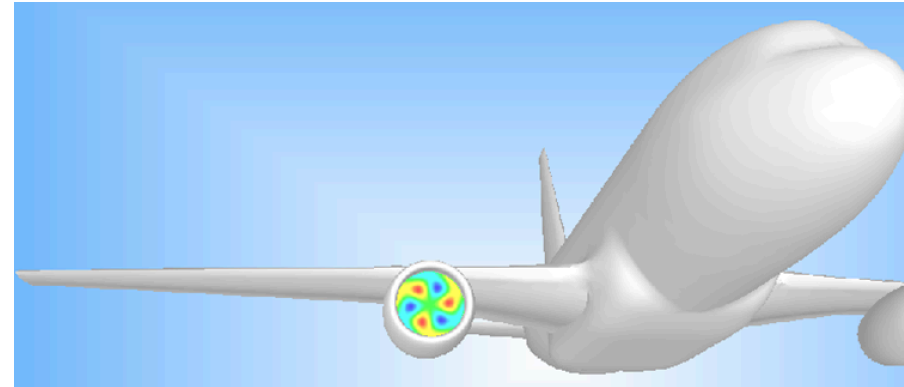
- To modelize engine sources : waveguides (cylindrical or coaxial)



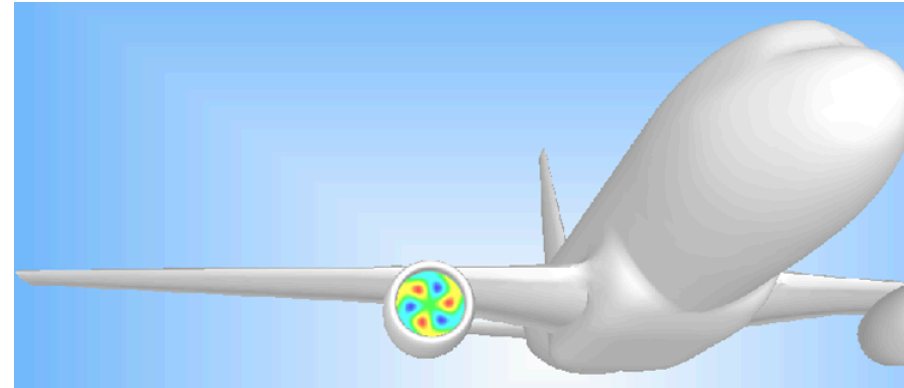
- To modelize engine sources : waveguides (cylindrical or coaxial)
  - Rough approximation...



- To modelize engine sources : waveguides (cylindrical or coaxial)
  - Rough approximation...  
... but can be used in comparison between configurations

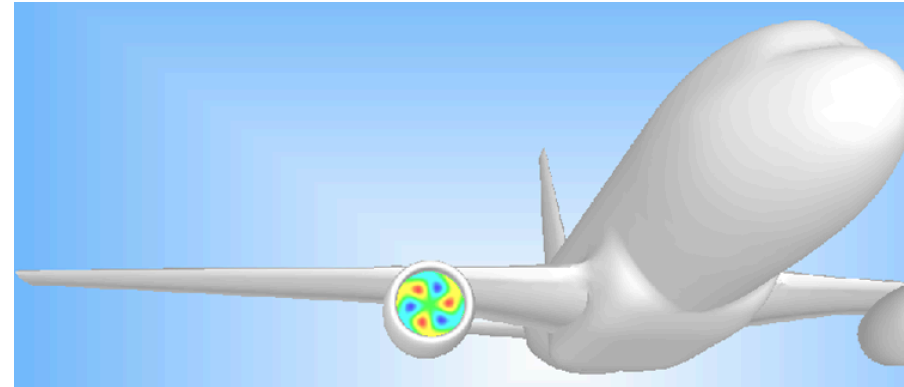


- To modelize engine sources : waveguides (cylindrical or coaxial)
  - Rough approximation...  
... but can be used in comparison between configurations
- To take into account a **uniform** air flow : Lorentz transform (or Prandtl–Glauert transformation)



- To modelize engine sources : waveguides (cylindrical or coaxial)
  - Rough approximation...  
... but can be used in comparison between configurations
- To take into account a **uniform** air flow : Lorentz transform (or Prandtl–Glauert transformation)

*Boundary element and finite element coupling for aeroacoustics simulations, by Balin et al., JCP 2015.*







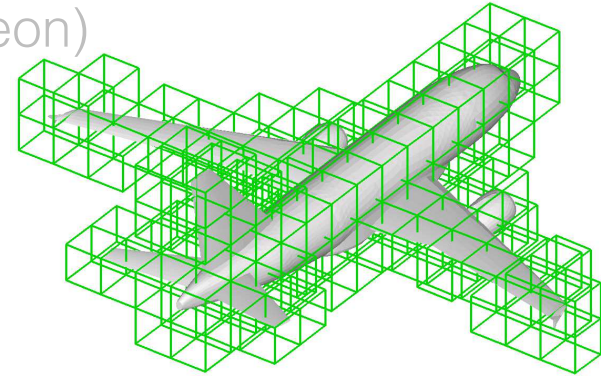
- Dense linear system

- Dense linear system
- Large number of unknowns (up to several millions)

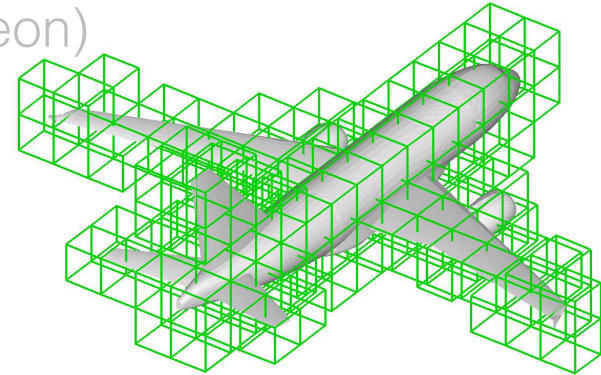
- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)

- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)
  - excellent accuracy, but expensive

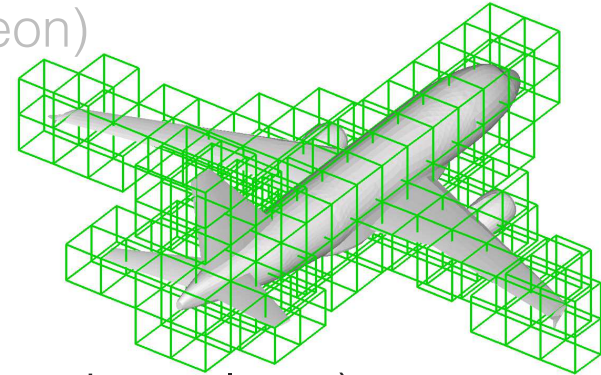
- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)
  - excellent accuracy, but expensive
- FMM Solver (Greengard & Rokhlin, 87)



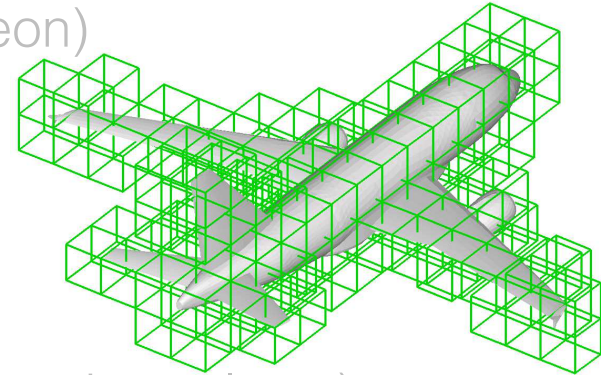
- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)
  - excellent accuracy, but expensive
- FMM Solver (Greengard & Rokhlin, 87)
  - Based on low-rank approximation



- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)
  - excellent accuracy, but expensive
- FMM Solver (Greengard & Rokhlin, 87)
  - Based on low-rank approximation
  - Only matrix-vector product (hence used in iterative solvers)



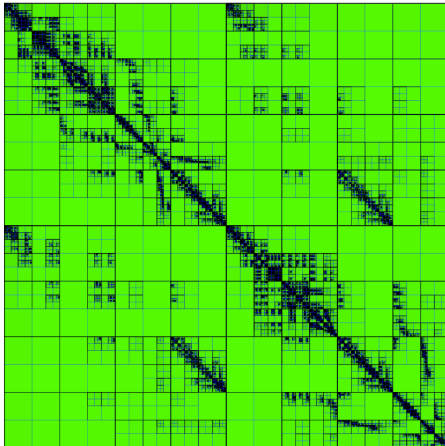
- Dense linear system
- Large number of unknowns (up to several millions)
- Direct solver (in-house, or OSS such as Chameleon)
  - excellent accuracy, but expensive
- FMM Solver (Greengard & Rokhlin, 87)
  - Based on low-rank approximation
  - Only matrix-vector product (hence used in iterative solvers)
  - much faster, but not fast enough (many iterations, many RHS)



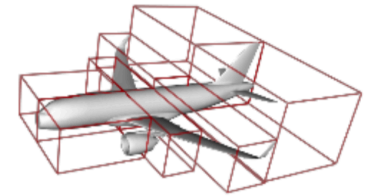
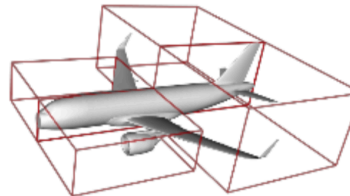
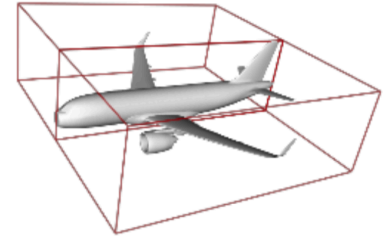
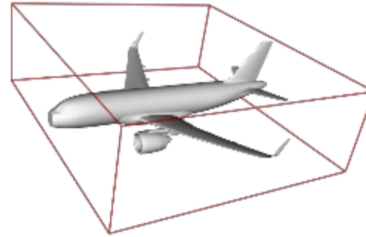
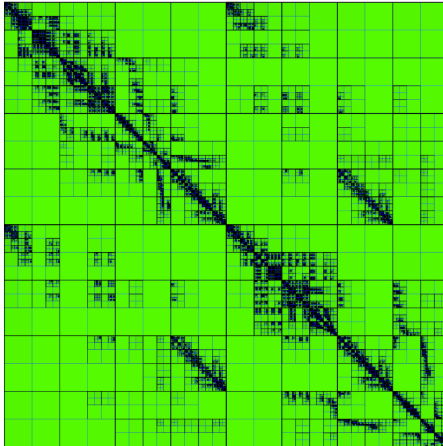




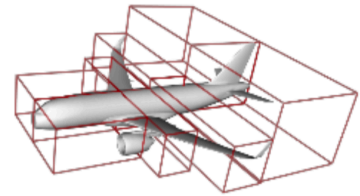
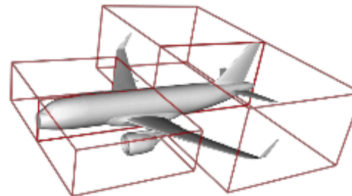
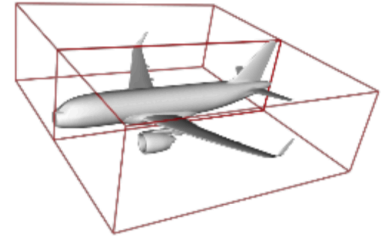
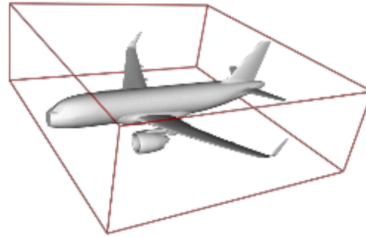
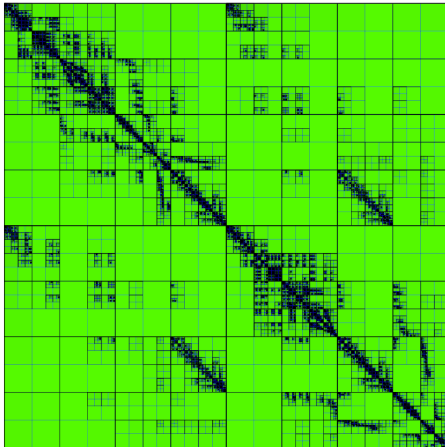
- Still based on Low-Rank properties



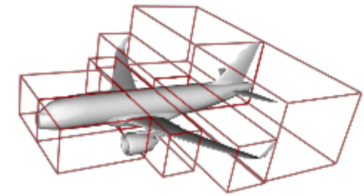
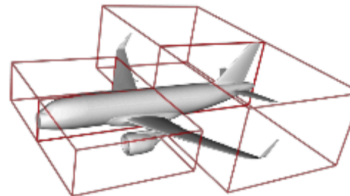
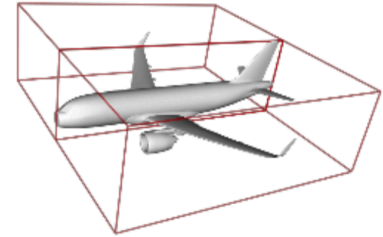
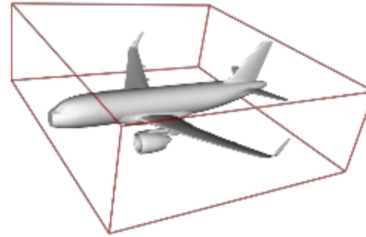
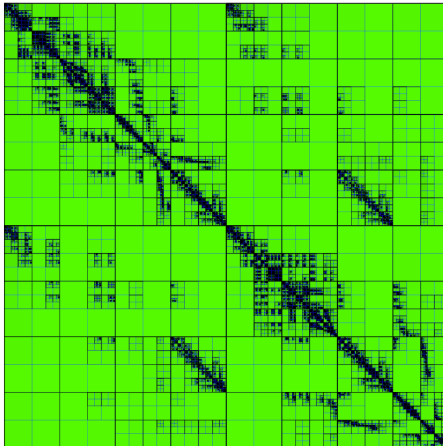
- Still based on Low-Rank properties
- Based on recursive bisection



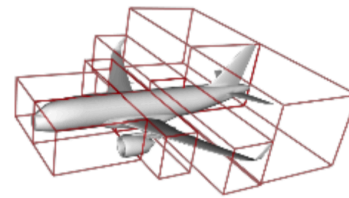
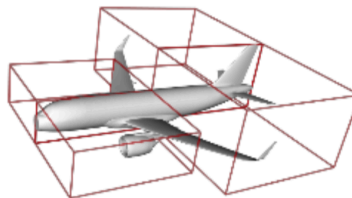
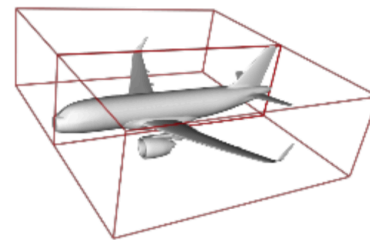
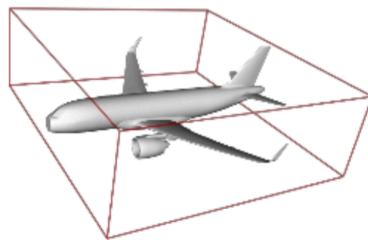
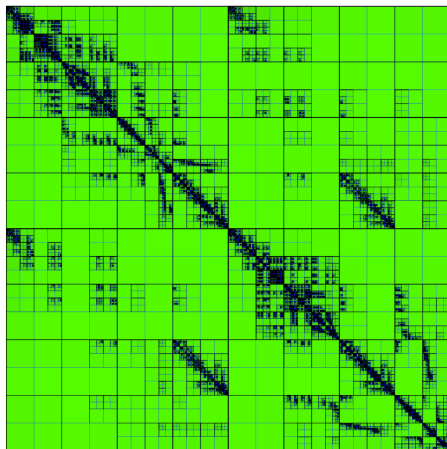
- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)



- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)
- Controlled accuracy, extremely fast, expensive in memory



- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)
- Controlled accuracy, extremely fast, expensive in memory
  - e.g. L.D.L<sup>t</sup> Facto 500k unknowns in 750 s on 32 cores, 42 Gb



- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)
- Controlled accuracy, extremely fast, expensive in memory
  - e.g. L.D.L<sup>t</sup> Facto 500k unknowns in 750 s on 32 cores, 42 Gb
- Packages available:

- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)
- Controlled accuracy, extremely fast, expensive in memory
  - e.g. L.D.L<sup>t</sup> Facto 500k unknowns in 750 s on 32 cores, 42 Gb
- Packages available:
  - HlibPro: parallel, free but not free;



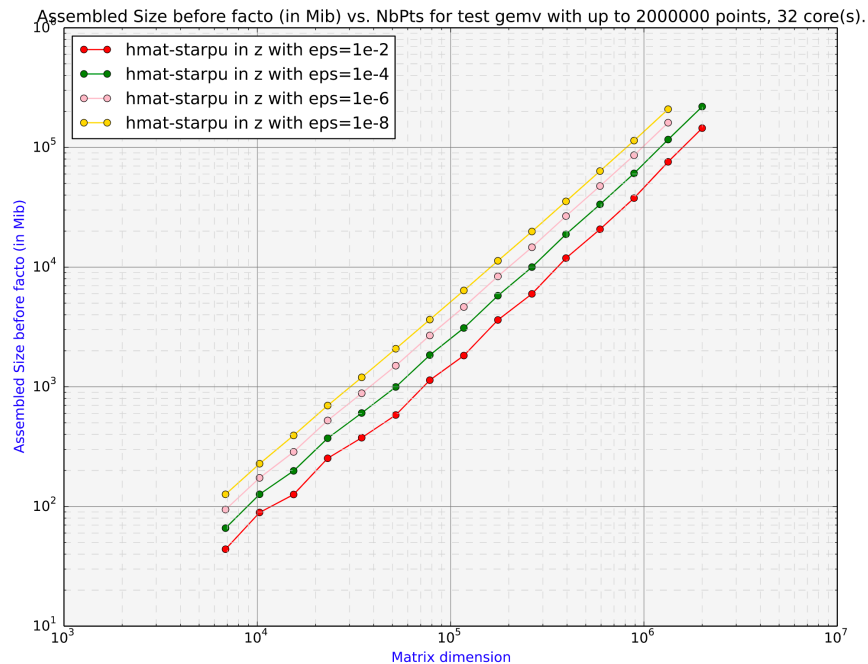
- Still based on Low-Rank properties
- Based on recursive bisection
- Leads to fast Mat-Vect product AND fast factorization (direct solver)
- Controlled accuracy, extremely fast, expensive in memory
  - e.g. L.D.L<sup>t</sup> Facto 500k unknowns in 750 s on 32 cores, 42 Gb
- Packages available:
  - HlibPro: parallel, free but not free;
  - Hmat-oss (airbus): sequential, free and free;

-

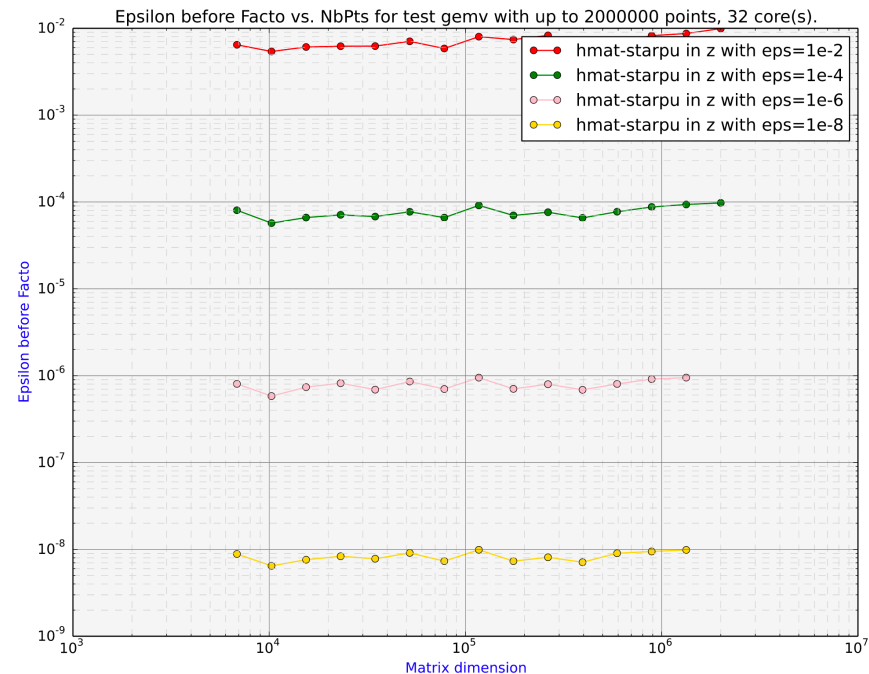
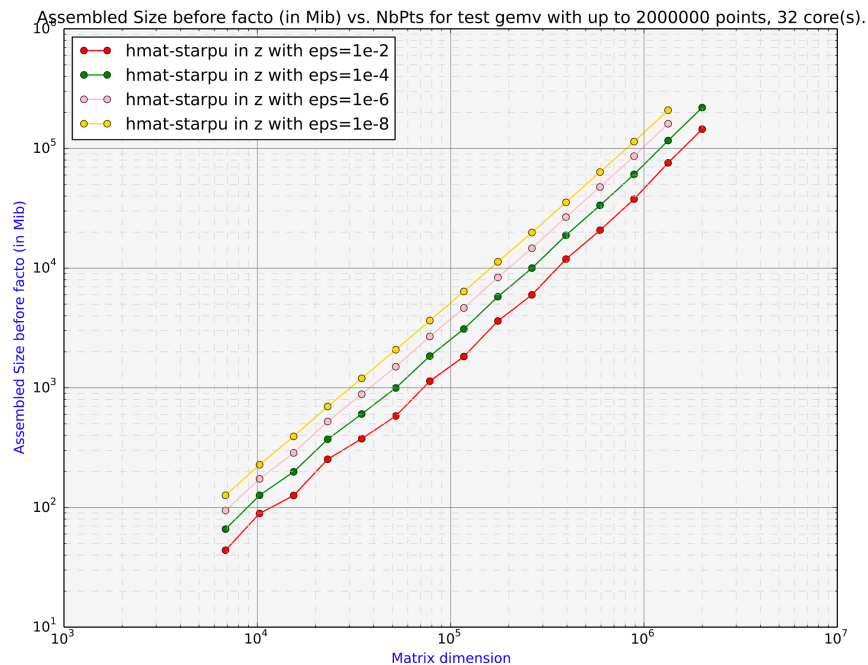


- Toy application with BEM-like kernel  $\exp(ikr)/r$

- Toy application with BEM-like kernel  $\exp(ikr)/r$
- Mat-Vec product before facto (size =  $n^{1.43}$ )



- Toy application with BEM-like kernel  $\exp(ikr)/r$
- Mat-Vec product before facto (size =  $n^{1,43}$ )



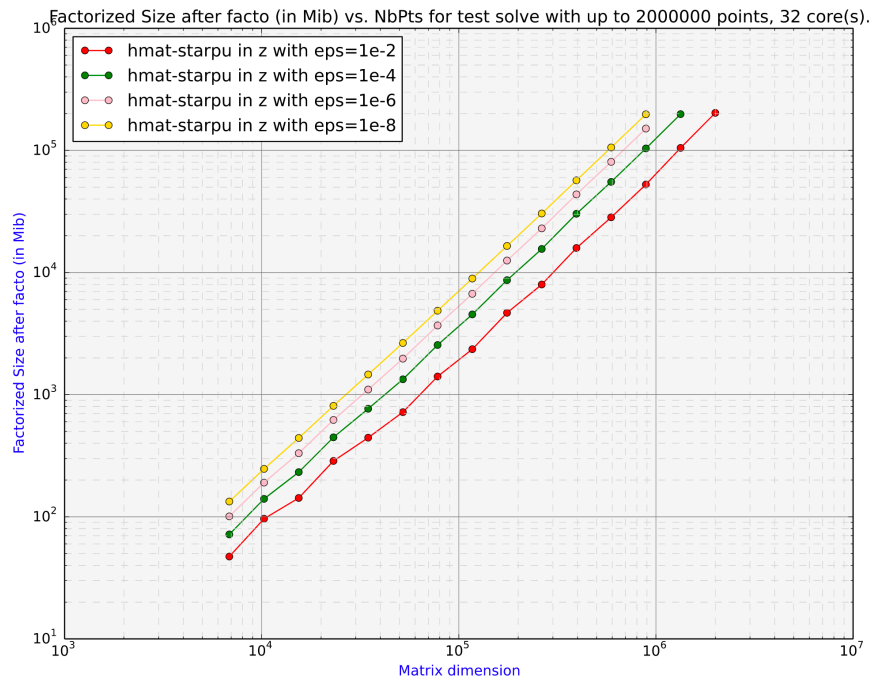


- Test on Factorization



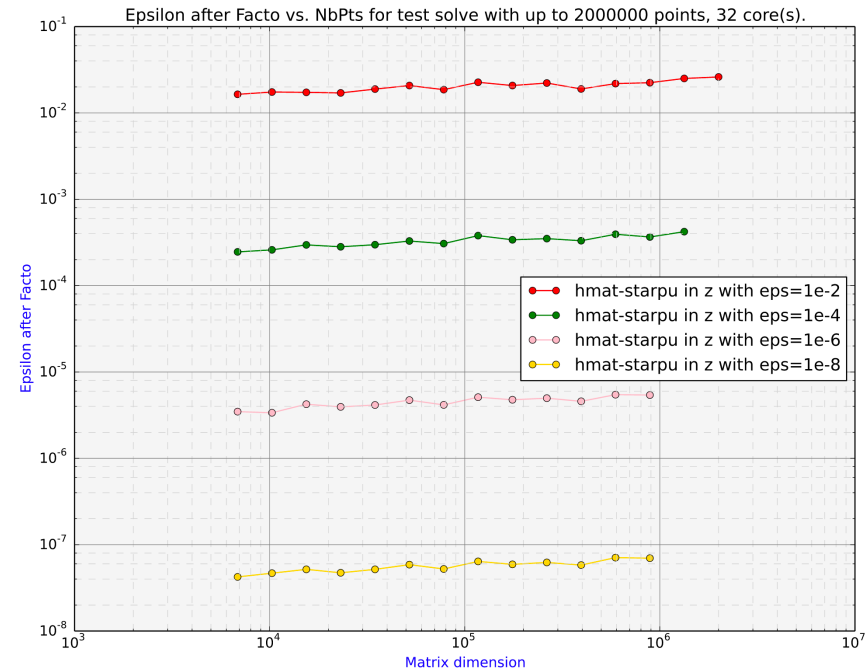
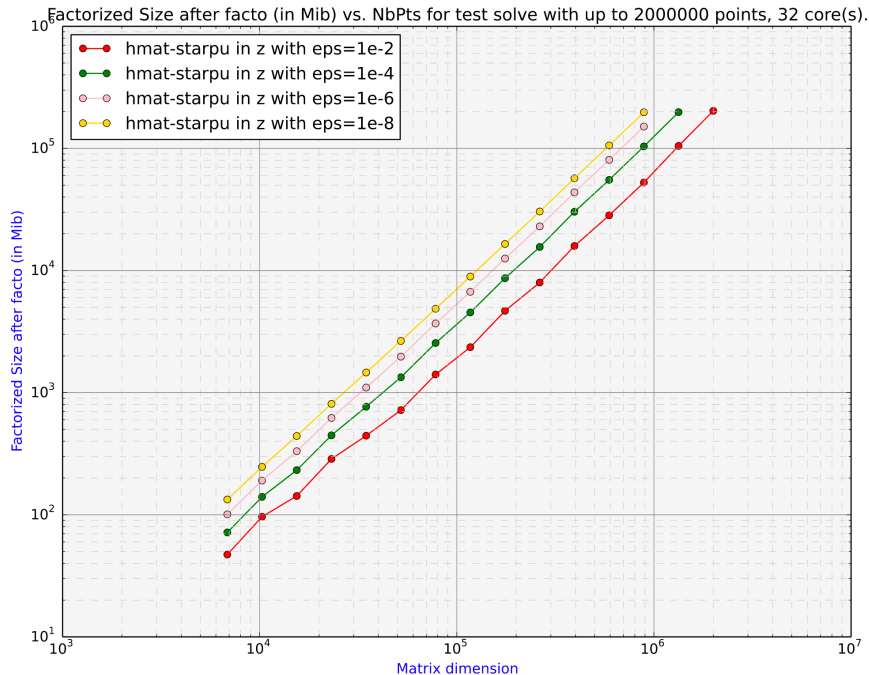
# H-Matrix performance (2)

- Test on Factorization
- size =  $n^{1,50}$



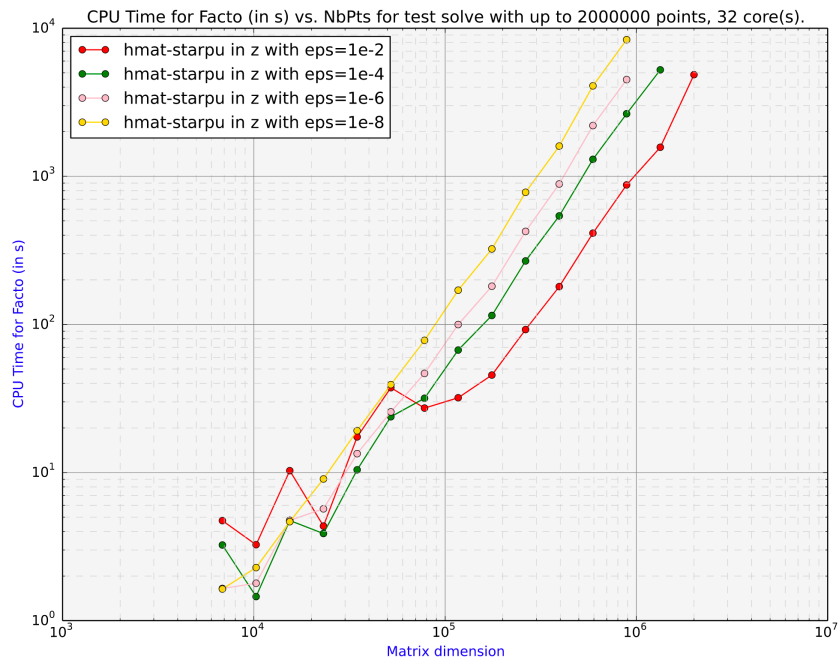
# H-Matrix performance (2)

- Test on Factorization
- size =  $n^{1,50}$

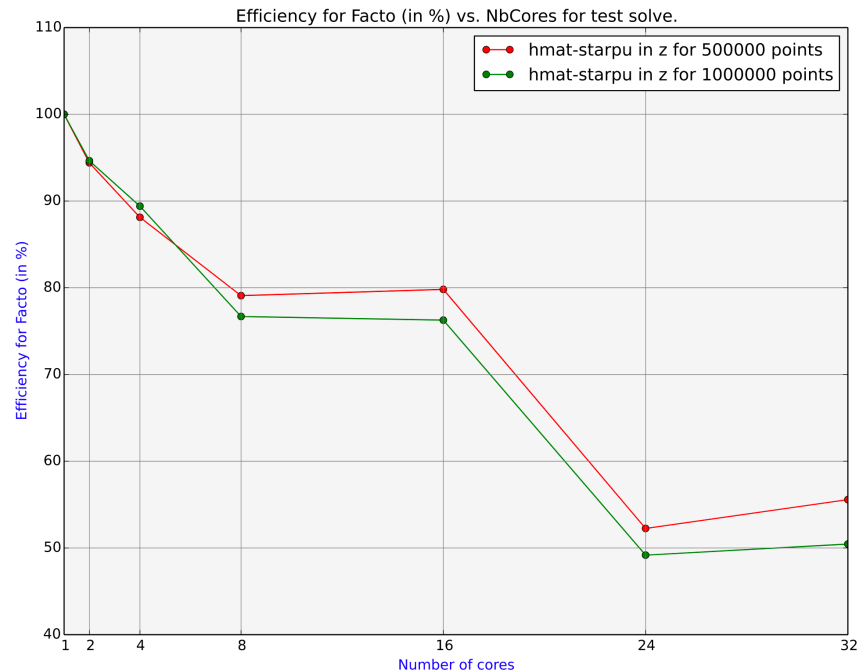
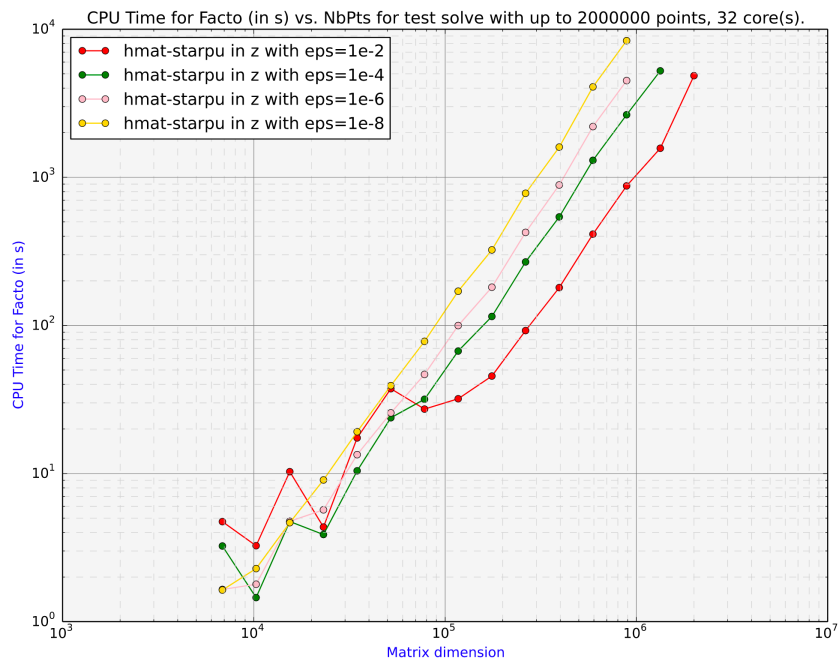




- Factorization: CPU time =  $n^{1,80}$



- Factorization: CPU time =  $n^{1,80}$



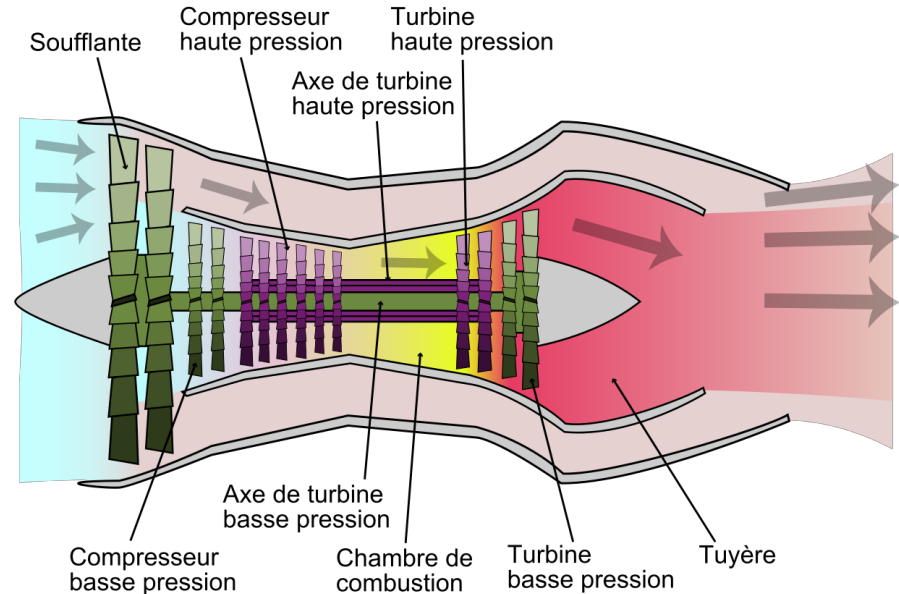


- Sufficient for forward radiation, not for rearward

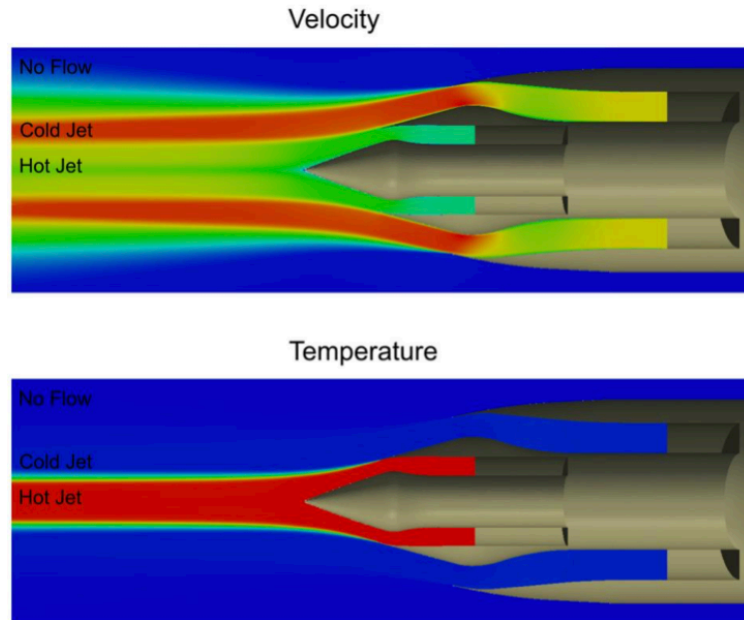
- Sufficient for forward radiation, not for rearward
- Propagation through jet flow (and interaction with the wing) must be taken into account



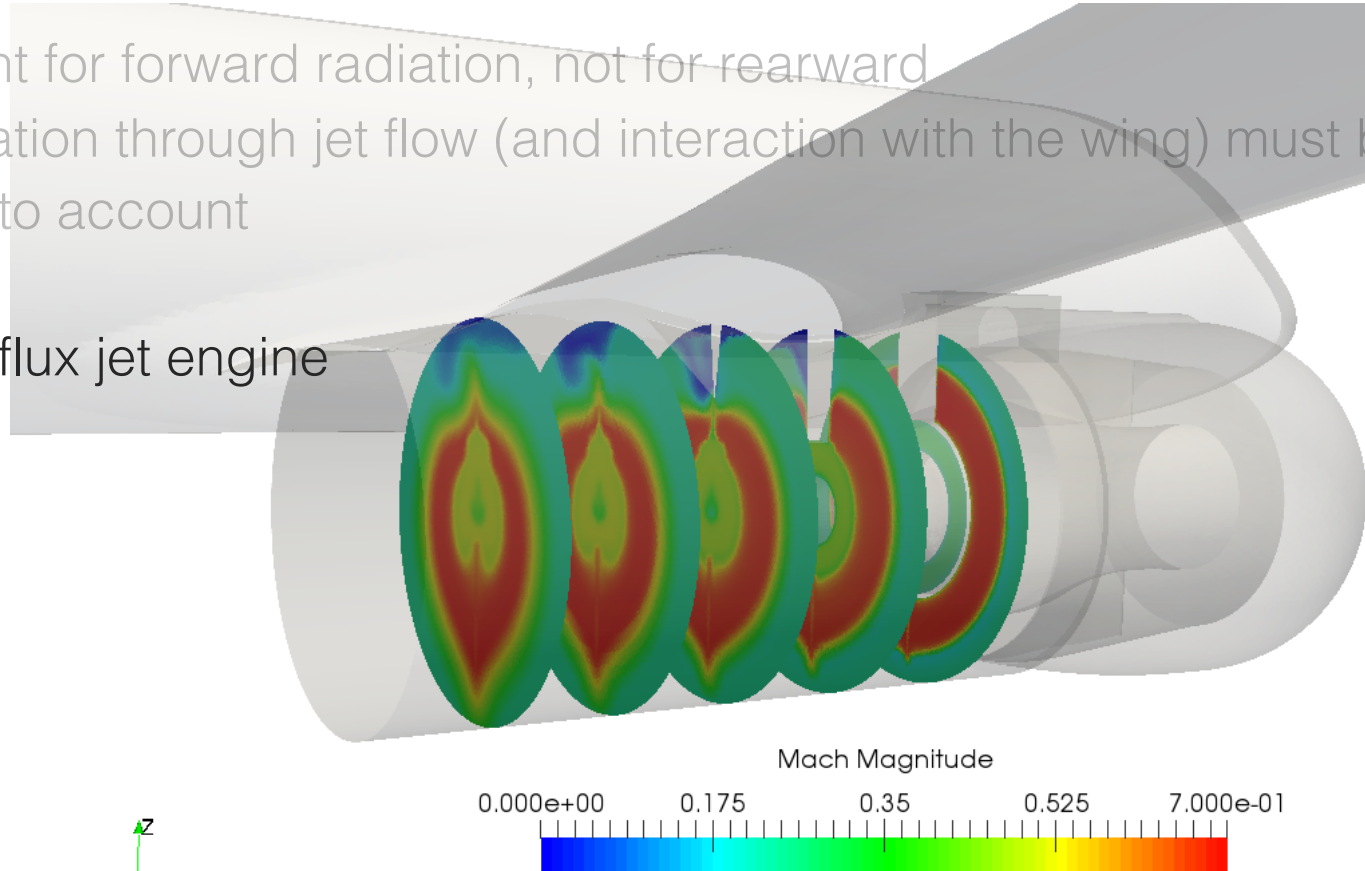
- Sufficient for forward radiation, not for rearward
- Propagation through jet flow (and interaction with the wing) must be taken into account
- Double flux jet engine



- Sufficient for forward radiation, not for rearward
- Propagation through jet flow (and interaction with the wing) must be taken into account
- Double flux jet engine



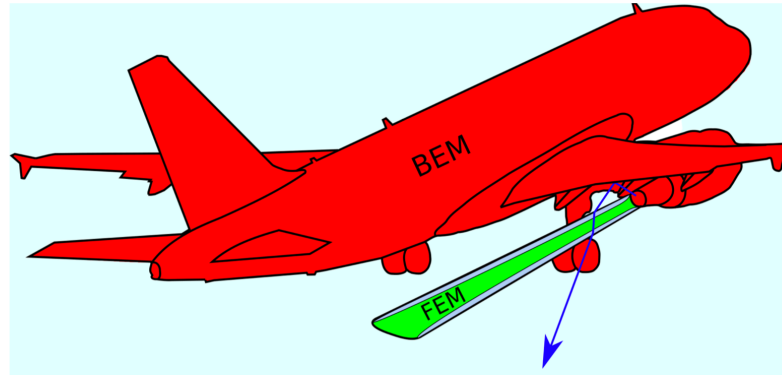
- Sufficient for forward radiation, not for rearward
- Propagation through jet flow (and interaction with the wing) must be taken into account
- Double flux jet engine



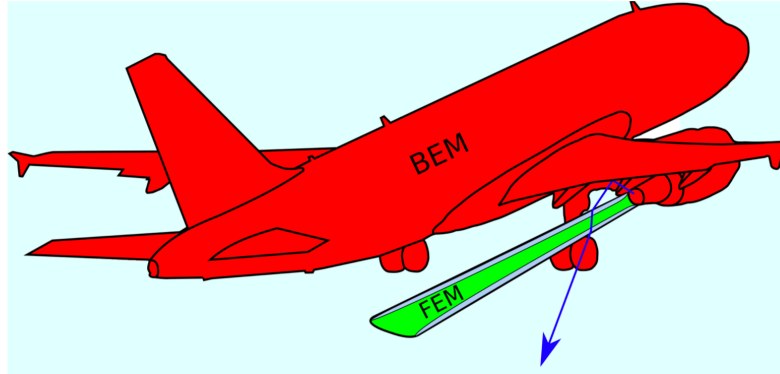




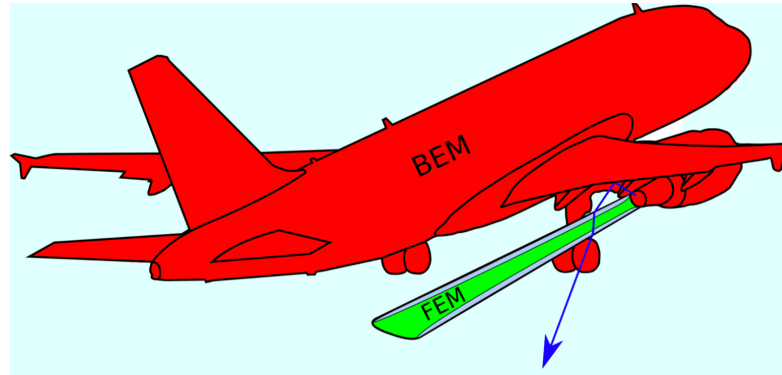
- Volume mesh (tetrahedras) in the jet flow (cylinder, potential flow)



- Volume mesh (tetrahedras) in the jet flow (cylinder, potential flow)
- Surface mesh = outer surface of the volume domain + the rest of the plane



- Volume mesh (tetrahedras) in the jet flow (cylinder, potential flow)
- Surface mesh = outer surface of the volume domain + the rest of the plane
- Same unknowns, same mesh: same linear system





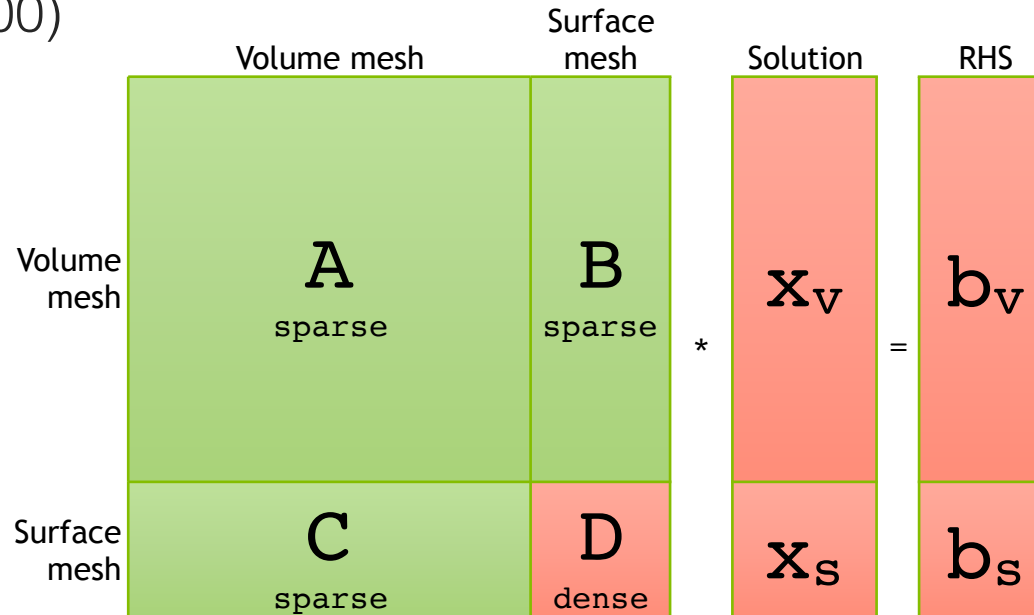


- Surface mesh: from 10k to 1M dofs

- Surface mesh: from 10k to 1M dofs
- Volume mesh: from 100k to 100M dofs

- Surface mesh: from 10k to 1M dofs
- Volume mesh: from 100k to 100M dofs
- Many RHS ( $>1000$ )

- Surface mesh: from 10k to 1M dofs
- Volume mesh: from 100k to 100M dofs
- Many RHS (>1000)





- Direct approach: Schur complement + dense solver

- Direct approach: Schur complement + dense solver
  - Build  $D - C.A^{-1}.B \rightarrow D'$  and  $b_s - C.A^{-1}.b_v \rightarrow b'_s$

$$\begin{bmatrix} D - C.A^{-1}.B \end{bmatrix} * \begin{bmatrix} x_s \end{bmatrix} = \begin{bmatrix} b'_s \end{bmatrix}$$



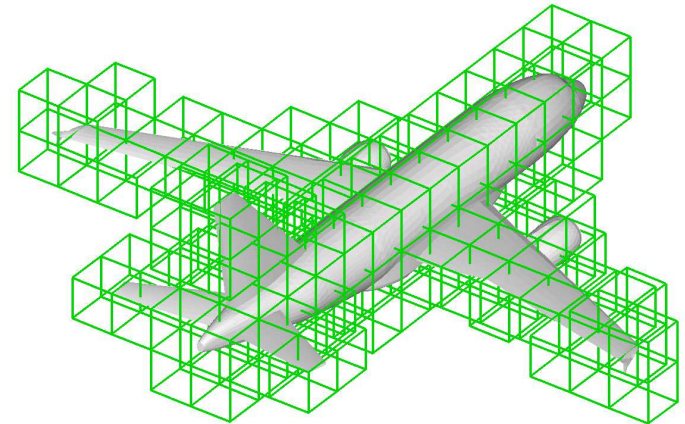
- Direct approach: Schur complement + dense solver
  - Build  $D - C.A^{-1}.B \rightarrow D'$  and  $b_s - C.A^{-1}.b_v \rightarrow b'_s$
  - Solve  $D'.x_s = b'_s$

$$\begin{bmatrix} D - C.A^{-1}.B \end{bmatrix} * \begin{bmatrix} x_s \end{bmatrix} = \begin{bmatrix} b'_s \end{bmatrix}$$

- Direct approach: Schur complement + dense solver
  - Build  $D - C.A^{-1}.B \rightarrow D'$  and  $b_s - C.A^{-1}.b_v \rightarrow b'_s$
  - Solve  $D'.x_s = b'_s$

$$\begin{matrix} D - \\ C.A^{-1}.B \end{matrix} * \mathbf{x}_s = \mathbf{b}'_s$$

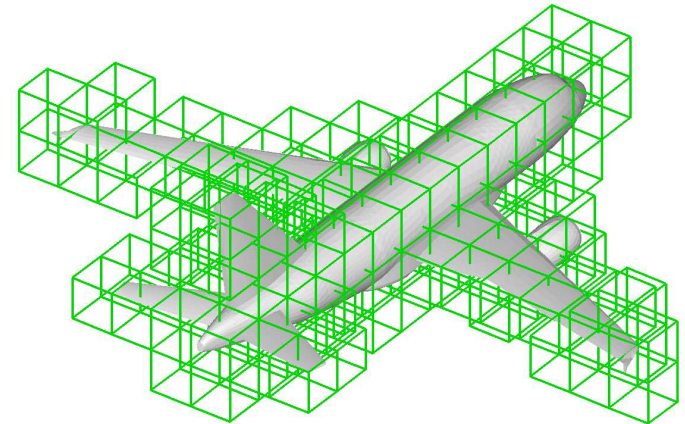
- FMM-based approach: iterative solver using FMM and on-the-fly Schur complement



- Direct approach: Schur complement + dense solver
  - Build  $D - C.A^{-1}.B \rightarrow D'$  and  $b_s - C.A^{-1}.b_v \rightarrow b'_s$
  - Solve  $D'.x_s = b'_s$

$$\begin{bmatrix} D - C.A^{-1}.B \end{bmatrix} * \begin{bmatrix} x_s \end{bmatrix} = \begin{bmatrix} b'_s \end{bmatrix}$$

- FMM-based approach: iterative solver using FMM and on-the-fly Schur complement
  - Solve iteratively  $(D_{FMM} - C.A^{-1}.B).x_s = b'_s$

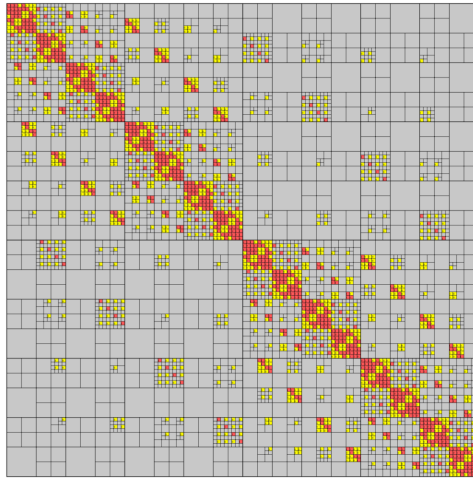




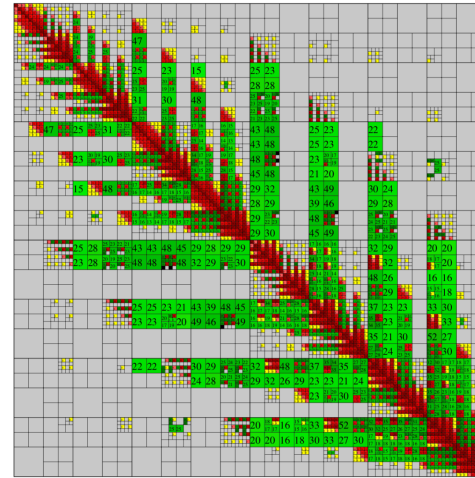
- We can use H-matrix on volume mesh

- We can use H-matrix on volume mesh
- Sparse factorisation leads to fill-in *which is low rank*

- We can use H-matrix on volume mesh
- Sparse factorisation leads to fill-in *which is low rank*

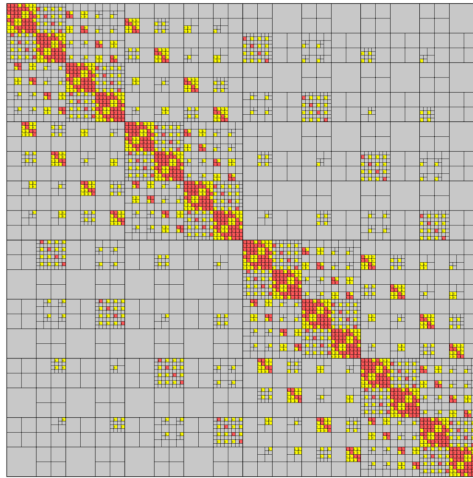


(a) Sparse matrix before factorization, with a bisection clustering.

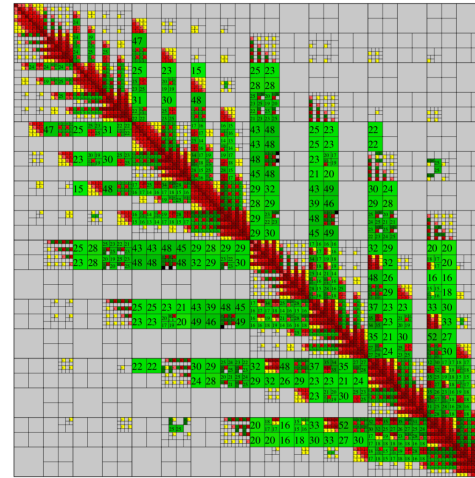


(b) Matrix after factorization, with a bisection clustering.

- We can use H-matrix on volume mesh
- Sparse factorisation leads to fill-in *which is low rank*
- Note: separates volume and surface parts



(a) Sparse matrix before factorization, with a bisection clustering.



(b) Matrix after factorization, with a bisection clustering.





- Excellent speed-up vs. previous methods

- Excellent speed-up vs. previous methods
- Example 1: Surface=130kdof, volume=300kdof, RHS=122
  - Direct Solver: 9h on 48 cores
  - H-mat Solver: 20min on 24 cores
  - Speedup: x55

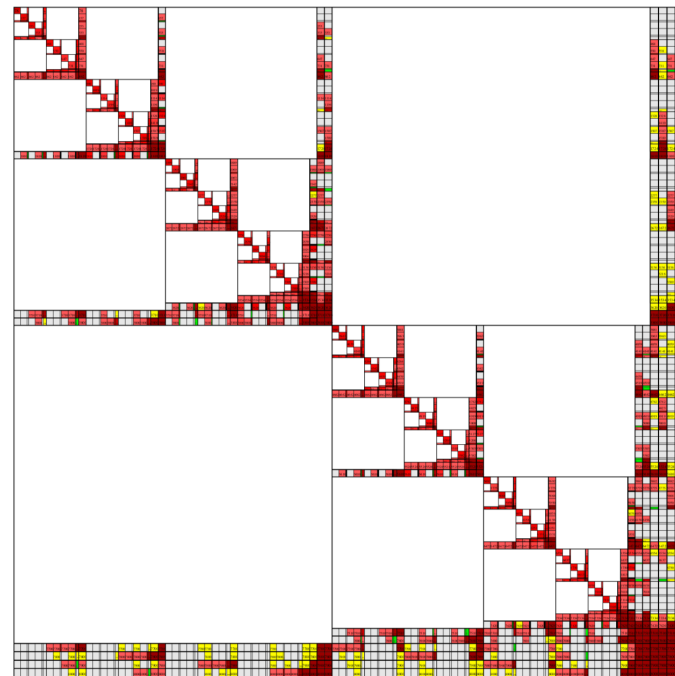
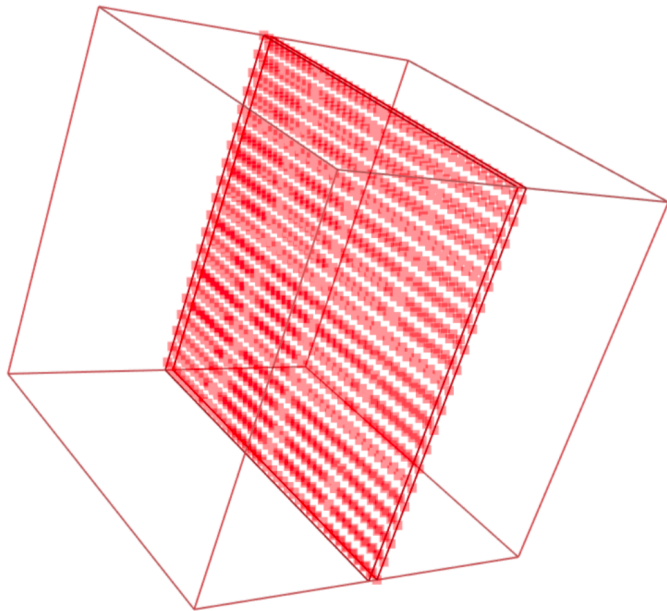
- Excellent speed-up vs. previous methods
- Example 1: Surface=130kdof, volume=300kdof, RHS=122
  - Direct Solver: 9h on 48 cores
  - H-mat Solver: 20min on 24 cores
  - Speedup: x55
- Example 2: Surface=600kdof, volume=4Mdof, RHS=2896, 4 problems
  - FMM Solver: 65h on 192 cores ( $\approx 90$  iterations)
  - H-mat Solver: 12h on 24 cores (mat sizes after facto: 16+18+66 Gb)
  - Speedup: x41



- Use ideas from the "sparse solver" community

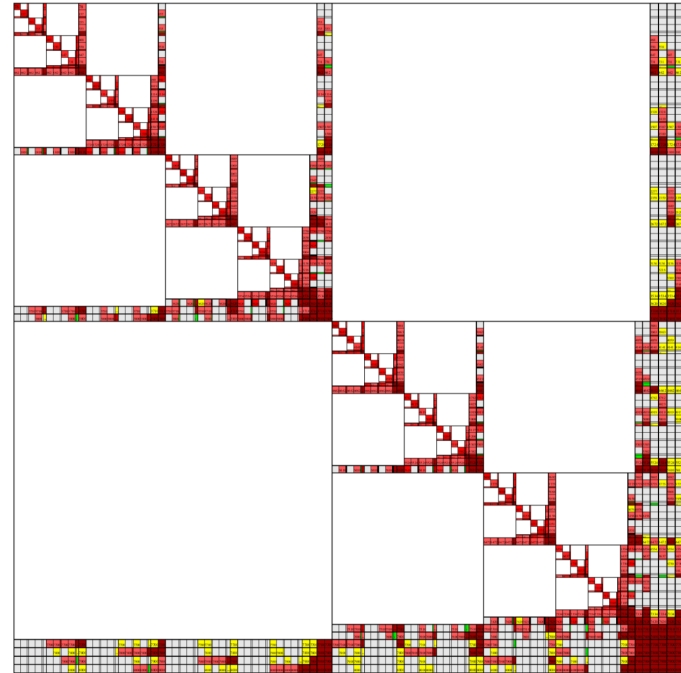
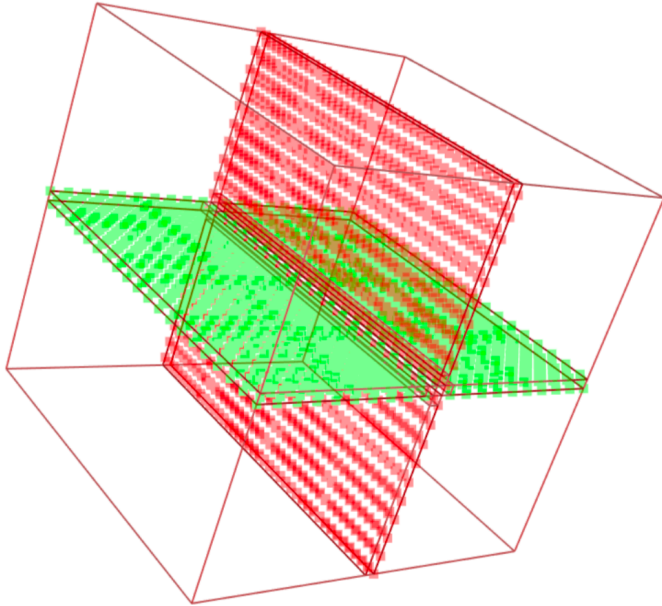
- Use ideas from the "sparse solver" community
  - Nested dissection (minimize fill-in)

- Use ideas from the "sparse solver" community
  - Nested dissection (minimize fill-in)

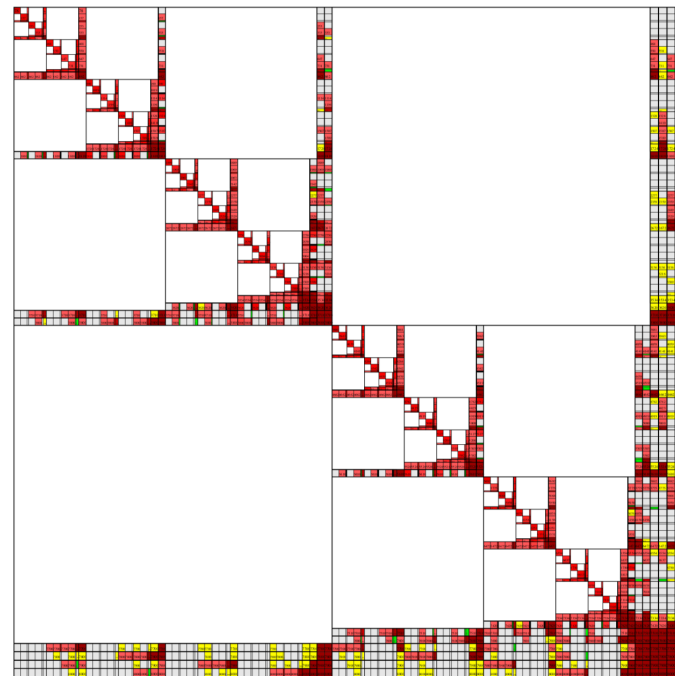
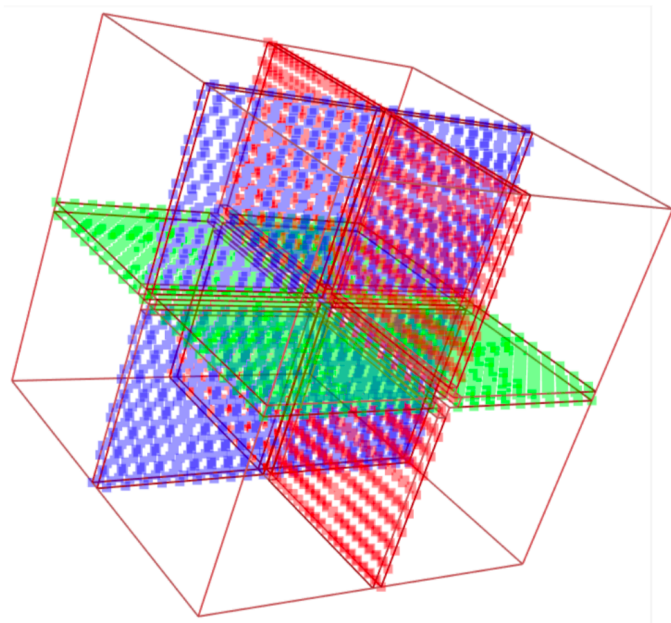




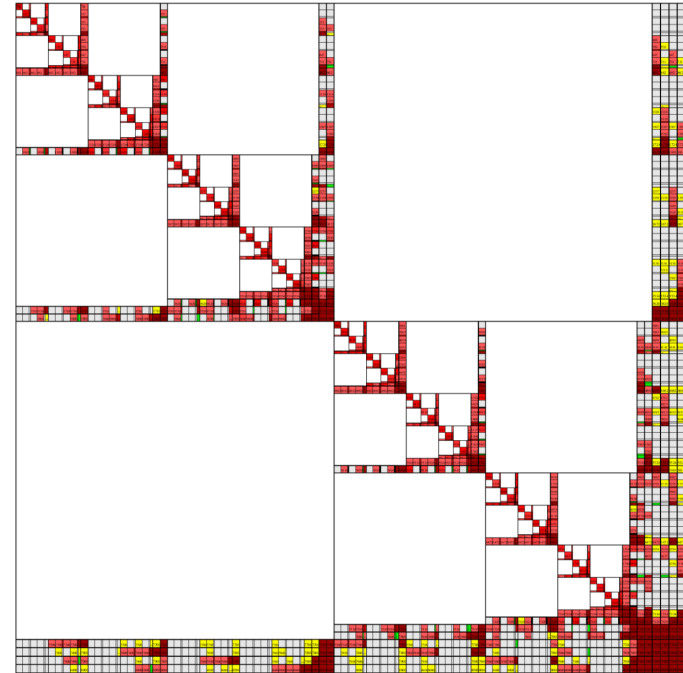
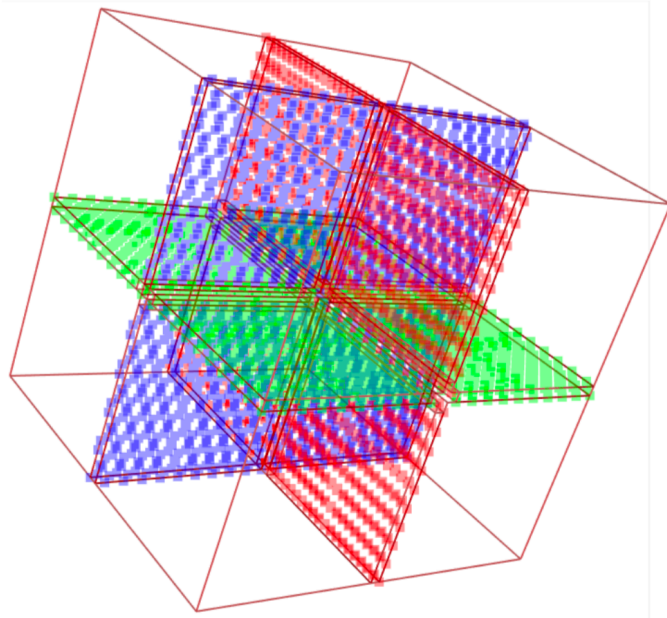
- Use ideas from the "sparse solver" community
  - Nested dissection (minimize fill-in)



- Use ideas from the "sparse solver" community
  - Nested dissection (minimize fill-in)



- Use ideas from the "sparse solver" community
  - Nested dissection (minimize fill-in)
  - Symbolic factorization (optimize storage)





- Mixed Ternary/Binary cluster tree

- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:

- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:
  - Faster assembly (less blocks, less data-sparsity)

- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:
  - Faster assembly (less blocks, less data-sparsity)
  - For Starpu: Less data, less dependencies



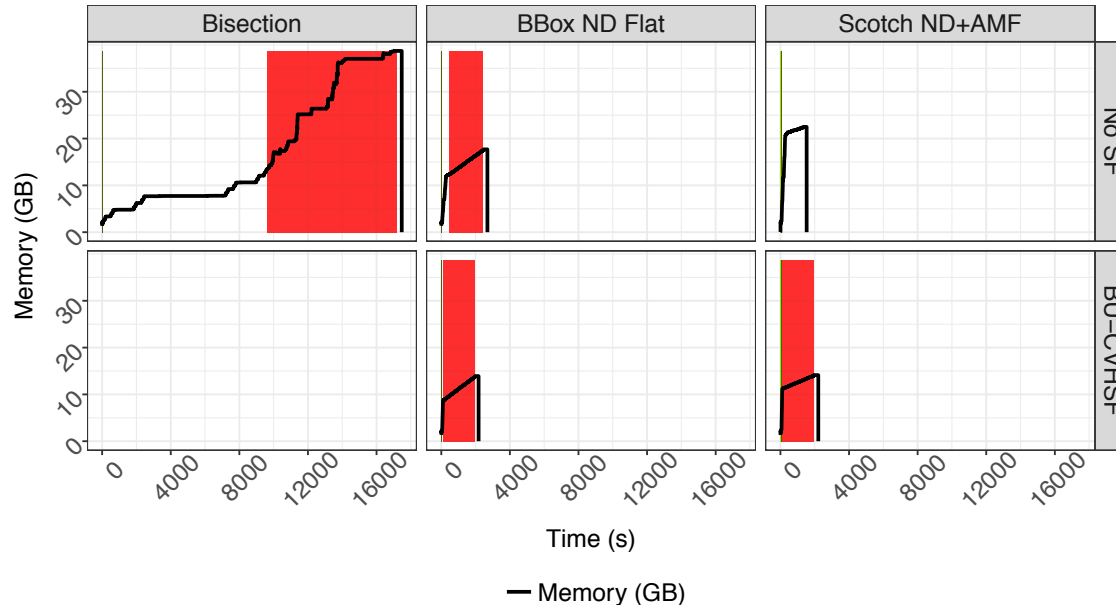
- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:
  - Faster assembly (less blocks, less data-sparsity)
  - For Starpu: Less data, less dependencies
  - Independent factorisation of subdomains

- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:
  - Faster assembly (less blocks, less data-sparsity)
  - For Starpu: Less data, less dependencies
  - Independent factorisation of subdomains
- New admissibility criteria

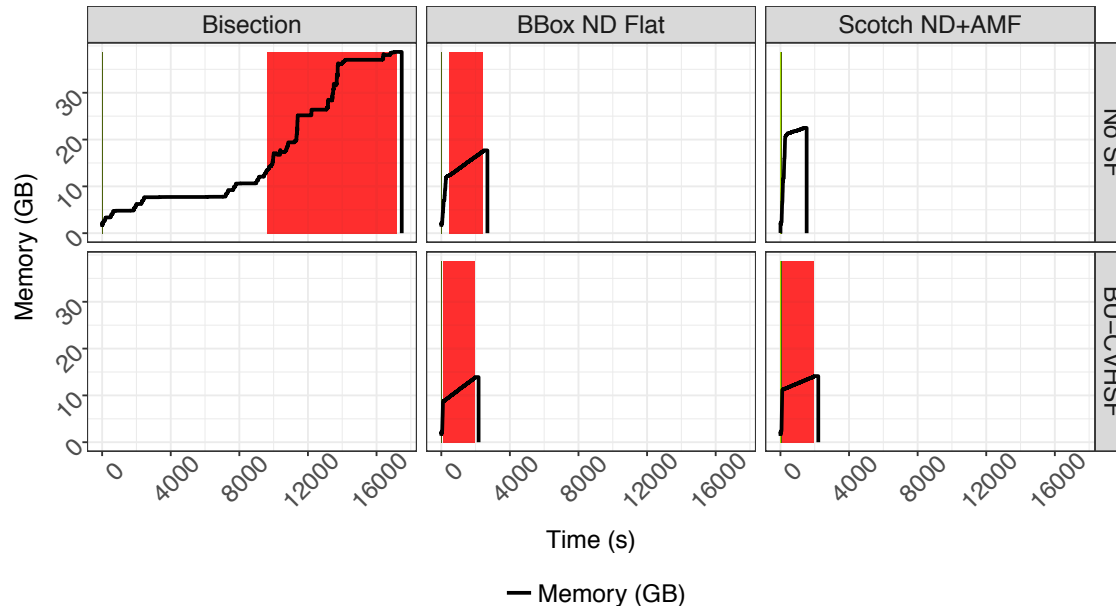
- Mixed Ternary/Binary cluster tree
- Big block of zeros that will remain during facto:
  - Faster assembly (less blocks, less data-sparsity)
  - For Starpu: Less data, less dependencies
  - Independent factorisation of subdomains
- New admissibility criteria
- New rank estimator



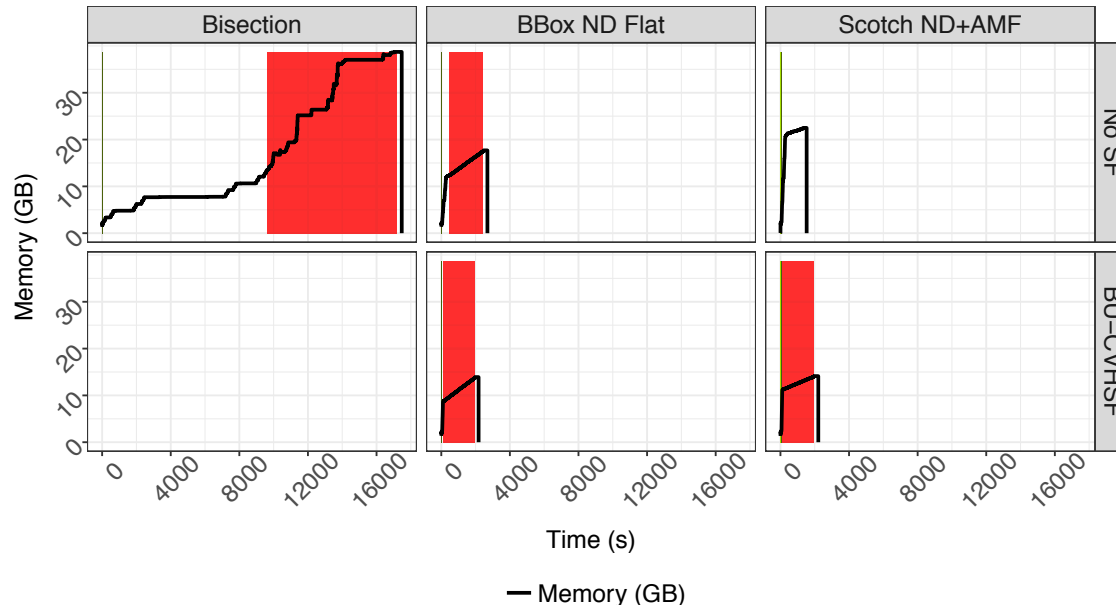
- As expected, reduces time and memory consumption



- As expected, reduces time and memory consumption
- WIP, to get closer to native sparse solvers on pure FEM problems



- As expected, reduces time and memory consumption
- WIP, to get closer to native sparse solvers on pure FEM problems
- For FEM-BEM, this is the current best approach

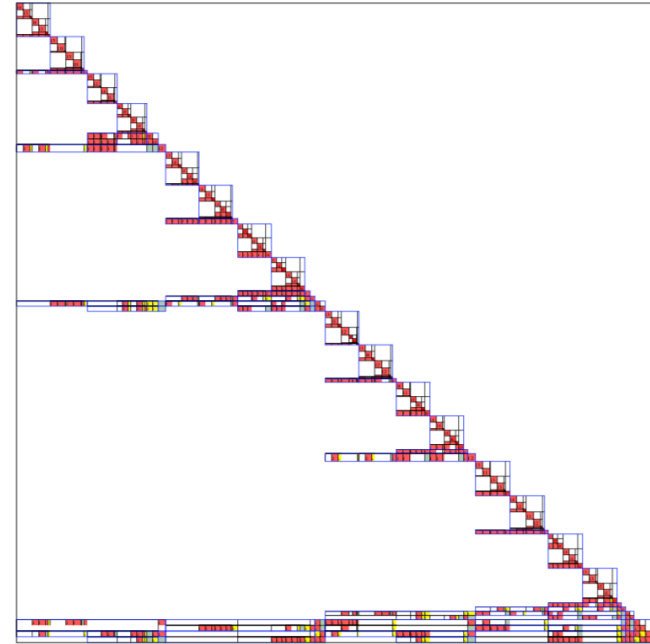




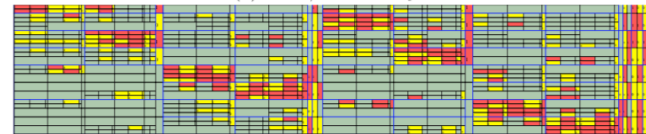


- Mumps-Hmat coupling: multi-facto based on H-mat clustering

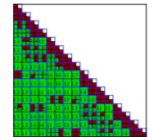
- Mumps-Hmat coupling: multifactor based on H-mat clustering
- Unique Hmat for surface AND volume



(a) FEM/FEM subsystem.

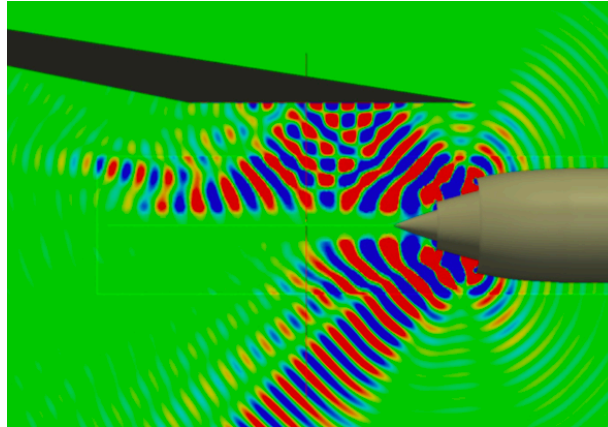


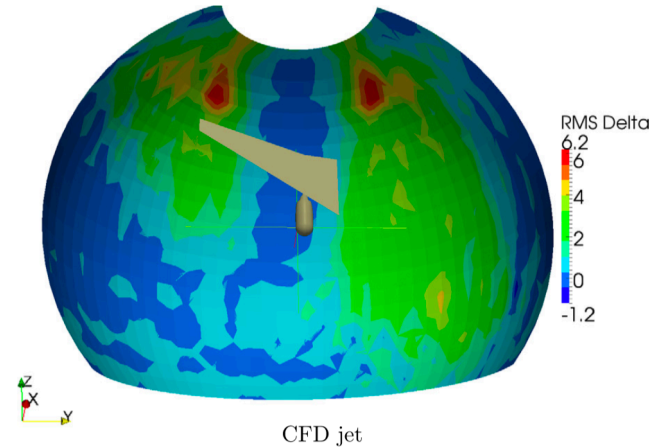
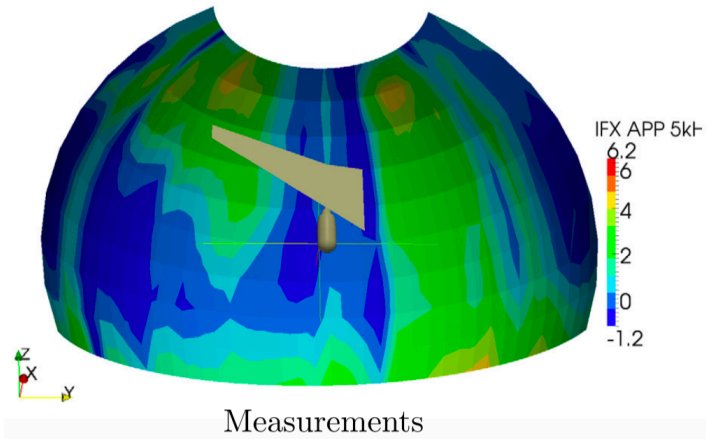
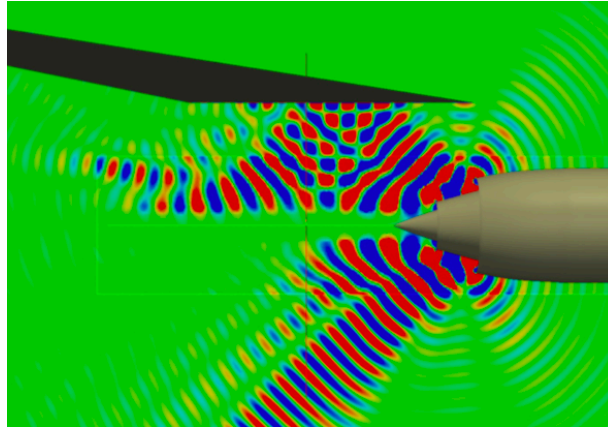
(b) BEM/FEM subsystem.



(c) BEM/BEM subsystem.









- Aeroacoustic simulation is of first importance for certification and acceptance

- Aeroacoustic simulation is of first importance for certification and acceptance
- FEM-BEM modelisation is a good compromise between cost and accuracy



- Aeroacoustic simulation is of first importance for certification and acceptance
- FEM-BEM modelisation is a good compromise between cost and accuracy
- H-matrix solver shows excellent performance both on BEM only or coupled FEM-BEM systems



- On the Solver side :

- On the Solver side :
  - Handle surface and volume in the same h-matrix

- On the Solver side :
  - Handle surface and volume in the same h-matrix
  - Improve parallel h-matrix solver to treat larger test cases

- On the Solver side :
  - Handle surface and volume in the same h-matrix
  - Improve parallel h-matrix solver to treat larger test cases
- On the formulation side:

- On the Solver side :
  - Handle surface and volume in the same h-matrix
  - Improve parallel h-matrix solver to treat larger test cases
- On the formulation side:
  - Higher order methods in the jet flow

- On the Solver side :
  - Handle surface and volume in the same h-matrix
  - Improve parallel h-matrix solver to treat larger test cases
- On the formulation side:
  - Higher order methods in the jet flow
  - New sources models in the flow



- On the Solver side :
  - Handle surface and volume in the same h-matrix
  - Improve parallel h-matrix solver to treat larger test cases
- On the formulation side:
  - Higher order methods in the jet flow
  - New sources models in the flow
- PhD position in Inria Bordeaux for HPC / algorithm