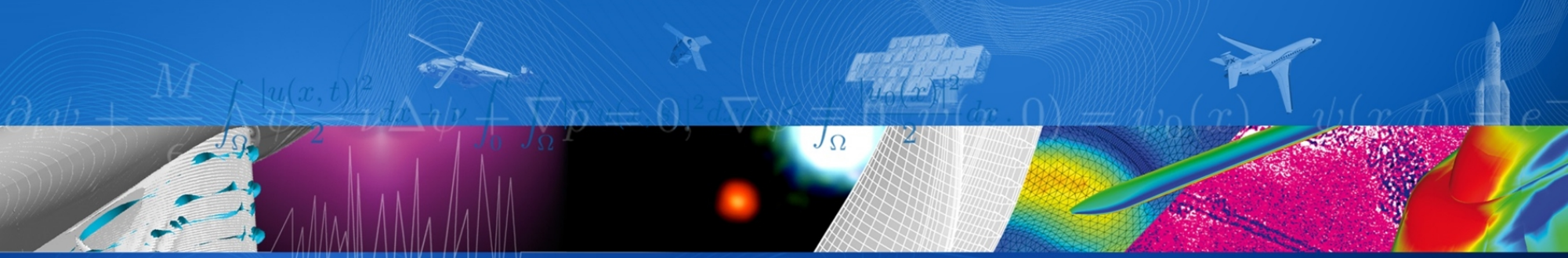


Augmented Lagrangian approach for large-scale linear stability analysis of incompressible flows and its extension to fluid-structure interaction

J. Moulin[1], P. Jolivet[2], O.Marquet[1]

[1] ONERA - Département d'Aérodynamique, Aéroélasticité et Aéroacoustique
[2] CNRS - Institut de Recherche en Informatique de Toulouse

Funded by *ERC Starting Grant*

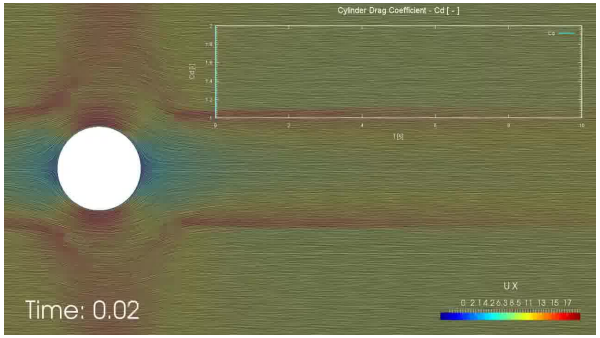
FreeFem++ Days, Paris, 13 December 2018

European Research Council
Established by the European Commission

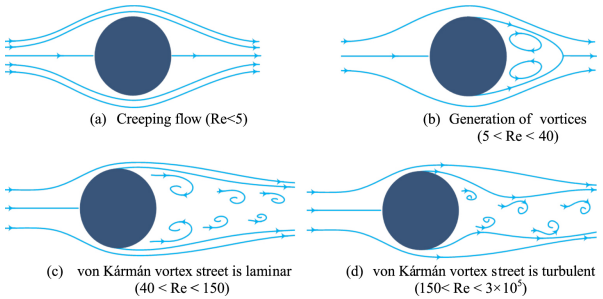
ONERA
THE FRENCH AEROSPACE LAB

Introduction

Some instability phenomena ...



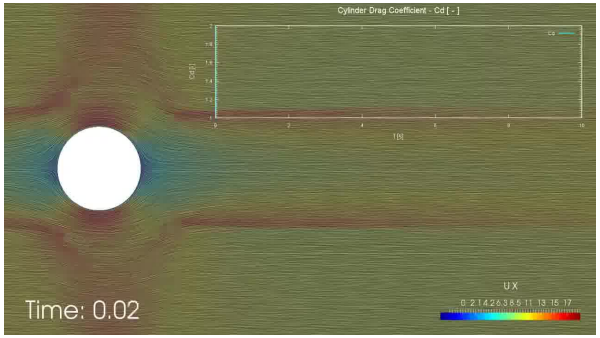
Cylinder flow



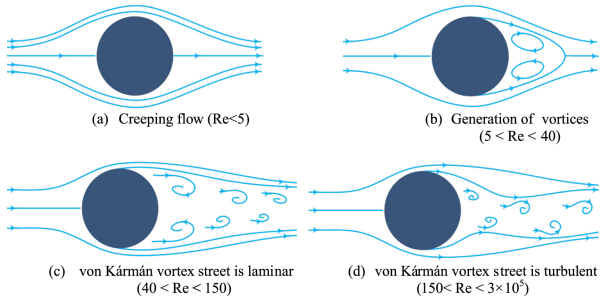
[Goharzadeh et al., *Eur. J. Phys.* 2015]

Introduction

Some instability phenomena ...



Cylinder flow



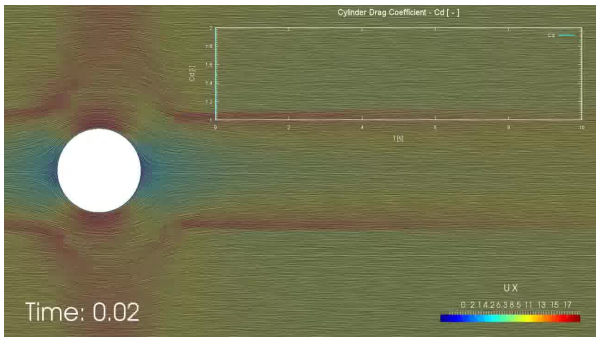
[Goharzadeh et al., *Eur. J. Phys.* 2015]

VIV

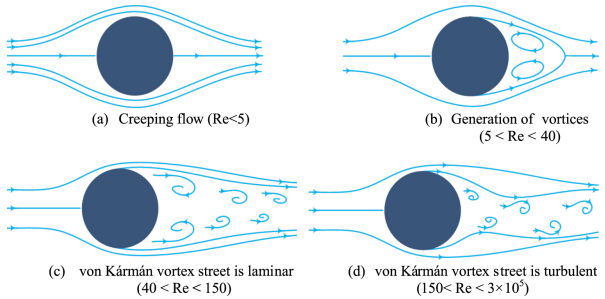
Wing flutter

Introduction

Some instability phenomena ...



Cylinder flow



[Goharzadeh et al., *Eur. J. Phys.* 2015]

VIV

Wing flutter

➥How to predict the occurence of those instabilities ?

Outline

Outline

Linear Stability Analysis

$$\mathbf{M}\frac{\partial \mathbf{q}}{\partial t} + \mathbf{R}(\mathbf{q}) = 0 \tag{1}$$

with $\mathbf{q}$ is the vector of all unknowns.

A steady solution – the *base flow* – verifies:

$$\mathbf{R}(\mathbf{q}_b) = 0 \tag{2}$$

The stability of small perturbations is investigated assuming $\mathbf{q}(\mathbf{x}, t) = \mathbf{q}_b(\mathbf{x}) + \epsilon\,(\hat{\mathbf{q}}(\mathbf{x})e^{\sigma t} + \text{c.c.})$, which leads to the eigenvalue problem:

$$\sigma\,\mathbf{M}\,\hat{\mathbf{q}} + \mathbf{J}(\mathbf{q}_b)\,\hat{\mathbf{q}} = 0 \qquad , \qquad \mathbf{J}(\mathbf{q}_b) = \left.\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right|_{\mathbf{q}_b} \tag{3}$$

The system is linearly unstable if at least one eigenvalue has $\mathfrak{Re}(\sigma) > 0$

Numerical methods

Steady solution:

$$\mathbf{R}(\mathbf{q}_b) = 0$$

Eigenvalue problem:

$$\sigma\,\mathbf{M}\,\hat{\mathbf{q}} + \mathbf{J}(\mathbf{q}_b)\,\hat{\mathbf{q}} = 0$$

Newton method

Given an initial guess $\mathbf{q}_b^0$
for $k = 1 \ldots n$,

$$\delta\mathbf{q}_b^k = -\mathbf{J}(\mathbf{q}_b^{k-1})^{-1}\mathbf{R}(\mathbf{q}_b^{k-1})$$
$$\mathbf{q}_b^k = \mathbf{q}_b^{k-1} + \delta\mathbf{q}_b^k,$$

Krylov–Schur (+ shift-invert) method

Transformed problem:

$$\mu\,\hat{\mathbf{q}} + \left[(\mathbf{J}(\mathbf{q}_b) + s\mathbf{M})^{-1}\,\mathbf{M}\right]\hat{\mathbf{q}} = 0$$
$$\mu = (\sigma - s)^{-1},\ s \in \mathbb{C} \text{ (user-defined)}$$

Need to build the Krylov space:

$$\mathcal{K}_n\left(\left[(\mathbf{J}(\mathbf{q}_b) + s\mathbf{M})^{-1}\,\mathbf{M}\right], \mathbf{x_0}\right)$$

In both cases, one needs to apply repeatedly $(\mathbf{J} + s\mathbf{M})^{-1}$, *i.e.* solve:

$$(\mathbf{J} + s\mathbf{M})\,\mathbf{q} = \mathbf{f} \qquad : \text{ unsteady linearized equations}$$

Linear solver

$$(\mathbf{J} + s\mathbf{M})\,\mathbf{q} = \mathbf{f}$$

Two options:

- Direct solvers (sparse LU decompositions)
  - ✔ robust and particularly efficient for eigenvalue computations
  - ✘ memory intensive and bad parallel performance
- Preconditioned iterative methods (*e.g.* GMRES)
  - ✔ smaller memory requirements and good parallel performance
  - ✘ require a **good** preconditioner to converge in decent cpu time

  The iterative method is applied on the right-preconditioned system:

$$(\mathbf{J} + s\mathbf{M})\,\mathbf{P}^{-1}\tilde{\mathbf{q}} = \mathbf{f}$$

$$\mathbf{P}^{-1} \simeq (\mathbf{J} + s\mathbf{M})^{-1} \text{ and “cheap”}$$

And the solution is retrieved by solving

$$\mathbf{P}\mathbf{q} = \tilde{\mathbf{q}}$$

Outline

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the linear system reads:

$$\underbrace{\begin{pmatrix} \mathbf{u}_b \cdot \nabla \bullet + \bullet \cdot \nabla \mathbf{u}_b - \nu \nabla^2 \bullet + s\,\mathrm{Id} & \nabla \\ \nabla \cdot & 0 \end{pmatrix}}_{\mathbf{J}+s\mathbf{M}} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} \\ f_p \end{pmatrix}$$

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the linear system reads:

$$\underbrace{\begin{pmatrix} \mathbf{J_{uu}} + s\mathbf{M_u} & \mathbf{J}_{pu}{}^{\mathsf{T}} \\ \mathbf{J}_{pu} & 0 \end{pmatrix}}_{\mathbf{J}+s\mathbf{M}} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} \\ f_p \end{pmatrix}$$

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the augmented linear system reads:

$$\begin{pmatrix} \mathbf{J_{uu},\gamma} + s\mathbf{M_u} & \mathbf{J_{pu}}^{\mathsf{T}} \\ \mathbf{J_{pu}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} f_p \\ f_p \end{pmatrix}$$

$$\mathbf{J_{uu},\gamma} = \mathbf{J_{uu}} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} \mathbf{J_{pu}}\,, \qquad \mathbf{W} = \text{some SPD matrix}$$

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the augmented linear system reads:

$$\begin{pmatrix} \mathbf{J_{uu},\gamma} + s\mathbf{M_u} & \mathbf{J_{pu}}^{\mathsf{T}} \\ \mathbf{J_{pu}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} f_p \\ f_p \end{pmatrix}$$

$$\mathbf{J_{uu},\gamma} = \mathbf{J_{uu}} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} \mathbf{J_{pu}}\,, \qquad \mathbf{W} = \text{some SPD matrix}$$

The Augmented Lagrangian preconditioner [Benzi et al., *J. Sci. Comput.* 2006]:

$$\mathbf{P}_{\mathrm{AL}} = \begin{pmatrix} \mathbf{J_{uu},\gamma} + s\mathbf{M_u} & 0 \\ \mathbf{J}_{\mathrm{pf}} & \hat{\mathbf{S}}_p \end{pmatrix}$$

with $\hat{\mathbf{S}}_p^{-1} = -\nu \mathbf{M}_p^{-1} - s\mathbf{L}_p{}^{-1} - \gamma \mathbf{W}^{-1}$

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the augmented linear system reads:

$$\begin{pmatrix} \mathbf{J_{uu},\gamma} + s\mathbf{M_u} & \mathbf{J_{pu}}^{\mathsf{T}} \\ \mathbf{J_{pu}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} f_p \\ f_p \end{pmatrix}$$

$$\mathbf{J_{uu},\gamma} = \mathbf{J_{uu}} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} \mathbf{J_{pu}}\,, \qquad \mathbf{W} = \text{some SPD matrix}$$

The Augmented Lagrangian preconditioner [Benzi et al., *J. Sci. Comput.* 2006]:

$$\mathbf{P}_{\mathsf{AL}} = \begin{pmatrix} \begin{pmatrix} \mathbf{J}_{uu,\gamma} + s\mathbf{M}_u & \mathbf{J}_{uv,\gamma} \\ \mathbf{J}_{vu,\gamma} & \mathbf{J}_{vv,\gamma} + s\mathbf{M}_v \end{pmatrix} & 0 \\ \mathbf{J_{pf}} & \hat{\mathbf{S}}_p \end{pmatrix}$$

with $\hat{\mathbf{S}}_p^{-1} = -\nu \mathbf{M}_p^{-1} - s\mathbf{L}_p{}^{-1} - \gamma \mathbf{W}^{-1}$

AL preconditioning for Navier–Stokes

For the incompressible Navier–Stokes problem the augmented linear system reads:

$$\begin{pmatrix} \mathbf{J_{uu},\gamma} + s\mathbf{M_u} & \mathbf{J_{pu}}^{\mathsf{T}} \\ \mathbf{J_{pu}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f_u} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} f_p \\ f_p \end{pmatrix}$$

$$\mathbf{J_{uu},\gamma} = \mathbf{J_{uu}} + \gamma \mathbf{J_{pu}}^{T} \mathbf{W}^{-1} \mathbf{J_{pu}}\,, \qquad \mathbf{W} = \text{some SPD matrix}$$

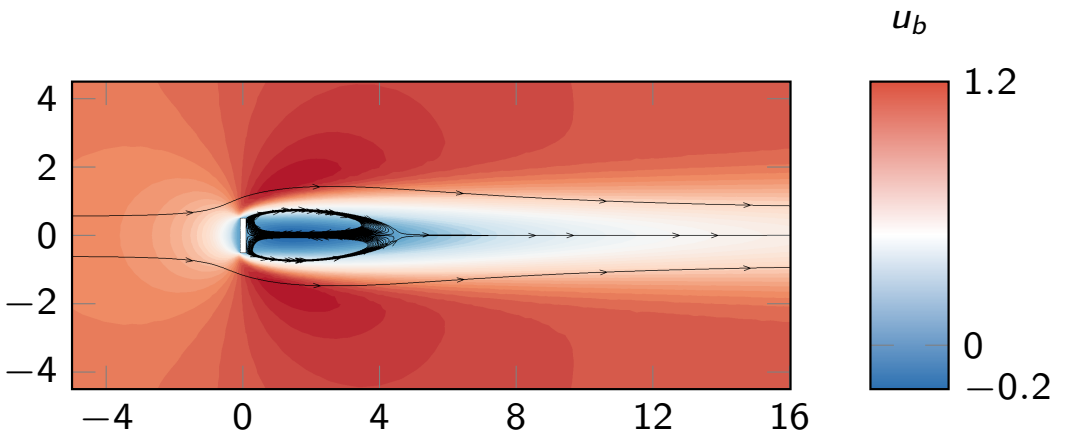
The modified Augmented Lagrangian preconditioner [Benzi et al., *J. Sci. Comput.* 2006]:

$$\mathbf{P}_{\mathsf{AL}} = \begin{pmatrix} \begin{pmatrix} \mathbf{J}_{uu,\gamma} + s\mathbf{M}_u & 0 \\ \mathbf{J}_{vu,\gamma} & \mathbf{J}_{vv,\gamma} + s\mathbf{M}_v \end{pmatrix} & 0 \\ \mathbf{J_{pf}} & \hat{\mathbf{S}}_p \end{pmatrix}$$

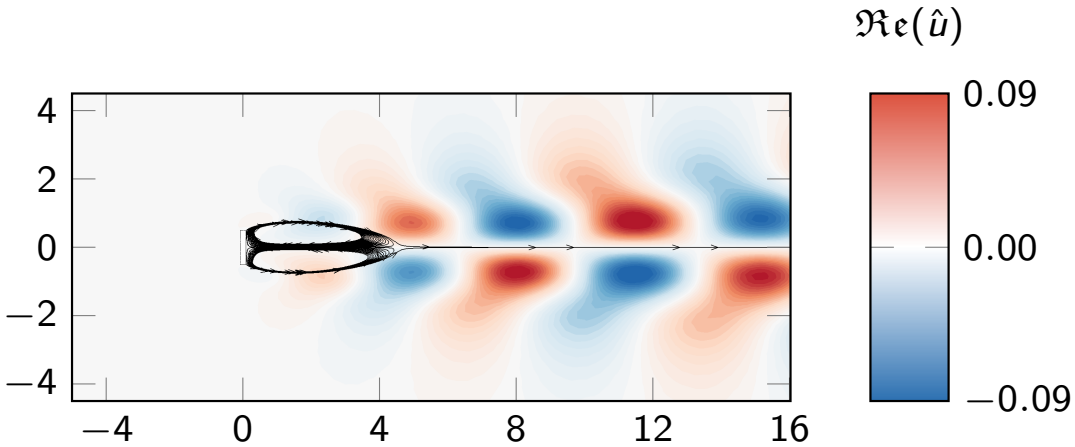
with $\hat{\mathbf{S}}_p^{-1} = -\nu \mathbf{M}_p^{-1} - s\mathbf{L}_p{}^{-1} - \gamma \mathbf{W}^{-1}$

➥Diagonal blocks can be solved with standart preconditioners (*e.g.* algebraic multigrid)

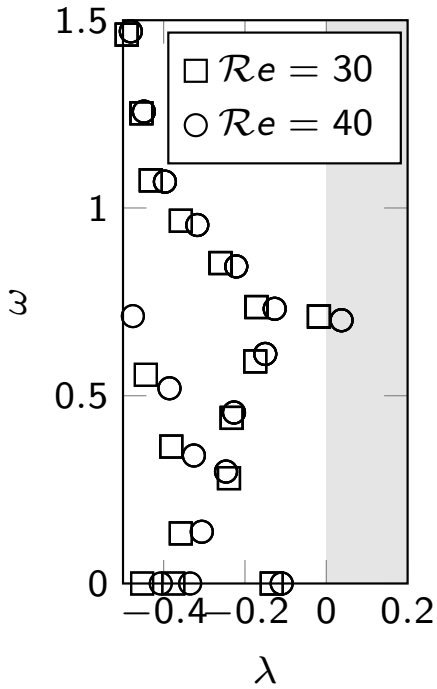
2D test case



(a) Base flow ($\mathcal{R}e = 40$)



(b) Leading eigenmode ($\mathcal{R}e = 40$)



(c) Eigenvalue spectra

Figure: Linear stability analysis of the flow aroud a thick plate

Numerical performance of the AL preconditioner

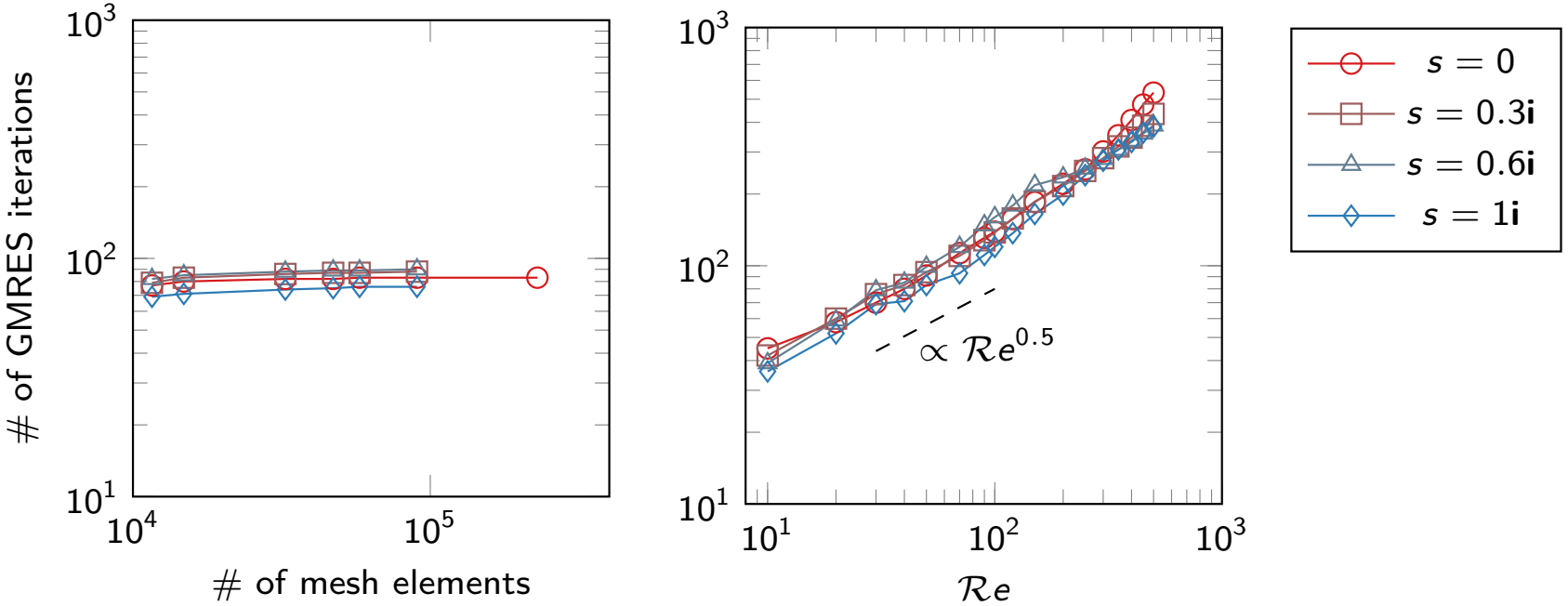


Figure: Effect of mesh refinement and Reynolds number mAL preconditioning efficiency ($tol_{GMRES} = 10^{-6}$).

Numerical performance of the AL preconditioner

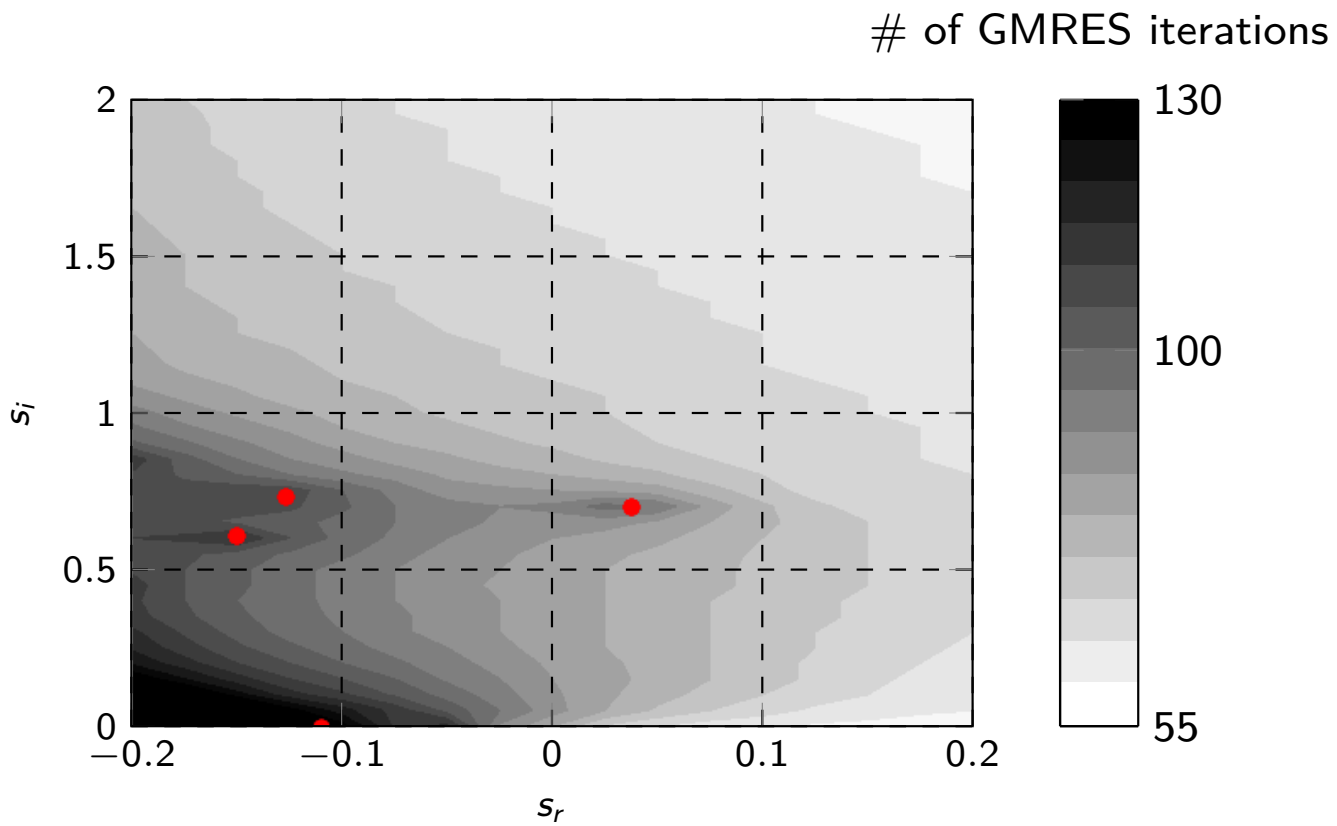


Figure: Influence of the complex shift. ($\mathcal{R}e = 40$ and $tol_{GMRES} = 10^{-6}$). The red circles are the eigenvalues of $\sigma\,\mathbf{M}\,\hat{\mathbf{q}} + \mathbf{J}(\mathbf{q}_b)\,\hat{\mathbf{q}} = 0$

Numerical performance of the AL preconditioner

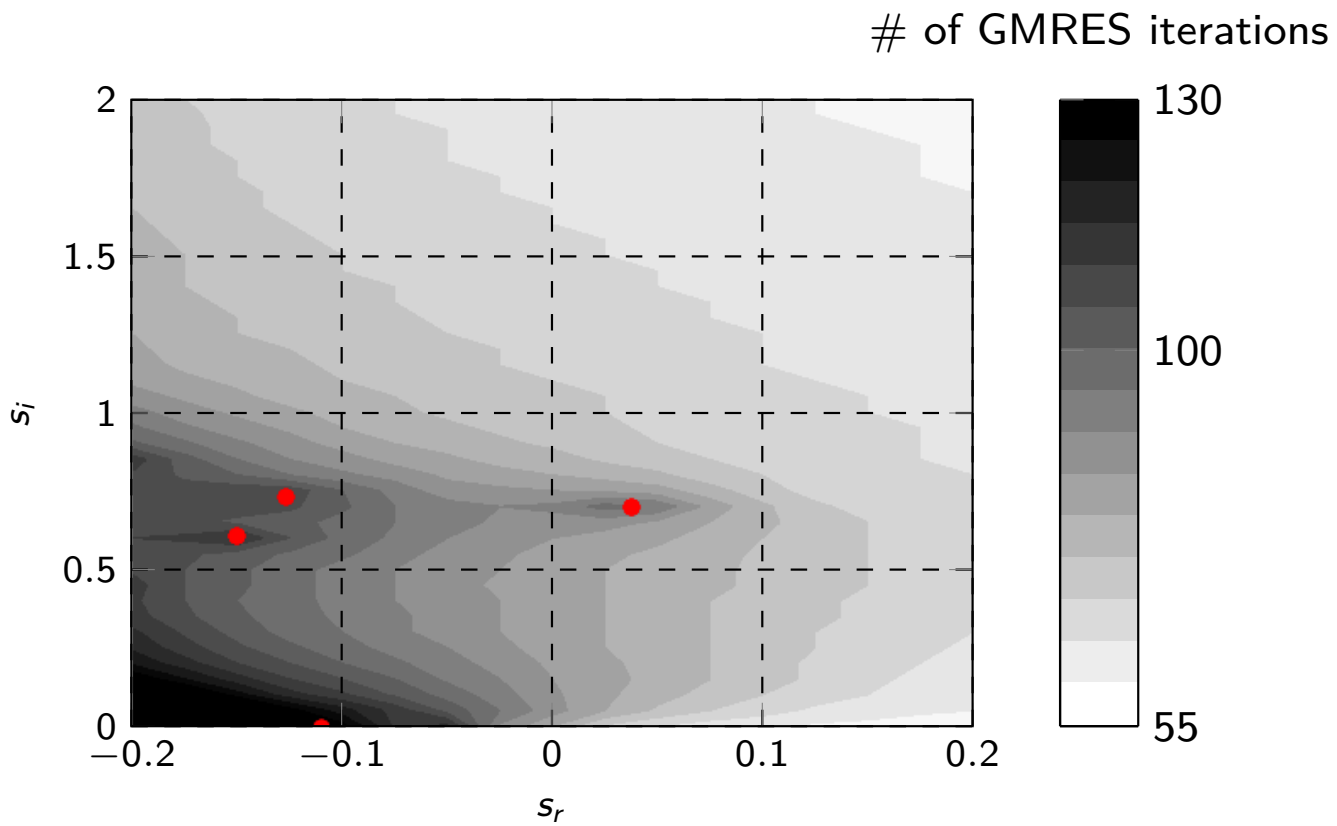


Figure: Influence of the complex shift. ($\mathcal{R}e = 40$ and $tol_{GMRES} = 10^{-6}$). The red circles are the eigenvalues of $\sigma\,\mathbf{M}\,\hat{\mathbf{q}} + \mathbf{J}(\mathbf{q}_b)\,\hat{\mathbf{q}} = 0$

$$s \simeq \sigma \implies \mathbf{J} + s\mathbf{M} \simeq \text{singular} \tag{4}$$

Numerical performance of the AL preconditioner

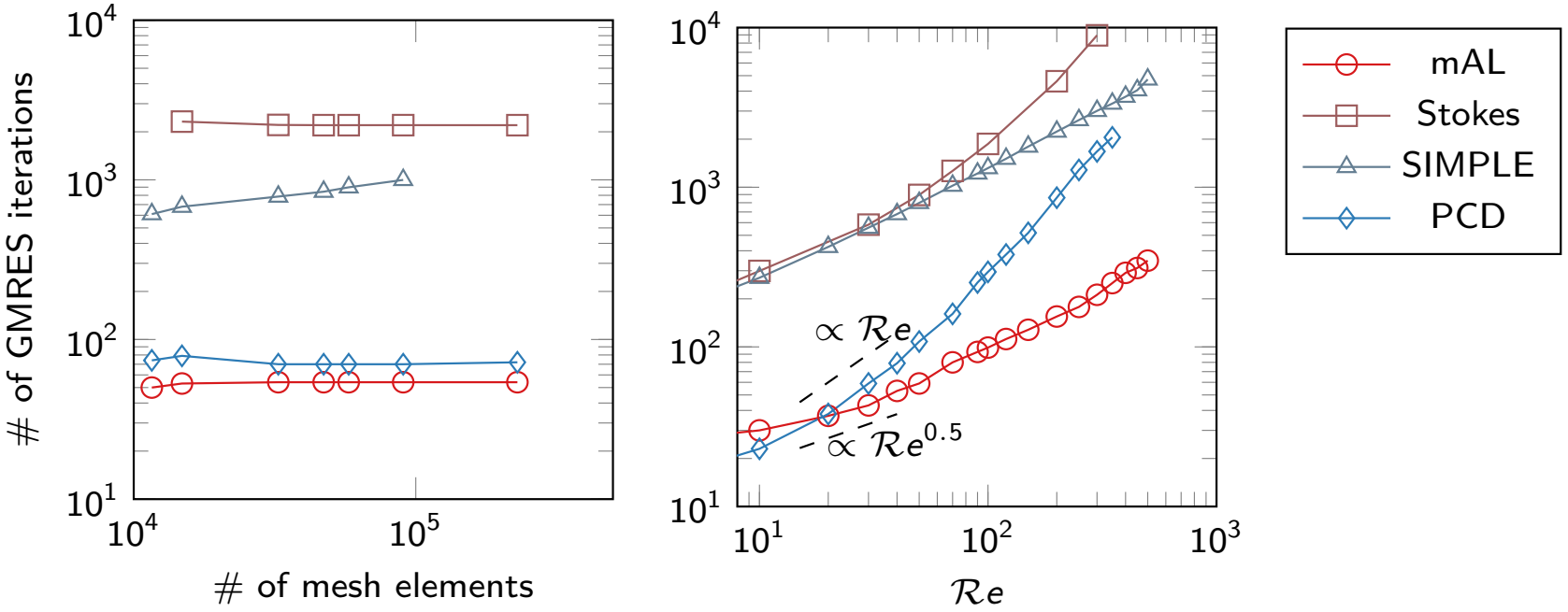


Figure: Influence of the number of mesh elements and the Reynolds number on GMRES iterations with various state-of-the-art preconditioners.

Outline

A FreeFem++/PETSc/SLEPc parallel framework

- FreeFem++
  - Driver
  - Partitioning (Metis)
  - Spatial discretization
- PETSc [http://www.mcs.anl.gov/petsc]
  - Linear solvers (GMRES, BiCGStab, CG, etc)
  - Preconditioners (GAMG, ASM, ILU, etc)
- SLEPc [Hernandez et al., *ACM Trans. Math. Software* 2005]
  - Eigensolvers (Arnoldi, Krylov–Schur, etc)

Some elements of code

```
fespace Wh(th,[P2,P2,P2,P1])//velocity-pressure space
```

```
set(dJ, sparams = params, fields = vX[], names = names, schurPreconditioner
    = S, schurList = listX[]);
```

Some elements of code

```
fespace Wh(th,[P2,P2,P2,P1])//velocity-pressure space
```

```
set(dJ, sparams = params, fields = vX[], names = names, schurPreconditioner
      = S, schurList = listX[]);
```

with:

```
string params = " -ksp_type fgmres -pc_type fieldsplit"
+" -pc_fieldsplit_type multiplicative" // lower block-triangular structure
+ paramsXYZ + // def. velocity sub-solvers
+ paramsP;   // def. pressure sub-solver
```

Some elements of code

```
fespace Wh(th,[P2,P2,P2,P1])//velocity-pressure space
```

```
set(dJ, sparams = params, fields = vX[], names = names, schurPreconditioner
      = S, schurList = listX[]);
```

with:

```
string params = " -ksp_type fgmres -pc_type fieldsplit"
+" -pc_fieldsplit_type multiplicative" // lower block-triangular structure
+ paramsXYZ + // def. velocity sub-solvers
+ paramsP;   // def. pressure sub-solver
```

```
Wh [vX, vY, vZ, p] = [1, 2, 3, 4]; // numbering of each field
string[int] names(4); // prefix of each field
names[0] = "vX"; // x-velocity
names[1] = "vY"; // y-velocity
names[2] = "vZ"; // z-velocity
names[3] = "p"; // pressure
```

Some elements of code

```
fespace Wh(th,[P2,P2,P2,P1])//velocity-pressure space
```

```
set(dJ, sparams = params, fields = vX[], names = names, schurPreconditioner ↵
      ↳ = S, schurList = listX[]);
```

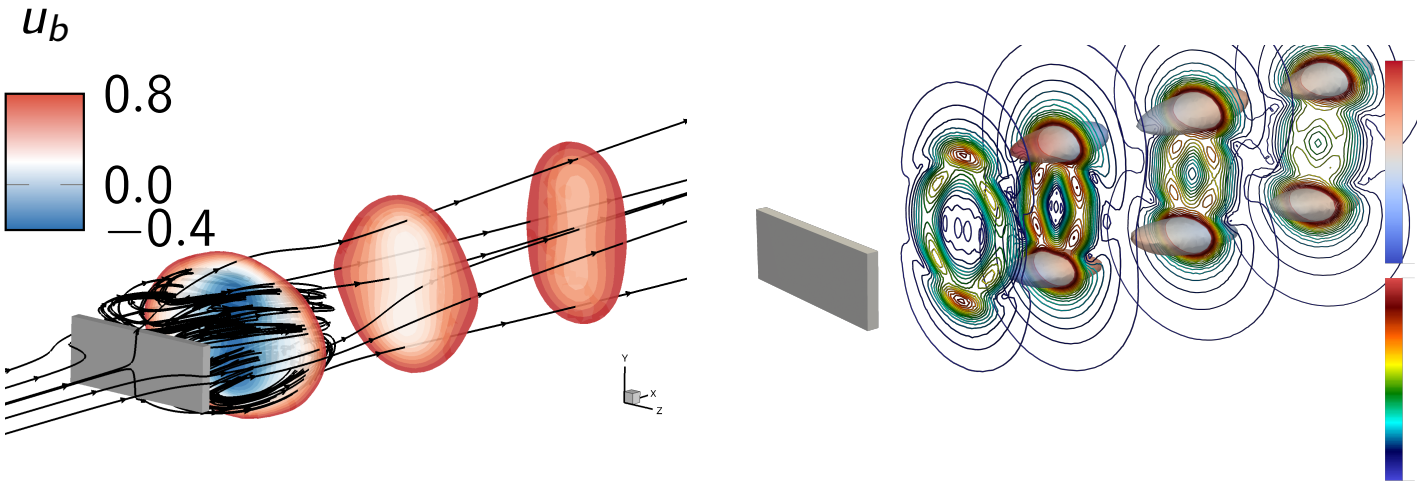
with:

```
string params = " -ksp_type fgmres -pc_type fieldsplit"
+" -pc_fieldsplit_type multiplicative" // lower block-triangular structure
+ paramsXYZ + // def. velocity sub-solvers
+ paramsP;   // def. pressure sub-solver
```

```
Wh [vX, vY, vZ, p] = [1, 2, 3, 4]; // numbering of each field
string[int] names(4); // prefix of each field
names[0] = "vX"; // x-velocity
names[1] = "vY"; // y-velocity
names[2] = "vZ"; // z-velocity
names[3] = "p"; // pressure
```

```
fespace Qh(th, P1);
matrix[int] S(1); // array with a single matrix
varf vSchur(p, q) = int3d(th, qforder = 3) (-1.0/(gamma + 1.0/Re) * p * q); ↵
      ↳ // Ŝ_p
S[0] = vSchur(Qh, Qh); // matrix assembly
```

3D test case



(a) Steady solution (b) Unstable eigenmode

Figure: Linear stability analysis for the 3D flow around a thick plate at $\mathcal{R}e = 100$ (75 million unknowns)

Parallel performance (fluid only)

P	Setup (s)	Solve (s)	Speedup
256	364.6	5,739.9	—
512	88.9	2,994.9	2.0
1,024	56.6	1,523.6	3.9
2,048	13.9	895.1	6.7

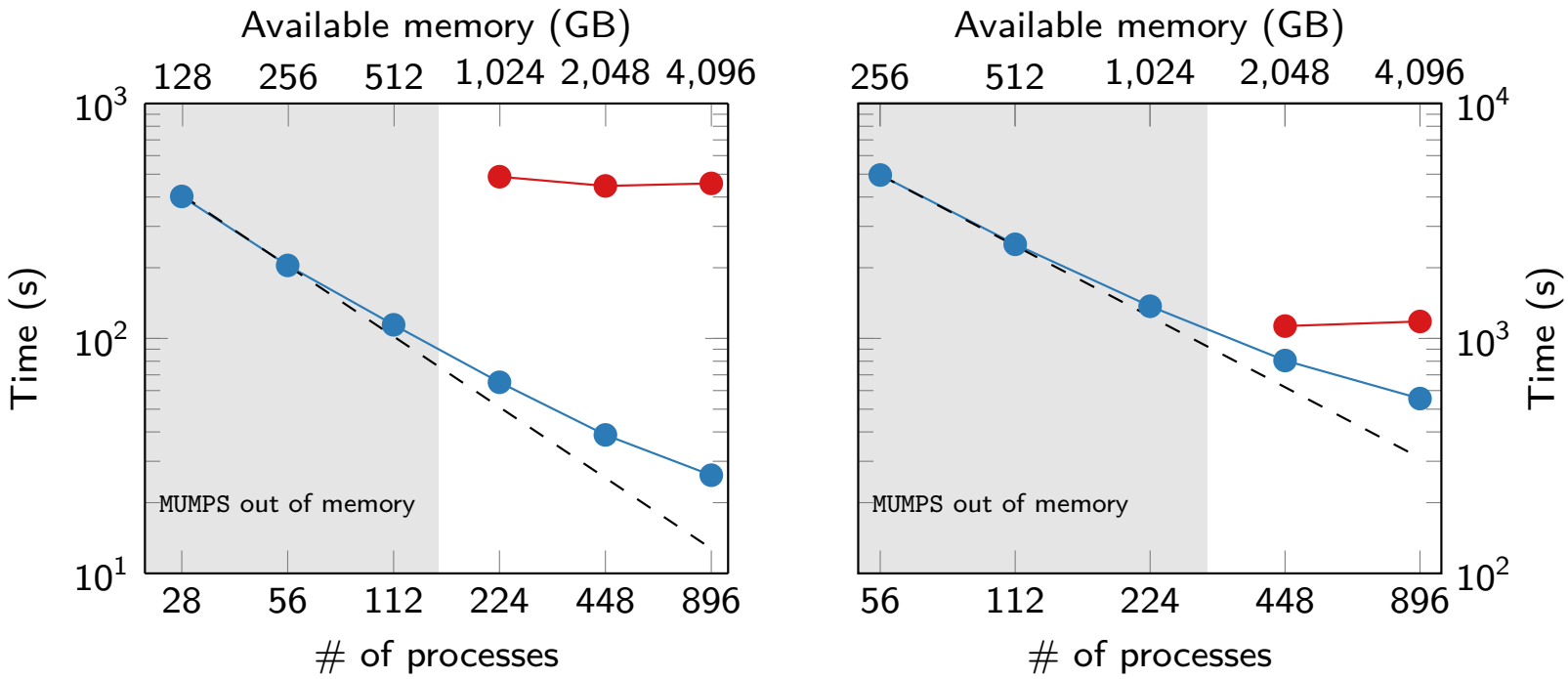
(a) Base flow solver

P	Setup (s)	Solve (s)	Speedup
512	55.3	39,160.7	—
1,024	25.7	24,508.3	1.6
2,048	27.1	11,849.9	3.3

(b) Eigensolver

Table: Scalability of the parallel fluid solvers (70 millions d.o.f., on Curie)

Parallel performance (fluid only)



(a) Base flow solver (average time per Newton iteration)

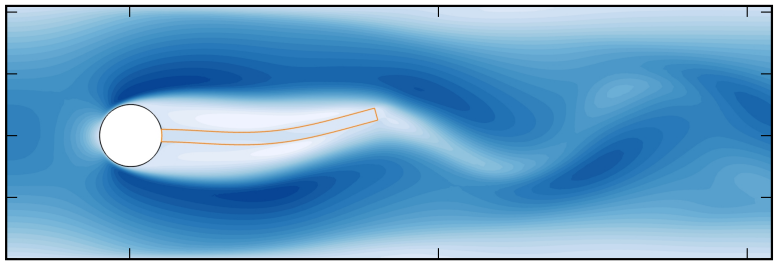
(b) Eigensolver (overall time for 5 eigenvalues, $s = 0.6\mathbf{i}$)

Figure: Comparison iterative (mAL-GMRES, —●—) vs direct solver (MUMPS, —●—) (4.8 million d.o.f.)

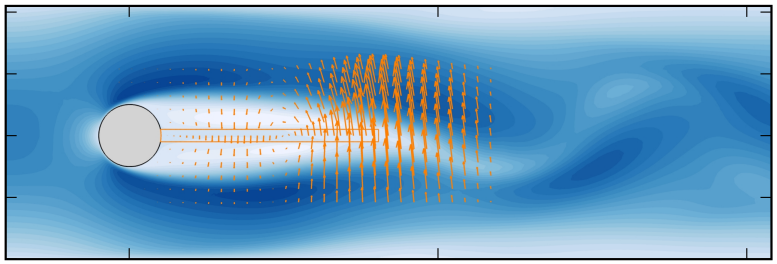
Outline

ALE formulation for moving domains

To handle the moving domain : Arbitrary Lagrangian Eulerian formulation [Donea et al., *Enc. Comp. Mech.* 2004]



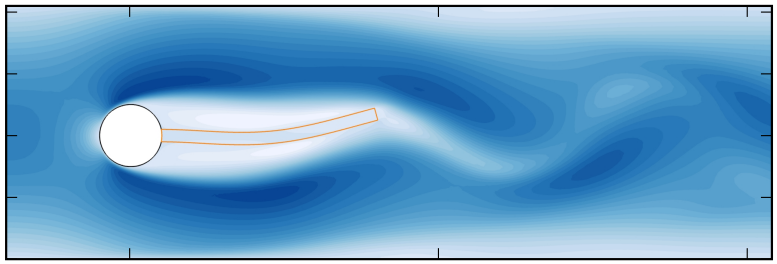
actual domain Ω_t (time-dependent)



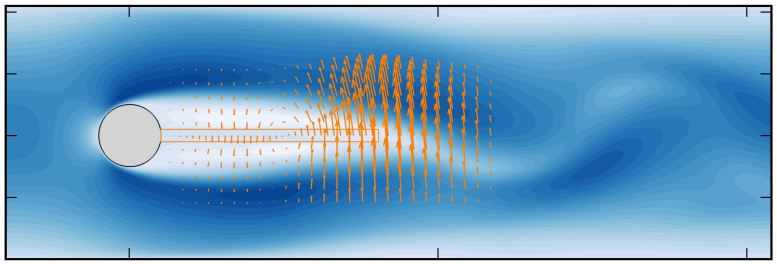
reference domain Ω (fixed)

ALE formulation for moving domains

To handle the moving domain : Arbitrary Lagrangian Eulerian formulation [Donea et al., *Enc. Comp. Mech.* 2004]



actual domain Ω_t (time-dependent)



reference domain Ω (fixed)

Coupled problem: $\mathbf{M}\dfrac{\partial \mathbf{q}}{\partial t} + \mathbf{R}(\mathbf{q}) = 0$ in a **fixed** domain Ω

$$\mathbf{q} = \begin{cases} \text{fluid variables (velocity+pressure)} \\ \text{solid variables (displacement + velocity)} \\ \text{fluid mesh variables (displacement)} \end{cases} \qquad \mathbf{M}, \mathbf{R} = \begin{cases} \text{Navier-Stokes equations} \\ \text{Elasticity equations (solid)} \\ \text{Stress/velocity continuity (interface)} \\ \text{Fluid mesh equation} \end{cases}$$

FSI Jacobian matrix

$$
\left(\begin{array}{c|cc|cc|c}
\mathbf{J}_{ss} & 0 & 0 & 0 & 0 & -\mathbf{M}_{is,1}{}^{T} \\
\hline
0 & \mathbf{J}_{ee} & \mathbf{M}_{ie}{}^{T} & 0 & 0 & 0 \\
-\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\
\hline
0 & \mathbf{J}_{fe} & 0 & \mathbf{J}_{ff} & \mathbf{J}_{pf}{}^{T} & \mathbf{M}_{if}{}^{T} \\
0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\
-\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0
\end{array}\right)
\left(\begin{array}{c}
\mathbf{q}_s \\ \mathbf{q}_e \\ \boldsymbol{\lambda} \\ \mathbf{u}_f \\ p \\ \phi
\end{array}\right)
=
\left(\begin{array}{c}
\mathbf{f}_s \\ \mathbf{f}_e \\ \mathbf{f}_{\boldsymbol{\lambda}} \\ \mathbf{f}_{\mathbf{u}_f} \\ f_p \\ \mathbf{f}_{\phi}
\end{array}\right)
$$

Three Lagrange multiplier variables : $p, \boldsymbol{\lambda}, \phi$

Block preconditioning for the FSI Jacobian

A block Gauss-Seidel approach [Deparis et al., *J. Comput. Phys.* 2016]:

$$\mathbf{J} = \left(\begin{array}{c|cc|cc|c} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & -\mathbf{M}_{is,1}^{T} \\ \hline 0 & \mathbf{J}_{ee} & \mathbf{M}_{ie}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{J}_{fe} & 0 & \mathbf{J}_{ff} & \mathbf{J}_{pf}^{T} & \mathbf{M}_{if}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

Block preconditioning for the FSI Jacobian

A block Gauss-Seidel approach [Deparis et al., *J. Comput. Phys.* 2016]:

$$\mathbf{P}_{FSI} = \left(\begin{array}{c|c|cc|cc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{J}_{ee} & \mathbf{M}_{ie}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{J}_{fe} & 0 & \mathbf{J}_{ff} & \mathbf{J}_{pf}^{T} & \mathbf{M}_{if}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

Block preconditioning for the FSI Jacobian

A block Gauss-Seidel approach [Deparis et al., *J. Comput. Phys.* 2016]:

$$\mathbf{P}_{FSI} = \left(\begin{array}{c|c|cc|cc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{J}_{ee} & \mathbf{M}_{ie}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{J}_{fe} & 0 & \mathbf{J}_{ff} & \mathbf{J}_{pf}^{T} & \mathbf{M}_{if}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

How to handle the 3 sub-problems (solid, extension, fluid) ?

AL preconditioning for the FSI Jacobian

Augmented FSI Jacobian:

$$\left(\begin{array}{c|c|c|ccc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & -\mathbf{M}_{is,1}{}^{T} \\ \hline \mathbf{J}_{es}(\gamma_\lambda) & \mathbf{J}_{ee}(\gamma_\lambda) & \mathbf{M}_{ie}{}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline \mathbf{J}_{fs}(\gamma_\phi) & \mathbf{J}_{fe}(\gamma_p) & 0 & \mathbf{J}_{ff}(\gamma_p,\gamma_\phi) & \mathbf{J}_{pf}{}^{T} & \mathbf{M}_{if}{}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

AL preconditioning for the FSI Jacobian

Augmented FSI Jacobian:

$$\left(\begin{array}{c|cc|ccc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & -\mathbf{M}_{is,1}^{T} \\ \hline \mathbf{J}_{es}(\gamma_\lambda) & \mathbf{J}_{ee}(\gamma_\lambda) & \mathbf{M}_{ie}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline \mathbf{J}_{fs}(\gamma_\phi) & \mathbf{J}_{fe}(\gamma_p) & 0 & \mathbf{J}_{ff}(\gamma_p,\gamma_\phi) & \mathbf{J}_{pf}^{T} & \mathbf{M}_{if}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

FSI Augmented Lagrangian preconditioner:

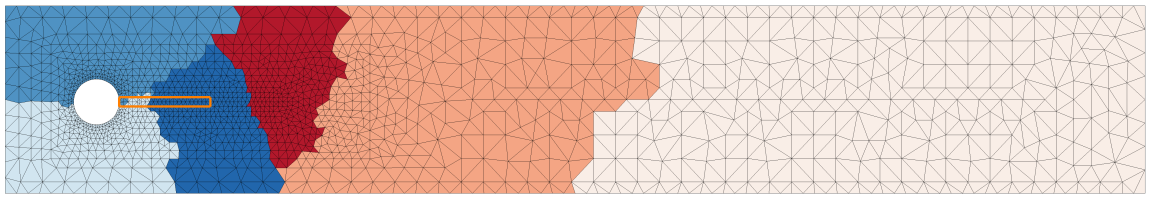
$$\mathbf{P}_{AL,FSI} = \left(\begin{array}{c|cc|ccc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & 0 \\ \hline \mathbf{J}_{es}(\gamma_\lambda) & \mathbf{J}_{ee}(\gamma_\lambda) & 0 & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & \hat{\mathbf{S}}_\lambda & 0 & 0 & 0 \\ \hline \mathbf{J}_{fs}(\gamma_\phi) & \mathbf{J}_{fe}(\gamma_p) & 0 & \mathbf{J}_{ff}(\gamma_p,\gamma_\phi) & 0 & 0 \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & \hat{\mathbf{S}}_p & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & \hat{\mathbf{S}}_\phi \end{array}\right)$$

With $\hat{\mathbf{S}}_\lambda^{-1} = -\gamma_\lambda \mathbf{W}_\lambda^{-1}$ $\quad \hat{\mathbf{S}}_\phi^{-1} = -\gamma_\phi \mathbf{W}_\phi^{-1}$ $\quad \hat{\mathbf{S}}_p^{-1} = -\mathcal{R}e^{-1}\mathbf{M}_p^{-1} - \gamma_p \mathbf{W}_p^{-1}$

Towards a parallel FSI solver

Lots of complications w.r.t. fluid solver:

- Partitioning strategy : the full mesh (fluid + solid) is partitioned

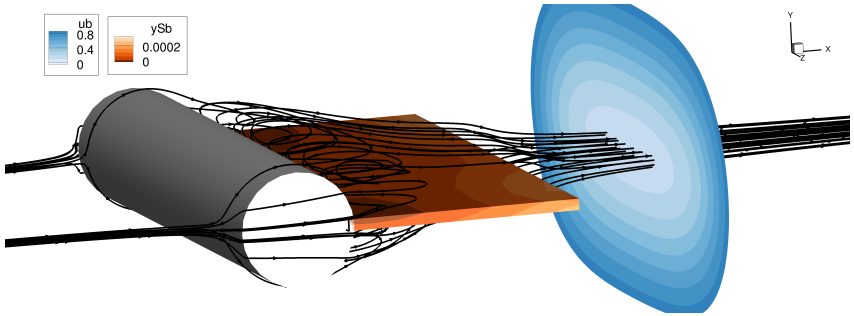


- all unknowns (and `fespace`) are defined on the full mesh
  - ➥ restrictions are needed to keep only relevant d.o.f.:

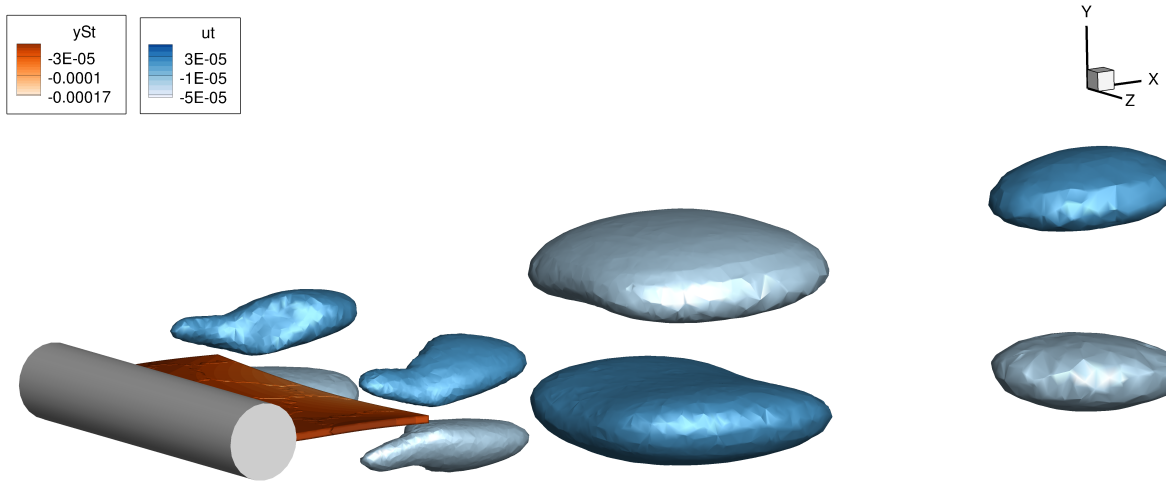
```
dmatrix dJss(Jss, arrayIntersection, ↴
  restrictionIntersection, D, restriction = Rs);
```

- the different unknown belong to different `fespace`
  - ➥ rectangular `dmatrix`: `dmatrix dMis(dJii,dJss,Mis);`
  - ➥ block `dmatrix`: `dmatrix dNest=[[dJss, dMis'],[dMis,0]]`

Preliminary FSI stability results



(a) Steady solution



(b) Unstable eigenmode

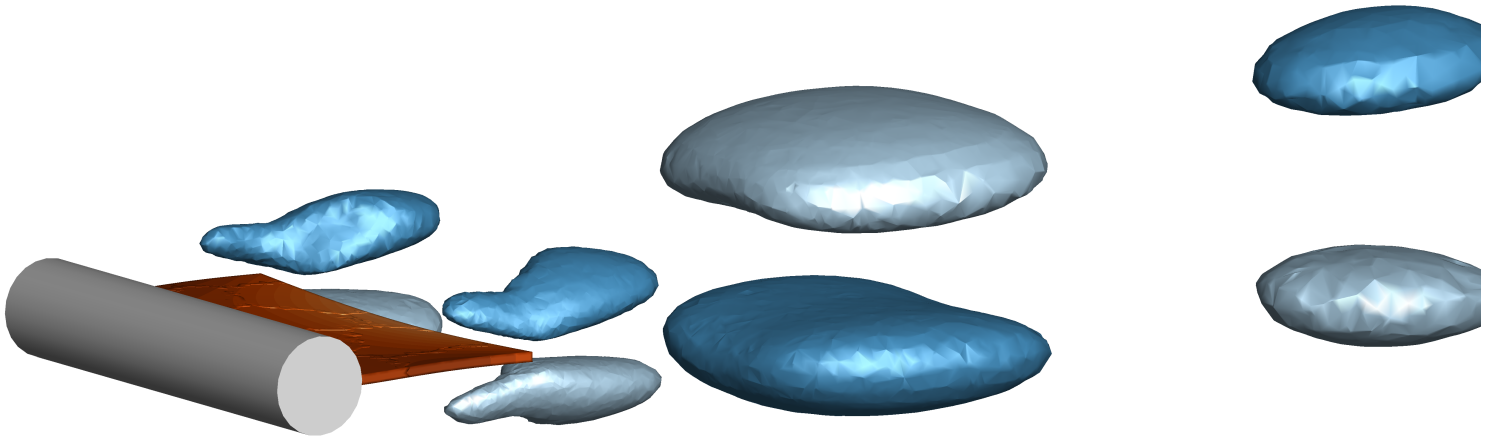
Figure: Linear stability analysis for the 3D flow around a flexible splitter plate

Outline

Conclusions

- The linear systems which arise in the linear stability analysis of incompressible flows are solved using a GMRES algorithm preconditioned by the modified Augmented Lagrangian.
- This strategy was implemented in FreeFem++ using the distributed linear algebra backends PETSc/SLEPc.
- This implementation is able to handle large-scale flow configurations ($\simeq 10^8$ d.o.f.) with satisfying scaling properties.
  - ➥ [Moulin et al., *Comput. Method. Appl. M.*, 2019 (submitted)]
- An extension of the Augmented Lagrangian preconditioner to coupled fluid-structure systems is proposed.
- Numerous tools have been developped in order to implement this preconditioner in parallel in FreeFem++/PETSc/SLEPc
  - ➥ validation in progress ...

Questions ?



Block-preconditioner for saddle-point problems

Augmented FSI Jacobian:

$$\left(\begin{array}{c|cc|ccc} \mathbf{J}_{ss} & 0 & 0 & 0 & 0 & -\mathbf{M}_{is,1}{}^{T} \\ \hline \mathbf{J}_{es}(\gamma_\lambda) & \mathbf{J}_{ee}(\gamma_\lambda) & \mathbf{M}_{ie}{}^{T} & 0 & 0 & 0 \\ -\mathbf{M}_{is,1} & \mathbf{M}_{ie} & 0 & 0 & 0 & 0 \\ \hline \mathbf{J}_{fs}(\gamma_\phi) & \mathbf{J}_{fe}(\gamma_p) & 0 & \mathbf{J}_{ff}(\gamma_p,\gamma_\phi) & \mathbf{J}_{pf}{}^{T} & \mathbf{M}_{if}{}^{T} \\ 0 & \mathbf{J}_{pe} & 0 & \mathbf{J}_{pf} & 0 & 0 \\ -\mathbf{M}_{is,2} & 0 & 0 & \mathbf{M}_{if} & 0 & 0 \end{array}\right)$$

$(2) \leftarrow (2) + \gamma_\lambda \mathbf{M}_{ie}{}^{T}\mathbf{W}_\lambda^{-1} \times (3)$:

$$\begin{cases} \mathbf{J}_{es}(\gamma_\lambda) & = \mathbf{0} - \gamma_\lambda \mathbf{M}_{ie}{}^{T}\mathbf{W}_\lambda^{-1}\mathbf{M}_{is,1} \\ \mathbf{J}_{ee}(\gamma_\lambda) & = \mathbf{J}_{ee} + \gamma_\lambda \mathbf{M}_{ie}{}^{T}\mathbf{W}_\lambda^{-1}\mathbf{M}_{ie} \end{cases}$$

$(4) \leftarrow (4) + \gamma_p \mathbf{J}_{pf}{}^{T}\mathbf{W}_p^{-1} \times (5) + \gamma_\phi \mathbf{M}_{if}{}^{T}\mathbf{W}_\phi^{-1} \times (6)$:

$$\begin{cases} \mathbf{J}_{fs}(\gamma_\phi) & = \mathbf{0} - \gamma_\phi \mathbf{M}_{if}{}^{T}\mathbf{W}_\phi^{-1}\mathbf{M}_{is,2} \\ \mathbf{J}_{fe}(\gamma_p) & = \mathbf{J}_{fe} + \gamma_p \mathbf{J}_{pf}{}^{T}\mathbf{W}_p^{-1}\mathbf{J}_{pe} \\ \mathbf{J}_{ff}(\gamma_p,\gamma_\phi) & = \mathbf{J}_{ff} + \gamma_p \mathbf{J}_{pf}{}^{T}\mathbf{W}_p^{-1}\mathbf{J}_{pf} + \gamma_\phi \mathbf{M}_{if}{}^{T}\mathbf{W}_\phi^{-1}\mathbf{M}_{if} \end{cases}$$