WHY THIS COURSE

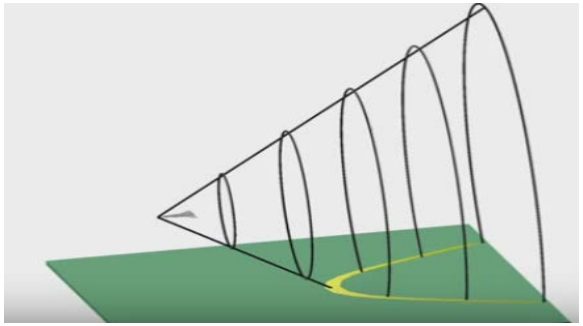
L. HALPERN, J. RYAN

ONERA
THE FRENCH AEROSPACE LAB

UNIVERSITE PARIS 13 Mathématiques

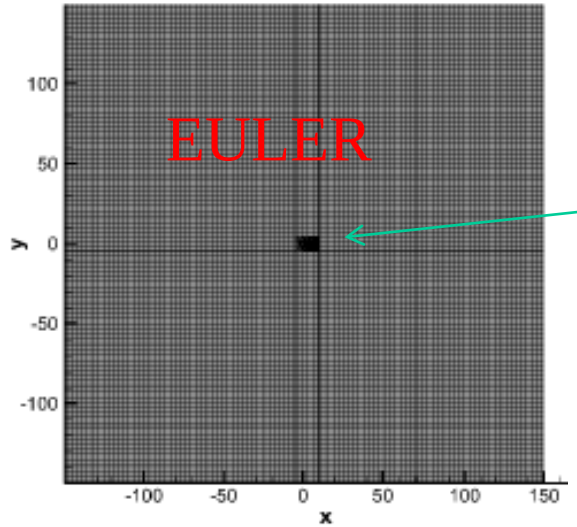
Probleme d'Aéro-acoustique

Un avion se déplaçant à vitesse supersonique génère des perturbations (bruit) qui se propagent depuis la source à travers l'atmosphère et persistent jusqu'au sol.

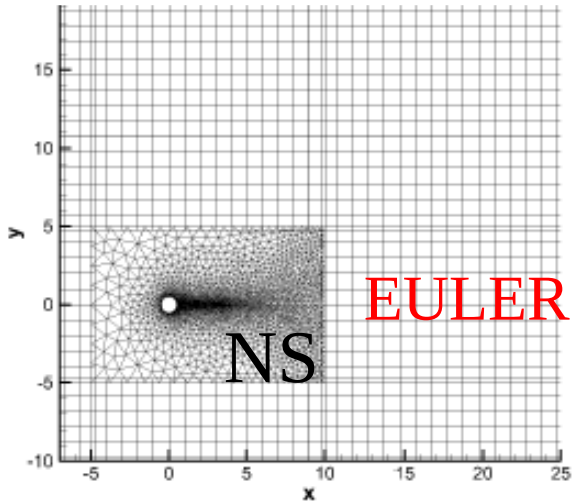


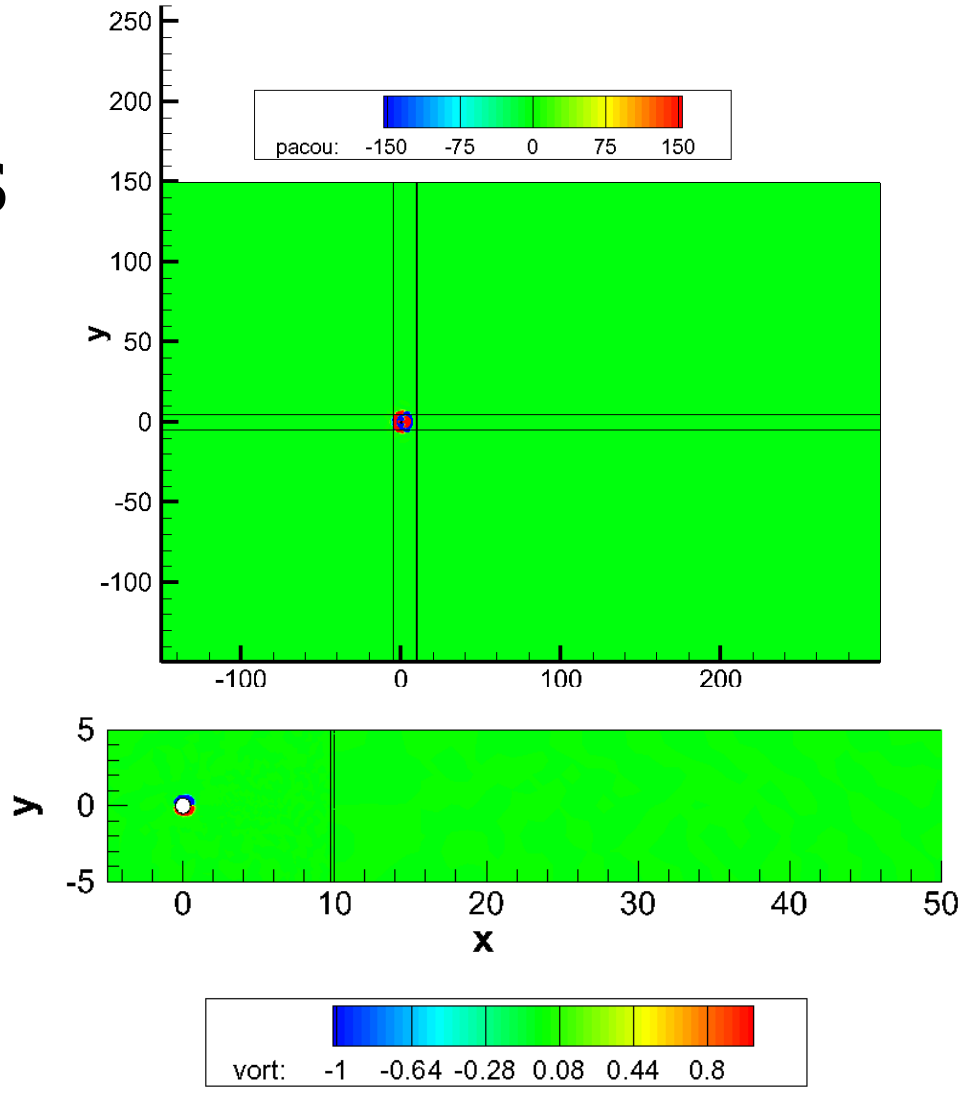
Prédire et réduire ce bruit sont essentiels pour l'environnement et constitue un élément clé du processus de conception d'un avion supersonique.

Low-Reynolds Number Flow around a Cylinder
Re=500 , DG-P2 results



NS

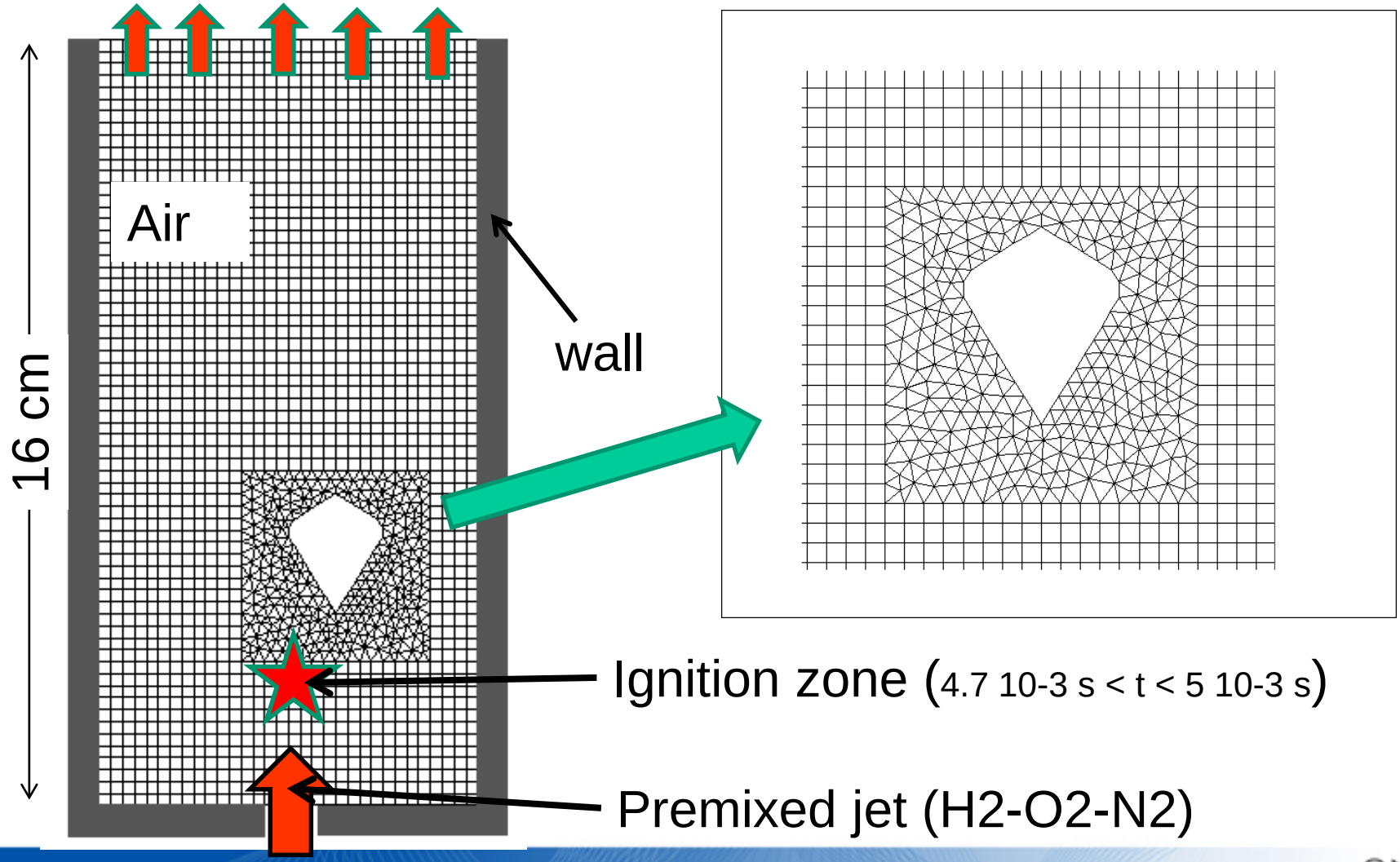


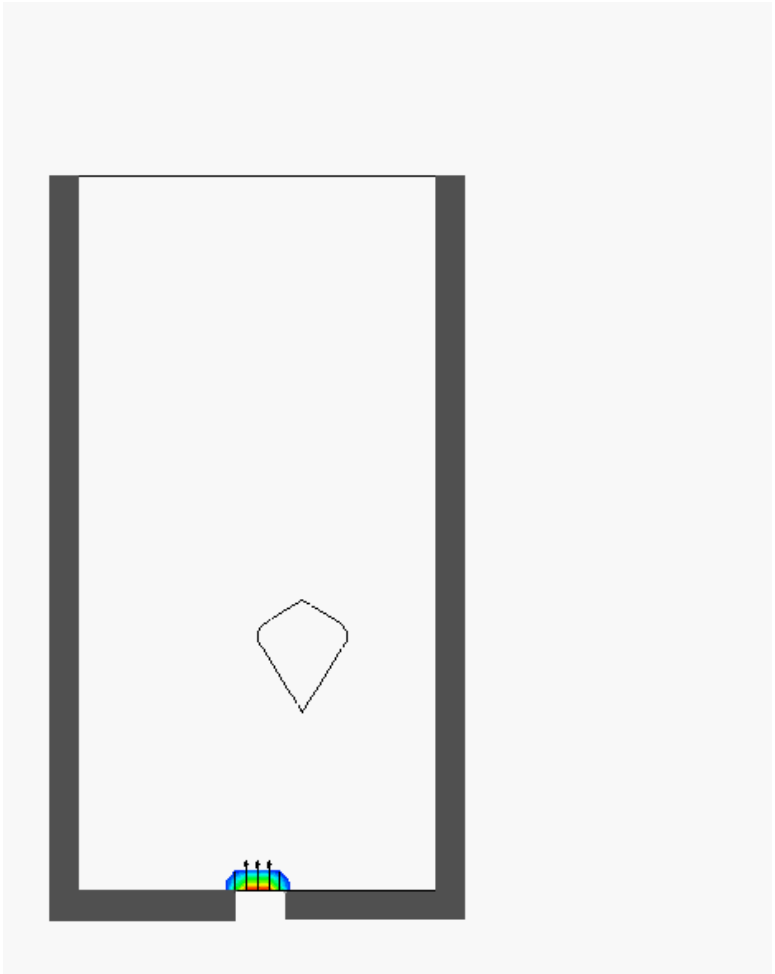


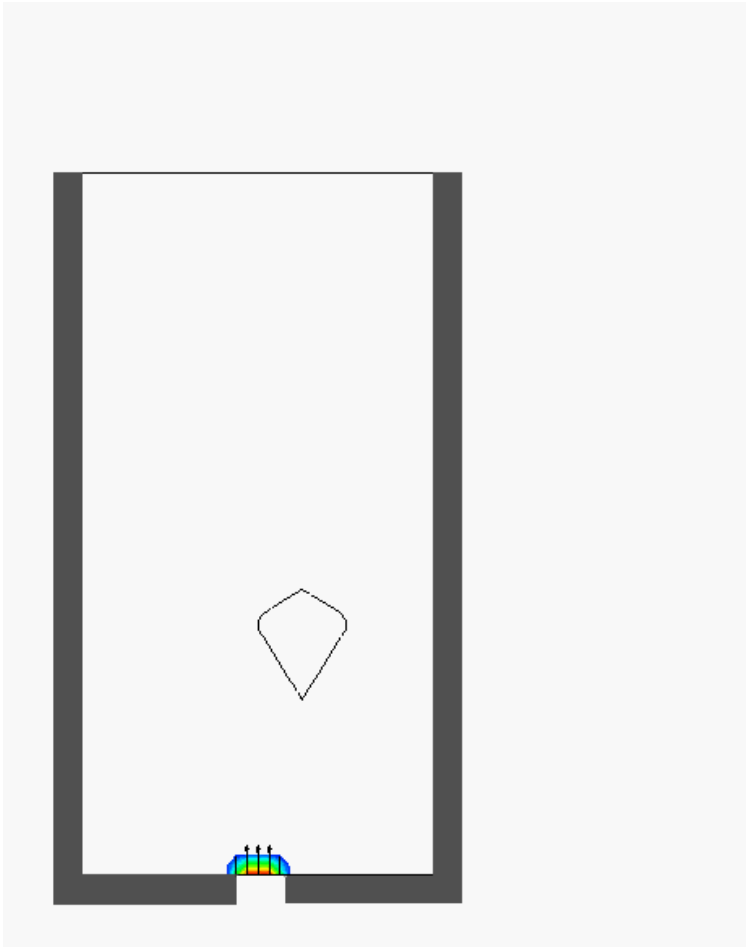
ONERA

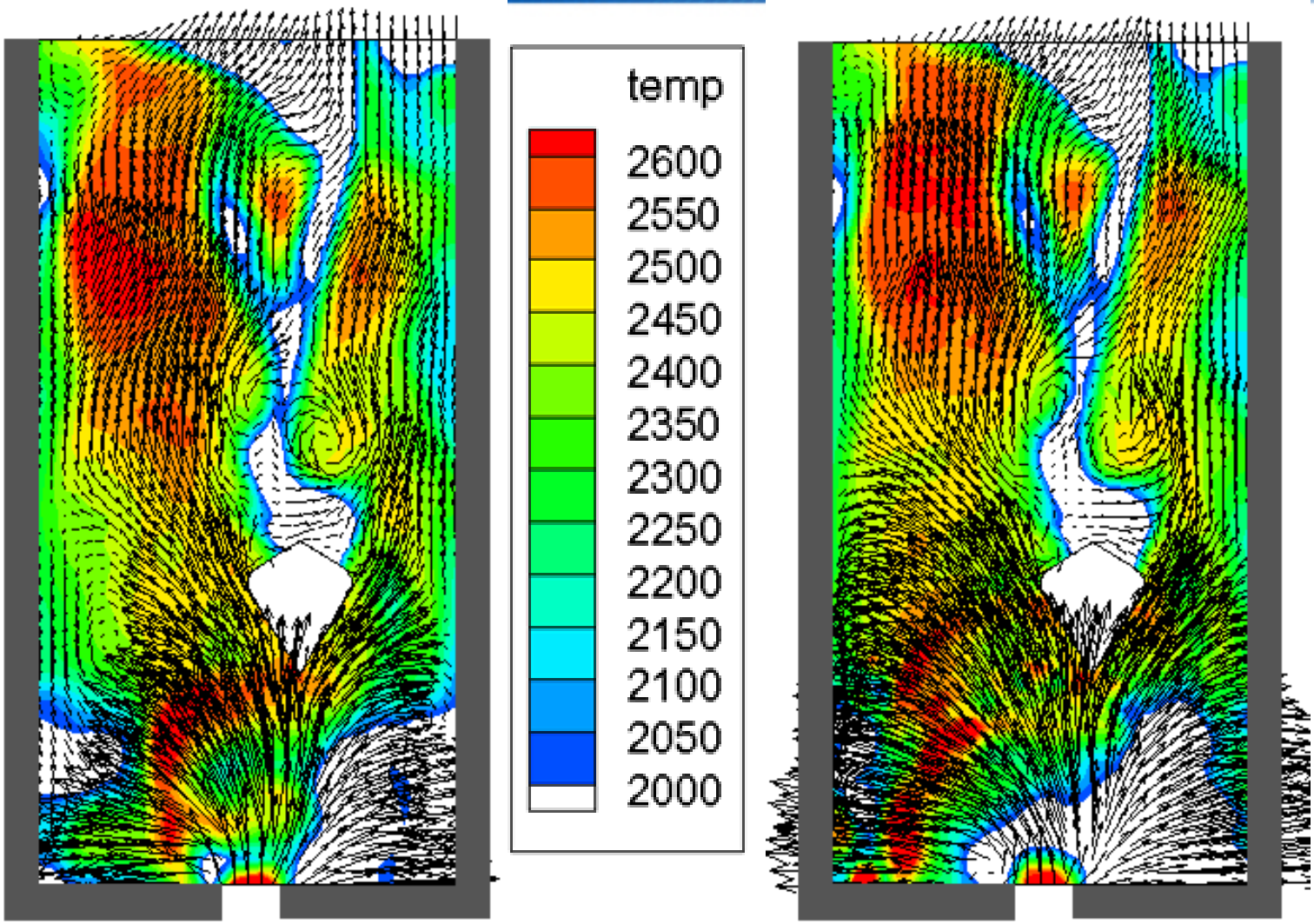
2D premixed flame in a combustion chamber

- 1792 Cartesian cells + 570 triangular elements
- Cockburn's limiter on mass fractions every time step
- Cockburn's limiter on other variables every 100 it

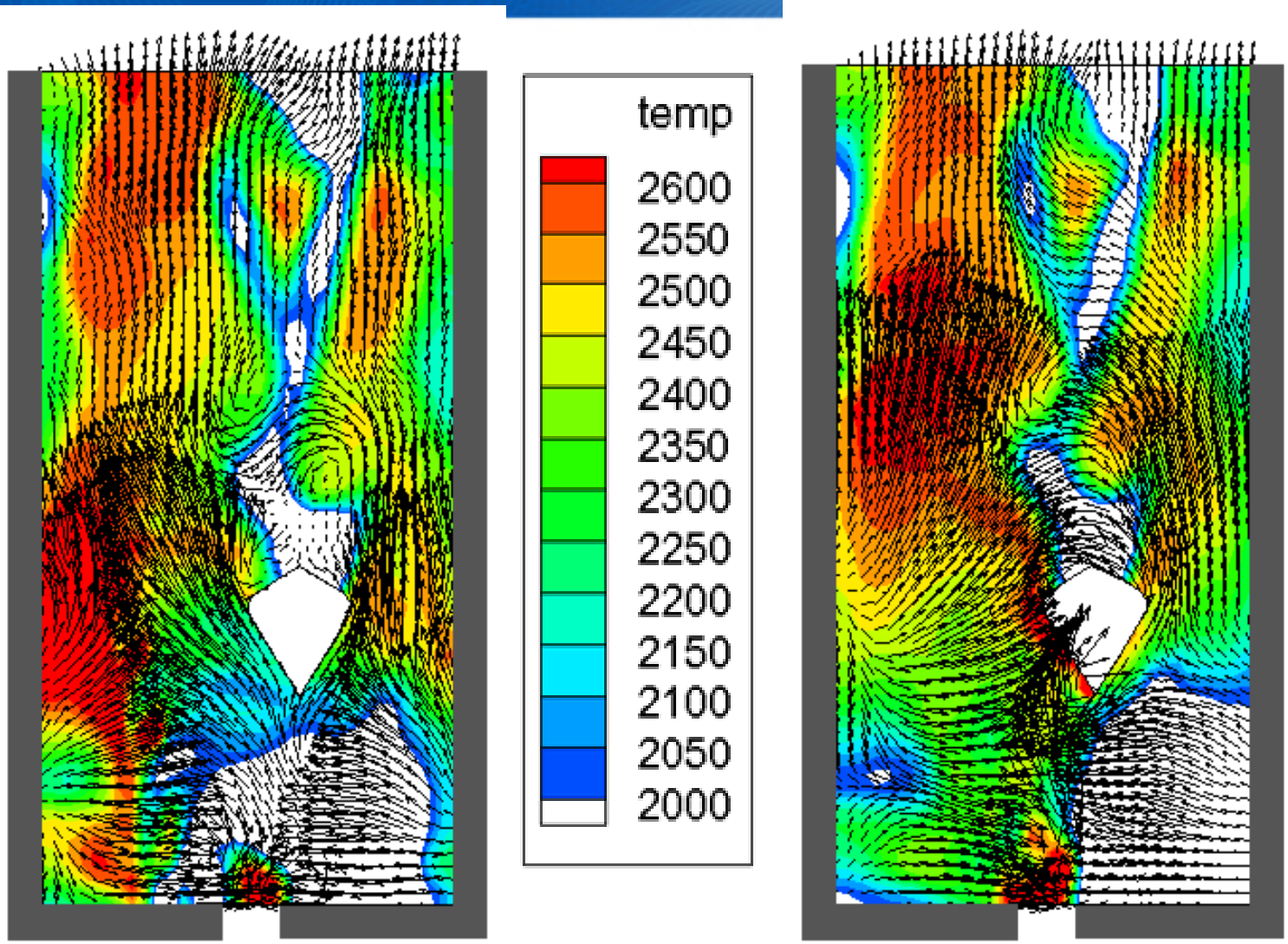






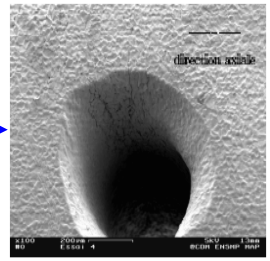


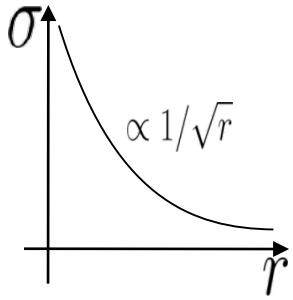
Set-up of the "third detonation wave"



Reflection on the walls of the "third detonation wave"

Structure : Système de refroidissement d'une aube de turbine





Prise en compte précise des trous fondamentale, conditionne l'amorçage des fissures

Problème

Nddl>> 2. 10^6

Conditionnement $\longrightarrow$ $+\infty$

Solution:

Résoudre les échelles séparément

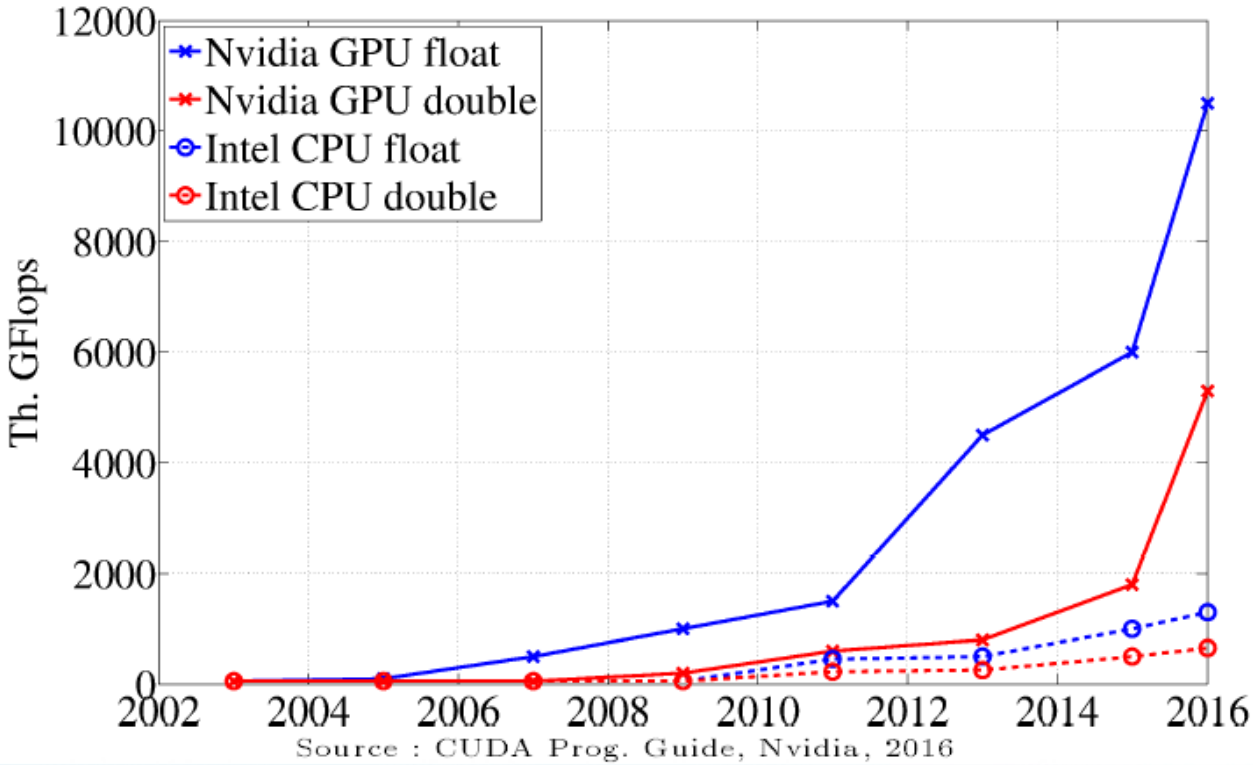
- Parallélisme
- Conditionnement $\longrightarrow$ 1

Patch Raffiné Adaptatif + Décomposition de Domaines

Optimisation de codes pour le calcul scientifique sur carte Graphique

Pourquoi le calcul sur carte graphique ?

1. Plus de transistors dédiés au calcul que les CPU,
2. Conçu pour les problèmes massivement parallèles,
3. Meilleur ratio Performances/Watts.



Source : CUDA Prog. Guide, Nvidia, 2016

ONERA

Optimisation de codes pour le calcul scientifique sur carte Graphique

Ref : Code CPU optimisé (1 thread OpenMP+vectorisation) : 1380 s

Liste des développements GPU

1. Transformation code C $\rightarrow$ GPU : 278 s (gain 396%/396%).
2. Diminution des accès à la mem. glob. : 70 s (gain 297%/1870%).
3. Regroupement calculs de flux+regroupement des intégrales volumiques :
   40 s (gain 75%/3450%).
4. Factorisation et simplification des calculs :
   33 s (gain 21%/4180%).
5. Alignement forcé sur N_x comme multiple de 32 :
   32 s (gain 3%/4210%).

Direct - Iterative solvers

$$A\,x = b$$

Direct:

- Small size full matrix
- Many righthand sides: factorisation made once and for all RHS

Iterative

- Large size sparse matrix
- Many solves for various matrices :

  Implicit scheme for evolution equation

  Use the descent directions for previous solves

Matrice creuse = Sparse

Usual definition: "..matrices that allow special techniques to take advantage of the large number of zero elements and the structure." (J. Wilkinson)

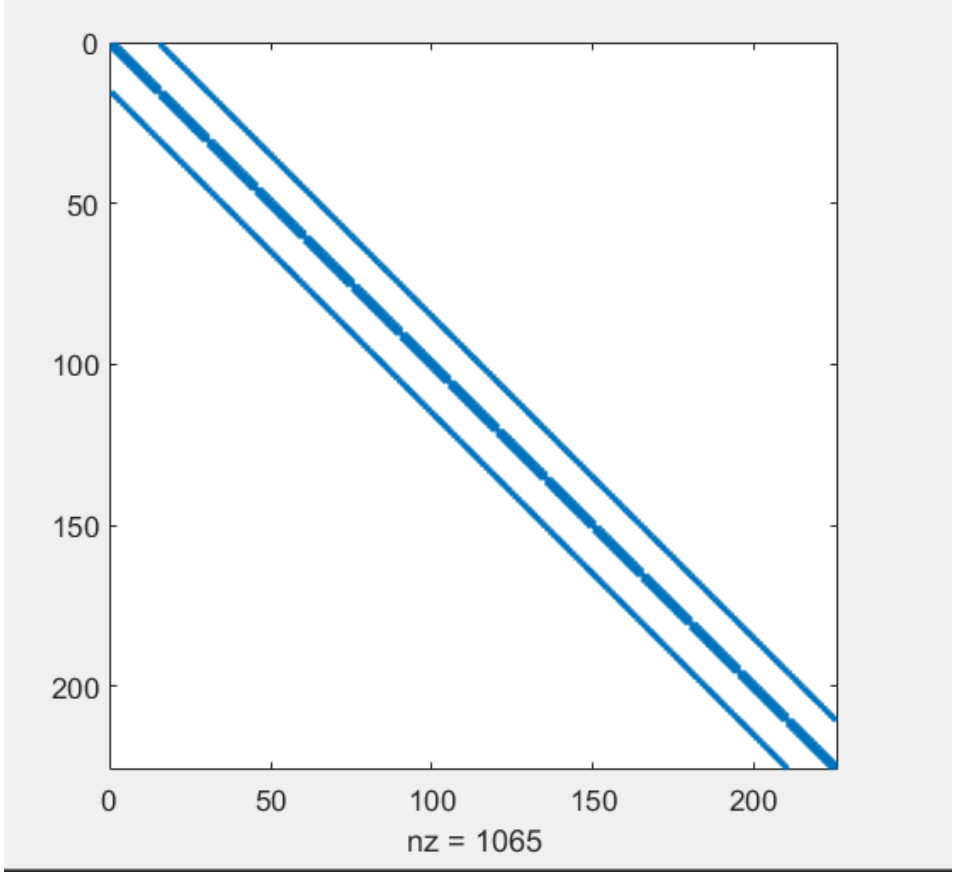
A few applications of sparse matrices: Structural Engineering, Reservoir simulation, Electrical Networks, optimization problems, ...

Goals: Much less storage and work than dense computations.

Observation: A^{-1} is usually dense, but L and U in the LU factorization may be reasonably sparse (if a good technique is used).

Sparse Matrix A

- With Matlab spy(A)



Sparse Matrices

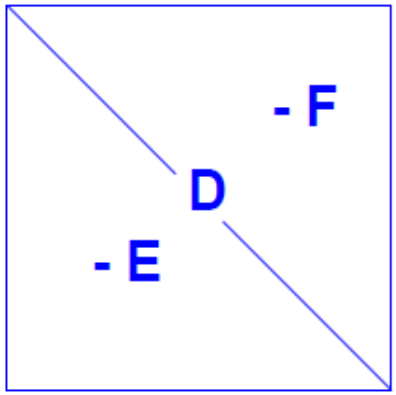
- Main goal of Sparse Matrix Techniques: To perform standard matrix computations economically i.e., without storing the zeros of the matrix.

- Example: To add two square dense matrices of size n requires $O(n^2)$ operations. To add two sparse matrices A and B requires $O(nnz(A) + nnz(B))$ where $nnz(X) =$ number of nonzero elements of a matrix X.

- For typical Finite Element /Finite difference matrices, number of nonzero elements is $O(n)$.

Basic Relaxation schemes

Relaxation schemes: based on the decomposition $A = D - E - F$



D = diag(A), $-E$ = strict lower part of A and $-F$ its strict upper part.

Gauss-Seidel iteration for solving $Ax = b$:

$$(D - E)x^{(k+1)} = Fx^{(k)} + b$$

$\rightarrow$ idea: correct the j-th component of the current approximate solution, $j = 1, 2, ..n$, to zero the $j - th$ component of residual.

Basic Relaxation schemes

Can also define a backward Gauss-Seidel Iteration:

$$(D - F)x^{(k+1)} = Ex^{(k)} + b$$

and a Symmetric Gauss-Seidel Iteration: forward sweep followed by backward sweep.

Over-relaxation is based on the decomposition:

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)D)$$

$\longrightarrow$ successive overrelaxation, (SOR):

$$(D - \omega E)x^{(k+1)} = [\omega F + (1 - \omega)D]x^{(k)} + \omega b$$

Basic Relaxation schemes

Iteration matrices

Jacobi, Gauss-Seidel, SOR, & SSOR iterations are of the form

$$x^{(k+1)} = Mx^{(k)} + f$$

- $M_{Jac} = D^{-1}(E+F) = I - D^{-1}A$
- $M_{GS}(A) = (D-E)^{-1}F == I - (D-E)^{-1}A$
- $M_{SOR}(A) = (D-\omega E)^{-1}(\omega F + (1-\omega)D) = I - (\omega^{-1}D - E)^{-1}A$
- $M_{SSOR}(A) = I - (2\omega^{-1}-1)(\omega^{-1}D - F)^{-1}D(\omega^{-1}D - E)^{-1}A$
  $= I - \omega(2\omega - 1)(D - \omega F)^{-1}D(D - \omega E)^{-1}A$

General convergence result

Consider the iteration: $x^{(k+1)} = Gx^{(k)} + f$

(1) Assume that $\rho(A) < 1$. Then $I - G$ is non-singular and G has a fixed point. Iteration converges to a fixed point for any f and $x^{(0)}$.

(2) If iteration converges for any f and $x^{(0)}$ then $\rho(G) < 1$.

- Jacobi and Gauss-Seidel converge for diagonal dominant A
- SOR converges for $0 < \omega < 2$ for SPD matrices

Projection Methods

Initial Problem: $b - Ax = 0$

Given two subspaces K and L of $\mathbb{R}^N$ de£ne the approximate problem:

Find $\tilde{x} \in K$ such that $b - A\tilde{x} \perp L$

- Leads to a small linear system ('projected problems') This is a basic projection step. Typically: sequence of such steps are applied

- With a nonzero initial guess x_0, the approximate problem is

Find $\tilde{x} \in x_0 + K$ such that $b - A\tilde{x} \perp L$

Write $\tilde{x} = x_0 + \delta$ and $r_0 = b - Ax_0$. Leads to a system for δ:

Find $\delta \in K$ such that $r_0 - A\delta \perp L$

Matrix Representation

Let

- $V = [v_1, \ldots, v_m]$ a basis of K &
- $W = [w_1, \ldots, w_m]$ a basis of L

Then letting x be the approximate solution $\tilde{x} = x_0 + \delta \equiv x_0 + Vy$ where y is a vector of $\mathbb{R}^m$, the Petrov-Galerkin condition yields,

$$W^T(r_0 - AVy) = 0$$

and therefore

$$\tilde{x} = x_0 + V[W^T AV]^{-1} W^T r_0$$

Remark: In practice $W^T AV$ is known from algorithm and has a simple structure [tridiagonal, Hessenberg,..]

One Dimensional projection processes

$$K = span\{d\}$$
and
$$L = span\{e\}$$

Then $\tilde{x} \leftarrow x + \alpha d$ and Petrov-Galerkin condition $r - A\delta \perp e$ yields

$$\alpha = \frac{(r,e)}{(Ad,e)}$$

Three popular choices:

(I) Steepest descent. A is SPD. Take at each step $d = r$ and $e = r$.

Iteration:

$$r \leftarrow b - Ax,$$
$$\alpha \leftarrow (r,r)/(Ar,r)$$
$$x \leftarrow x + \alpha r$$

- Each step minimizes $f(x) = \|x - x^*\|_A^2 = (A(x - x^*), (x - x^*))$

in direction $-\nabla f$. Convergence guaranteed if A is SPD.

Krylov Subspace Methods

Principle: Projection methods on Krylov subspaces, i.e., on

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$$

- probably the most important class of iterative methods.
- many variants exist depending on the subspace L.

A little review: Gram-Schmidt process

→ **Goal: given** $X = [x_1, \ldots, x_m]$ **compute an orthonormal set** $Q = [q_1, \ldots, q_m]$ **which spans the same susbpace.**

ALGORITHM : 1 ▪ ***Classical Gram-Schmidt***

1. ***For*** $j = 1, ..., m$ ***Do:***
2. ***Compute*** $r_{ij} = (x_j, q_i)$ ***for*** $i = 1, \ldots, j-1$
3. ***Compute*** $\hat{q}_j = x_j - \Sigma_{i=1}^{j-1} r_{ij} q_i$
4. $r_{jj} = \|\hat{q}_j\|_2$ ***If*** $r_{jj} == 0$ ***exit***
5. $q_j = \hat{q}_j / r_{jj}$
6. ***EndDo***

Gram-Schmidt process

ALGORITHM : 2 ▪ **Modified Gram-Schmidt**

1. **For** $j = 1, ..., m$ **Do:**
2. $\hat{q}_j := x_j$
3. **For** $i = 1, \ldots, j-1$ **Do**
4. $r_{ij} = (\hat{q}_j, q_i)$
5. $\hat{q}_j := \hat{q}_j - r_{ij} q_i$
6. **EndDo**
7. $r_{jj} = \|\hat{q}_j\|_2$. **If** $r_{jj} == 0$ **exit**
8. $q_j := \hat{q}_j / r_{jj}$
9. **EndDo**

Remark

Let:

$X = [x_1, \ldots, x_m]$ ($n \times m$ matrix)

$Q = [q_1, \ldots, q_m]$ ($n \times m$ matrix)

$R = \{r_{ij}\}$ ($m \times m$ upper triangular matrix)

- At each step,

$$x_j = \sum_{i=1}^{j} r_{ij} q_i$$

Result:

$$X = QR$$

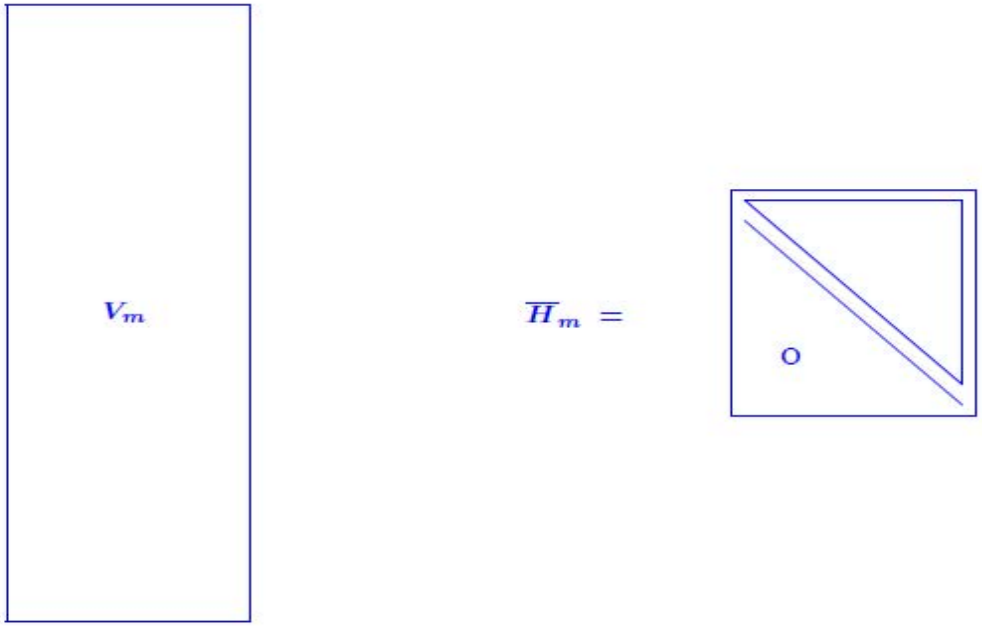
ARNOLDI'S ALGORITHM

- Goal: to compute an orthogonal basis of K_m.
- Input: Initial vector v_1, with $\|v_1\|_2 = 1$ and m.

For $j = 1, ..., m$ do

- Compute $w := Av_j$
- for $i = 1, \ldots, j$, do $\begin{cases} h_{i,j} := (w, v_i) \\ w := w - h_{i,j} v_i \end{cases}$
- $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$

ARNOLDI'S ALGORITHM



Result of orthogonalization process (Arnoldi's algorithm:)

1. $V_m = [v_1, v_2, \ldots, v_m]$ orthonormal basis of K_m.

2. $AV_m = V_{m+1}\overline{H}_m$

3. $V_m^T A V_m = H_m \equiv \overline{H}_m -$ last row.

Arnoldi's Method (Lm = Km)

From Petrov-Galerkin condition when $L_m = K_m$, we get

$$x_m = x_0 + V_m H_m^{-1} V_m^T r_0$$

If, in addition we choose $v_1 = r_0/\|r_0\|_2 \equiv r_0/\beta$ in Arnoldi's algorithm, then

$$x_m = x_0 + \beta V_m H_m^{-1} e_1$$

Several algorithms mathematically equivalent to this approach:

* FOM [Saad, 1981] (above formulation)

* Young and Jea's ORTHORES [1982].

* Axelsson's projection method [1981].

Minimal residual methods (Lm = AKm)

When $L_m = AK_m$, we let $W_m \equiv AV_m$ and obtain relation

$$x_m = x_0 + V_m[W_m^T A V_m]^{-1} W_m^T r_0 = x_0 + V_m[(AV_m)^T AV_m]^{-1}(AV_m)^T r_0.$$

Use again $v_1 := r_0/(\beta := \|r_0\|_2)$ and the relation $\boxed{AV_m = V_{m+1}\bar{H}_m}$:

$$x_m = x_0 + V_m[\bar{H}_m^T \bar{H}_m]^{-1} \bar{H}_m^T \beta e_1 = x_0 + V_m y_m$$

where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ over $y \in \mathbb{R}^m$. Therefore, (Generalized Minimal Residual method (GMRES) [Saad-Schultz, 1983]):

$$x_m = x_0 + V_m y_m \quad \text{where} \quad y_m : \min_y \|\beta e_1 - \bar{H}_m y\|_2$$

Equivalent methods:

- Axelsson's CGLS
- Orthomin (1980)
- Orthodir
- GCR

Some implementation details: GMRES

- Issue 1 : how to solve least-squares problem ?
- Issue 2: How to compute residual norm (without computing solution at each step)?
- Several solutions to both issues. Simplest: use Givens rotations.
- Recall: we want to solve least-squares problem

$$\min_y \|\beta e_1 - \overline{H}_m y\|_2$$

- Transform the problem into upper triangular one.

Some implementation details: GMRES

- Rotation matrices of dimension $m+1$. Define (with $s_i^2 + c_i^2 = 1$):

$$\Omega_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \\ & & & -s_i & c_i & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{row } i \\ \leftarrow \text{row } i+1 \\ \\ \\ \\ \end{matrix}$$

- Multiply $\bar{H}_m$ and right-hand side $\bar{g}_0 \equiv \beta e_1$ by a sequence of such matrices from the left. s_i, c_i selected to eliminate $h_{i+1,i}$

Some implementation details: GMRES

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- 1-st Rotation

$$\Omega_1 = \begin{pmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$$

with

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}, \quad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}}$$

Some implementation details: GMRES

$$\bar{H}_m^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- repeat with $\Omega_2, \ldots, \Omega_i$. Result:

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix}.$$

Some implementation details: GMRES

Define

$$
\begin{aligned}
Q_m &= \Omega_m \Omega_{m-1} \dots \Omega_1 \\
\bar{R}_m &= \bar{H}_m^{(m)} = Q_m \bar{H}_m, \\
\bar{g}_m &= Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T.
\end{aligned}
$$

- Since Q_m is unitary,

$$\min \|\beta e_1 - \bar{H}_m y\|_2 = \min \|\bar{g}_m - \bar{R}_m y\|_2.$$

- Delete last row and solve resulting triangular system.

$$R_m y_m = g_m$$

Some implementation details: GMRES

PROPOSITION:

1. The rank of AV_m is equal to the rank of R_m. In particular, if $r_{mm} = 0$ then A must be singular.

2. The vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ is given by

$$y_m = R_m^{-1} g_m.$$

3. The residual vector at step m satisfies

$$b - Ax_m = V_{m+1}\left(\beta e_1 - \bar{H}_m y_m\right) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$$

and, as a result,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}|.$$

MATRIX MARKET

R.F. Boisvert, R. Pozo, K. Remington, R. Barret, and J.J. Dongarra *The Matrix Market : A web resource for test matrix collections, the Quaity of numerical software, Assessment and Enhancement*, R.F. Boisvert, ed., Chapman Hall, London, 1997, pp. 125-137.

https://math.nist.gov/MatrixMarket/