Augmented Lagrangian Preconditioner for Linear Stability Analysis of incompressible fluid flows on large configurations

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

[1] Office National d'Etudes et de Recherches Aérospatiales

[2] ENSEEIHT – Institut de Recherche en Informatique de Toulouse

Funded by *ERC Starting Grant*

FreeFem++ Days, Paris, 14-15 december 2017

European Research Council

Established by the European Commission

THE FRENCH AEROSPACE LAB

What is Linear Stability Analysis ?

One wants to know if some steady solution of equation (1) is *temporally* stable or unstable :

$$\frac{\partial \boldsymbol{q}}{\partial t} = \boldsymbol{\mathcal{R}}(\boldsymbol{q}) \qquad (1)$$

Method : Linear Stability Analysis

Step 1 : compute a steady solution $\boldsymbol{\mathcal{R}}(\boldsymbol{q_b}) = \boldsymbol{0}$

Step 2 : test its stability for small monochromatic perturbations $\hat{\boldsymbol{q}}(x)e^{\sigma t}$ around the steady solution $\boldsymbol{q_b}$

$$\sigma M\hat{\boldsymbol{q}} = J(\boldsymbol{q_b})\hat{\boldsymbol{q}}$$

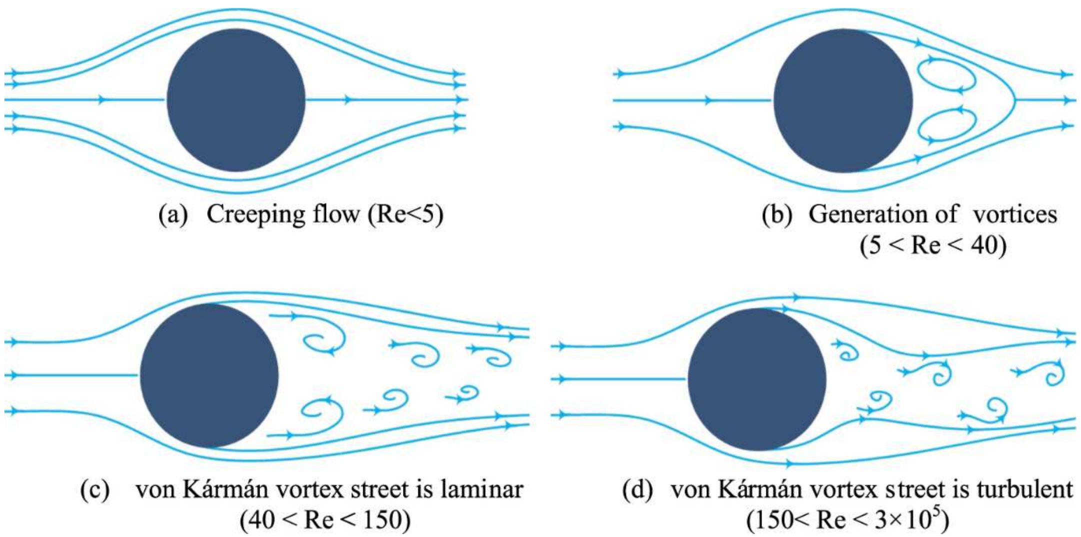
Mass matrix (spatial discretization)

Jacobian matrix : $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{q}}(\boldsymbol{q_b})$

Growth rate $\lambda = \Re[\sigma]$

Frequency $\omega = \Im[\sigma]$

A classical example

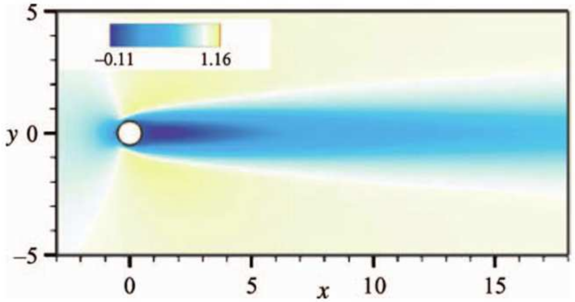


From [Goharzadeh & Molki, 2015]

Typical question :
What is the critical Reynolds number
above which the von Karman vortex street appears ?

A classical example

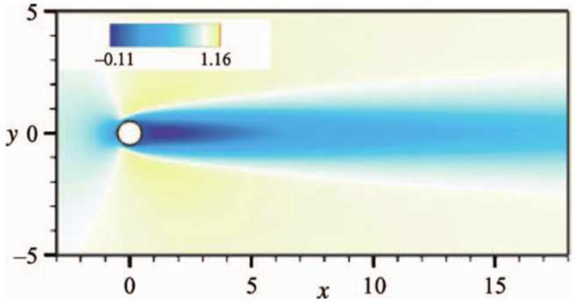
Step 1 : compute a steady solution



Steady Navier-Stokes solution ($Re = 50$) [Sipp et al, 2010]

A classical example

Step 1 : compute a steady solution



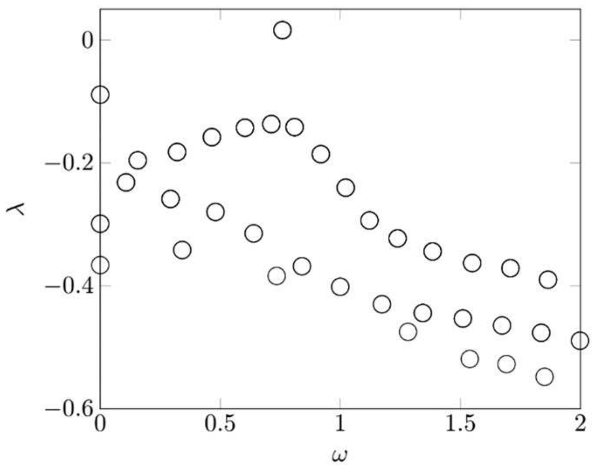
Steady Navier-Stokes solution ($Re = 50$) [Sipp et al, 2010]

Step 2 : test its stability for small monochromatic perturbations $\widehat{\boldsymbol{q}}(x)e^{\sigma t}$ around the steady solution $\boldsymbol{q_b}$

$$\sigma M\widehat{\boldsymbol{q}} = J(\boldsymbol{q_b})\widehat{\boldsymbol{q}}$$

$$\lambda = \Re[\sigma]$$

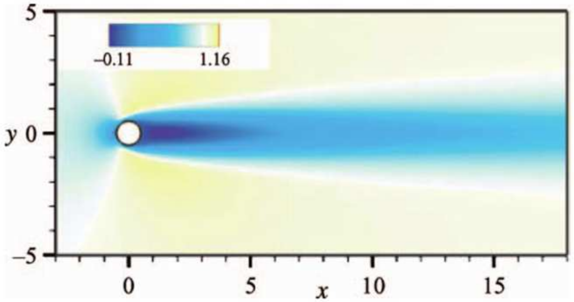
$$\omega = \Im[\sigma]$$



Introduction

A classical example

Step 1 : compute a steady solution



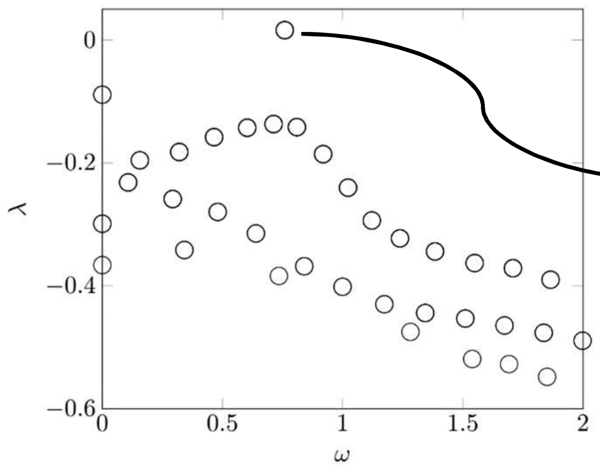
Steady Navier-Stokes solution ($Re = 50$) [Sipp et al, 2010]

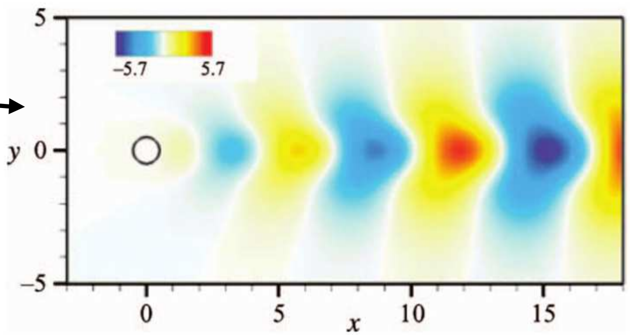
Step 2 : test its stability for small monochromatic perturbations $\widehat{\boldsymbol{q}}(x)e^{\sigma t}$ around the steady solution $\boldsymbol{q_b}$

$$\sigma M\widehat{\boldsymbol{q}} = J(\boldsymbol{q_b})\widehat{\boldsymbol{q}}$$

$$\lambda = \Re[\sigma]$$

$$\omega = \Im[\sigma]$$





Unstable eigenmode at $Re = 50$ [Sipp et al, 2010]

Why Linear Stability Analysis ?

Some nice features :

- Easy to determine a threshold value (sign of $\Re[\sigma]$)
- Less expensive than nonlinear time-integration

But some computational burdens :

- Find a (not necessarily stable) steady solution : **Newton method**
  - Multiple inversions of J
- Find *internal* eigenvalues of generalized EV problems : **Krylov-Schur + shift-and-invert**
  - Multiple inversions of $\mathrm{J} - s\,M$, where s is the shift

Introduction

How to invert matrix of the type $\mathbf{J} - s\,M$ efficiently ?

- For reasonably small configurations : direct sparse solvers (MUMPS, SUPERLU, etc)
- For large configurations : iterative method (GMRES, BiCGSTAB, …) + **good preconditioner**

Introduction

Linearized incompressible Navier-Stokes operator (i.e. $J - s\,M$) :

$$s\,\boldsymbol{u} + (\boldsymbol{u_b}\cdot\nabla)\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u_b} + \nabla p - \frac{1}{Re}\nabla^2\boldsymbol{u} = \boldsymbol{f}$$
$$-\nabla\cdot\boldsymbol{u} = \boldsymbol{g}$$

Complex shift
($s = 0$ in Newton)

Once discretized with FE : classical **saddle-point** problem

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

How to precondition this ?

- SIMPLE [Patankar 1980]
- Stokes Preconditioner [Tuckerman, 1989] (based on adaptation of existing time-stepping code)
- Pressure Convection Diffusion [Silvester et al. 2001]
- Least-Squares Commutator [Elman et al. 2006]
- Augmentated Lagrangian [Benzi and Olshanskii 2006], [Heister and Rapin 2013]

Introduction

Linearized incompressible Navier-Stokes operator (i.e. $J - s\,M$) :

$$s\,\boldsymbol{u} + (\boldsymbol{u_b}\cdot\nabla)\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u_b} + \nabla p - \frac{1}{Re}\nabla^2\boldsymbol{u} = \boldsymbol{f}$$

$$-\nabla\cdot\boldsymbol{u} = \boldsymbol{g}$$

Complex shift
($s = 0$ in Newton)

Once discretized with FE : classical **saddle-point** problem

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

How to precondition this ?

- SIMPLE [Patankar 1980]
- Stokes Preconditioner [Tuckerman, 1989] (based on adaptation of existing time-stepping code)
- Pressure Convection Diffusion [Silvester et al. 2001]
- Least-Squares Commutator [Elman et al. 2006]
- Augmentated Lagrangian [Benzi and Olshanskii 2006], [Heister and Rapin 2013]

ONERA
THE FRENCH AEROSPACE LAB

Overview

1 – Augmentation-based preconditioners

2 – Performances

3 – FreeFem++ parallel implementation

4 – Parallel 3D numerical examples

5 – Some further refinement …

Augmented Lagrangian (algebraic augmentation)

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \longrightarrow \begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f_\gamma \\ g \end{pmatrix}$$

$$A_\gamma = A + \gamma B^T W^{-1} B$$

$$f_\gamma = f + \gamma B^T W^{-1} g$$

Augmented Lagrangian (algebraic augmentation)

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \longrightarrow \begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f_\gamma \\ g \end{pmatrix}$$

$$A_\gamma = A + \gamma B^T W^{-1} B$$

$$f_\gamma = f + \gamma B^T W^{-1} g$$

Grad-Div augmentation (variational augmentation)

$$\int_{\boldsymbol{\Omega}} s\,\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + Re^{-1}\boldsymbol{\nabla}\boldsymbol{u}:\boldsymbol{\nabla}\breve{\boldsymbol{u}} - \mathrm{p}\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}$$

$$+\int_{\boldsymbol{\Omega}} \gamma(\boldsymbol{\nabla}\cdot\boldsymbol{u})(\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}) \quad = \mathbf{0}$$

$$-\int_{\boldsymbol{\Omega}} (\boldsymbol{\nabla}\cdot\boldsymbol{u})\breve{\boldsymbol{q}} = \mathbf{0}$$

1- Augmentation-based preconditioners

Augmented problems

Augmented Lagrangian (algebraic augmentation)

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \longrightarrow \begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f_\gamma \\ g \end{pmatrix}$$

$$A_\gamma = A + \gamma B^T W^{-1} B$$
$$f_\gamma = f + \gamma B^T W^{-1} g$$

Grad-Div augmentation (variational augmentation)

$$\int_\Omega s\,\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + Re^{-1}\boldsymbol{\nabla}\boldsymbol{u}:\boldsymbol{\nabla}\breve{\boldsymbol{u}} - \mathrm{p}\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}$$
$$+\int_\Omega \gamma(\boldsymbol{\nabla}\cdot\boldsymbol{u})(\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}) \quad = \boldsymbol{0}$$

$$-\int_\Omega (\boldsymbol{\nabla}\cdot\boldsymbol{u})\breve{\boldsymbol{q}} = \boldsymbol{0}$$

Augmented Lagrangian leaves the **discrete** solution unchanged
Grad-Div leaves the **continuous** solution unchanged

Augmented Lagrangian (algebraic augmentation)

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \longrightarrow \begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f_\gamma \\ g \end{pmatrix}$$

$$A_\gamma = A + \gamma B^T W^{-1} B$$
$$f_\gamma = f + \gamma B^T W^{-1} g$$

Grad-Div augmentation (variational augmentation)

$$\int_\Omega s\,\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + Re^{-1}\boldsymbol{\nabla}\boldsymbol{u}:\boldsymbol{\nabla}\breve{\boldsymbol{u}} - \mathrm{p}\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}$$
$$+\int_\Omega \gamma(\boldsymbol{\nabla}\cdot\boldsymbol{u})(\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}) \quad = \mathbf{0}$$

$$-\int_\Omega (\boldsymbol{\nabla}\cdot\boldsymbol{u})\breve{\boldsymbol{q}} = \mathbf{0}$$

Augmented Lagrangian leaves the **discrete** solution unchanged
Grad-Div leaves the **continuous** solution unchanged

1- Augmentation-based preconditioners

Classical vs. *modified* version

In both cases, the same block structure arises :

$$\begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix} = \underbrace{\begin{pmatrix} I & 0 \\ BA_\gamma^{-1} & I \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} A_\gamma & 0 \\ 0 & S \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} I & A_\gamma^{-1}B^T \\ 0 & I \end{pmatrix}}_{U}$$

$$S = -\mathrm{B}\mathrm{A}_\gamma^{-1}\mathrm{B}^\mathrm{T}$$

Classical preconditioner

$$\mathcal{P}_{class} = DU = \begin{pmatrix} A_\gamma & B^T \\ 0 & S \end{pmatrix}$$

with

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - s(BM_uB^T)^{-1}$$

$A_\gamma^{-1} \simeq$ it's complicated …

Main features :
- Mesh optimality
- Reynolds optimality
- The higher γ, the less iterations (A_γ^{-1} ouch !)

1- Augmentation-based preconditioners

Classical vs. *modified* version

In both cases, the same block structure arises :

$$\begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix} = \underbrace{\begin{pmatrix} I & 0 \\ BA_\gamma^{-1} & I \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} A_\gamma & 0 \\ 0 & S \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} I & A_\gamma^{-1}B^T \\ 0 & I \end{pmatrix}}_{U} \qquad S = -\mathrm{B}\mathrm{A}_\gamma^{-1}\mathrm{B}^\mathrm{T}$$

Classical preconditioner

$$\mathcal{P}_{class} = DU = \begin{pmatrix} A_\gamma & B^T \\ 0 & S \end{pmatrix}$$

with

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - s(BM_uB^T)^{-1}$$

$A_\gamma^{-1} \simeq$ it's complicated …

Main features :

- Mesh optimality
- Reynolds optimality
- The higher γ, the less iterations (A_γ^{-1} ouch !)

Modified preconditioner

$$\mathcal{P}_{modif} = \begin{pmatrix} \begin{bmatrix} A_{11,\gamma} & A_{12,\gamma} \\ 0 & A_{22,\gamma} \end{bmatrix} & B^T \\ 0 & S \end{pmatrix}$$

with

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - s(BM_uB^T)^{-1}$$

$A_{ii,\gamma}^{-1} \simeq$ off-the-shelf algebraic multigrid

Main features :

- Mesh optimality
- Reynolds dependent
- Exists an optimal and case dependent γ

1- Augmentation-based preconditioners

Classical vs. *modified* version

In both cases, the same block structure arises :

$$\begin{pmatrix} A_\gamma & B^T \\ B & 0 \end{pmatrix} = \underbrace{\begin{pmatrix} I & 0 \\ BA_\gamma^{-1} & I \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} A_\gamma & 0 \\ 0 & S \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} I & A_\gamma^{-1}B^T \\ 0 & I \end{pmatrix}}_{U}$$

$$S = -\mathrm{B}\mathrm{A}_\gamma^{-1}\mathrm{B}^\mathrm{T}$$

Classical preconditioner

$$\mathcal{P}_{class} = DU = \begin{pmatrix} A_\gamma & B^T \\ 0 & S \end{pmatrix}$$

with

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - s(BM_uB^T)^{-1}$$

$A_\gamma^{-1} \simeq$ it's complicated …

Main features :

- Mesh optimality
- Reynolds optimality
- The higher γ, the less iterations (A_γ^{-1} ouch !)

Modified preconditioner

$$\mathcal{P}_{modif} = \begin{pmatrix} \begin{bmatrix} A_{11,\gamma} & A_{12,\gamma} \\ 0 & A_{22,\gamma} \end{bmatrix} & B^T \\ 0 & S \end{pmatrix}$$

with

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - s(BM_uB^T)^{-1}$$

$A_{ii,\gamma}^{-1} \simeq$ off-the-shelf algebraic multigrid

Main features :

- Mesh optimality
- Reynolds dependent
- Exists an optimal and case dependent γ

The choice of a good γ is determinant for the preconditioning efficiency !

Bright side : since the preconditioner is independent of the mesh

- Optimal γ can be found on a coarse mesh

Dark side : Optimal γ is problem and Re – dependent

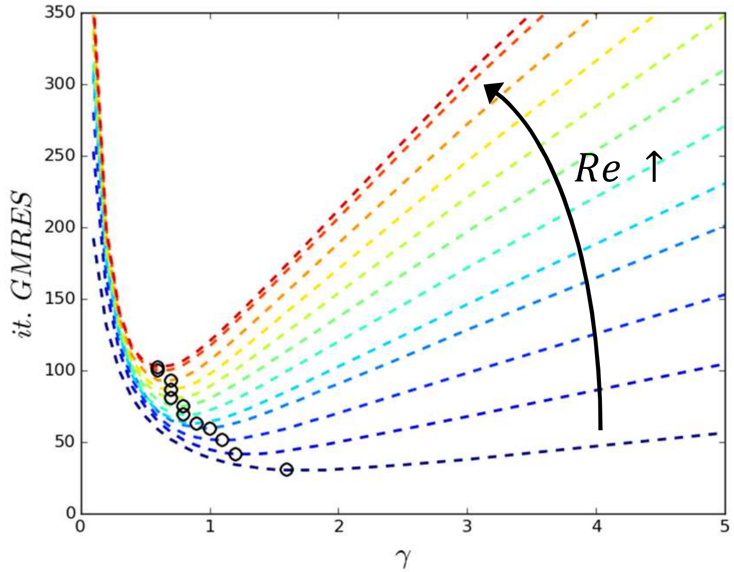


Figure : Influence of $Re \in [10,120]$ on optimal γ for modified Grad-Div preconditioner

2- Performances
CPU time in Newton method

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS			Modified Grad-Div		
			Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$	Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$
32x32	9900	1300	140	0	20	30	50	14
64x64	39000	5000	810	10	27	320	250	20
96x96	88000	11000	2250	40	33	840	580	21
256x256	623400	78200	34480	290	62	8090	4780	25

Averaged timings for 1 Newton iteration (**2D** lid-driven cavity, $Re = 100, \gamma = 0{,}1$)

*All sub-systems are solved with MUMPS

2- Performances

CPU time in Newton method

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS			Modified Grad-Div		
			Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$	Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$
32x32	9900	1300	140	0	20	30	50	14
64x64	39000	5000	810	10	27	320	250	20
96x96	88000	11000	2250	40	33	840	580	21
256x256	623400	78200	34480	290	62	8090	4780	25

Averaged timings for 1 Newton iteration (**2D** lid-driven cavity, $Re = 100, \gamma = 0{,}1$)

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS			Modified Grad-Div		
			Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$	Facto $(\mathbf{ms})$	Reso $(\mathbf{ms})$	tot/ndof $(\boldsymbol{\mu s})$
8x8x8	9900	1300	3,2	0,01	263	0,6	0,26	112
16x16x16	39000	5000	295	0,3	2675	21	2,8	274

Averaged timings for 1 Newton iteration (**3D** lid-driven cavity, $Re = 100, \gamma = 0{,}1$)

*All sub-systems are solved with MUMPS

2- Performances

CPU time for eigenvalue computation

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS		Modified Grad-Div	
			Fact [s]	Eig [s]	Fact [s]	Eig [s] (it. inner GMRES)
32x32	9890	1269	0,27	0,36	0,05	9 (29)
64x64	39306	4978	1,7	1,3	0,45	34 (30)
256x256	623482	78192	85	36	15	841 (30)

Timings for computing 10 ev with ARPACK (**2D** lid-driven cavity, $Re = 100$, $\gamma = 0{,}1$)

*All sub-systems are solved with MUMPS

2- Performances

CPU time for eigenvalue computation

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS		Modified Grad-Div	
			Fact [s]	Eig [s]	Fact [s]	Eig [s] (it. inner GMRES)
32x32	9890	1269	0,27	0,36	0,05	9 (29)
64x64	39306	4978	1,7	1,3	0,45	34 (30)
256x256	623482	78192	85	36	15	841 (30)

Timings for computing 10 ev with ARPACK (**2D** lid-driven cavity, $Re = 100, \gamma = 0{,}1$)

Mesh	Velocity DOFs	Pressure DOFs	Full MUMPS		Modified Grad-Div	
			Fact [s]	Eig [s]	Fact [s]	Eig [s] (it. inner GMRES)
8x8x8	14739	729	8	2	1,6	35 (24)
16x16x16	107811	4913	753	31	57	353 (23)

Timings for computing 10 ev with ARPACK (**3D** lid-driven cavity, $Re = 100, \gamma = 0{,}1$)

*All sub-systems are solved with MUMPS

2- Performances

What to remember ?

Iterative strategy will be faster than the direct solver when :
time facto >> time solving

- For **Newton method** : **always the case** because the jacobian is new at each iteration
- For **eigenvalue** computation : true **only for large configurations** (3D typically)

2- Performances

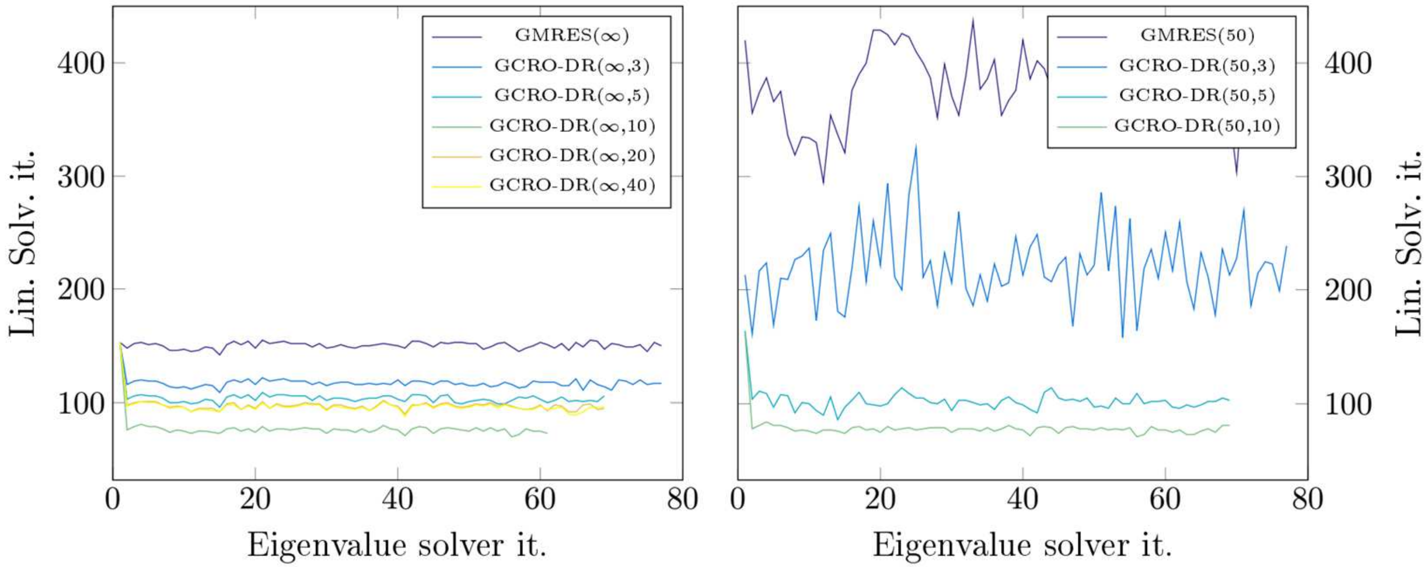
Krylov subspace recycling techniques and eigenvalue computation

Idea : In Krylov-Schur + shift-invert, one has to perform many $(J - s\, M)^{-1}$ with the same matrix !

- ➢ Why not use Krylov subspace recycling from one linear solve to the next ?

Idea : In Krylov-Schur + shift-invert, one has to perform many $(J - s\ M)^{-1}$ with the same matrix !

- Why not use Krylov subspace recycling from one linear solve to the next ?



$n_{Arnoldi} = +\infty$ (no restart) $\quad n_{Arnoldi} = 50$

Figure : Effect of recycling during eigenvalue computation. Test case : 2D circular cylinder at $Re = 50$.
Preconditioner : Modified Grad-Div with $\gamma = 1$
Eigenvalue solver : ARPACK with shift-invert

3- Parallel implementation in FreeFem++

PETSc/SLEPc interface (P. Jolivet)

Ingredient 1 : handle the preconditioner's block structure

PETSc solution : use of `PCFIELDSPLIT` preconditioner

FreeFem++ interface :

```
fespace Wh(th,[P2,P2,P2,P1]); // full space

Wh [u,v,w,p];
Wh [b,bv,bw,bp] = [1.0, 2.0, 3.0, 4.0];
string[int] names(4);
names[0] = "xvelocity" ;
names[1] = "yvelocity" ;
names[2] = "zvelocity" ;
names[3] = "pressure" ;

// Set PETSc solver
set(A, sparams = " -ksp_type fgmres -pc_type fieldsplit -pc_fieldsplit_type multiplicative"
               + " -prefix_push fieldsplit_xvelocity_"
                        + " -ksp_type preonly -pc_type lu -pc_factor_mat_solver_package mumps"
               + " -prefix_pop"
               + " -prefix_push fieldsplit_yvelocity_" , … … …
    fields = b[], names = names);
```


ONERA THE FRENCH AEROSPACE LAB

3- Parallel implementation in FreeFem++

PETSc/SLEPc interface (P. Jolivet)

Ingredient 2 : provide a specific Schur complement approximation

PETSc solution :

```
PCFieldSplitGetSubKSP(pc, &nfields, &subksp)
KSPSetOperators(subksp[nfields-1], Sapprox, Sapprox)
```

FreeFem++ interface :

```
fespace Wh(th,[P2,P2,P2,P1]); // full space
fespace Qh(th,P1); // pressure space

Wh [u,v,w,p];
Wh [b,bv,bw,bp] = [1.0, 2.0, 3.0, 4.0];
string[int] names(4);
names[0] = "xvelocity" ;
names[1] = "yvelocity" ;
names[2] = "zvelocity" ;
names[3] = "pressure" ;
Qh pind;
pind[] = 1:pind[].n;
Wh [list, listv, listw, listp]= [0, 0, 0, pind]; // correspondance between Wh and Qh pressure DOFs
matrix[int] S(1);
S[0]=vSchur(Qh,Qh); // Schur complement approximation

// Set PETSc solver
set(A, sparams = " … … … " ,
       fields = b[], names = names, schurPreconditioner = S, schurList = list[]);
```

Ingredient 3 : provide the inverse Schur complement approx. as a composition of two simple inverses

$$S^{-1} \simeq (\mathrm{Re}^{-1} + \gamma)M_p^{-1} - sL_p^{-1}$$

PETSc solution : use of `PCCOMPOSITE` preconditioner

FreeFem++ interface :

```
matrix[int] S(2);
S[0]=vMp(Qh,Qh); // pressure mass matrix
S[1]=vLp(Qh,Qh); // pressure laplacian matrix

// Set PETSc solver
set(A, sparams = " … … … "
               + " -prefix_push fieldsplit_pressure_
                    -ksp_type preonly -pc_type composite -pc_composite_type additive"
                                   + " -prefix_push sub_0_"
                                               + " -pc_type ksp -ksp_ksp_type cg -ksp_pc_type jacobi"
                                   + " -prefix_pop"
                                   + " -prefix_push sub_1_"
                                               + " -pc_type ksp -ksp_ksp_type fgmres -ksp_pc_type gamg"
                                   + " -prefix_pop"
               + " -prefix_pop" ,
       fields = b[], names = names, schurPreconditioner = S, schurList = list[]);
```

3- Parallel implementation in FreeFem++

PETSc/SLEPc interface (P. Jolivet)

Ingredient 4 : Recycling of Krylov basis bewteen two consecutive solve $(J - s\,M)^{-1}$ in SLEPc

PETSc solution : interface HPDDM's solvers with PETSc/SLEPc

FreeFem++ interface :

```
// Set SLEPc eigensolver Ax = sigma Bx
int k = zeigensolver
(DistA,
 DistB,
 vectors = EigenVEC, // Array to store the FEM-EigenFunctions
 values  = EigenVAL, // Array to store the EigenValues
 sparams = " -eps_type krylovschur -st_type sinvert  -eps_target 0+0.6i"
               + " -st_ksp_type hpddm -hpddm_st_krylov_method gcrodr -hpddm_st_variant flexible … ",
       fields = b[], names = names, schurPreconditioner = S, schurList = list[]);
```

4- Parallel 3D numerical examples

Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]

<u>Test case</u> :

- 1 million tetrahedrons / Taylor-Hood FE pair / 4,8 millions DOFs
- $Re = 100$

<u>Solvers</u> :

- Steady solution : Newton method with FGMRES preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block solved with CG preconditionned by jacobi, tol=10^{-3}

- Eigenvalues : Krylov-Schur + shift-invert + GCRO-DR(100,30) preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block 1 solved with CG preconditionned by jacobi, tol=10^{-3}
  - Schur complement sub-block 2 solved with FMGRES preconditionned by gamg, tol=10^{-3}

4- Parallel 3D numerical examples

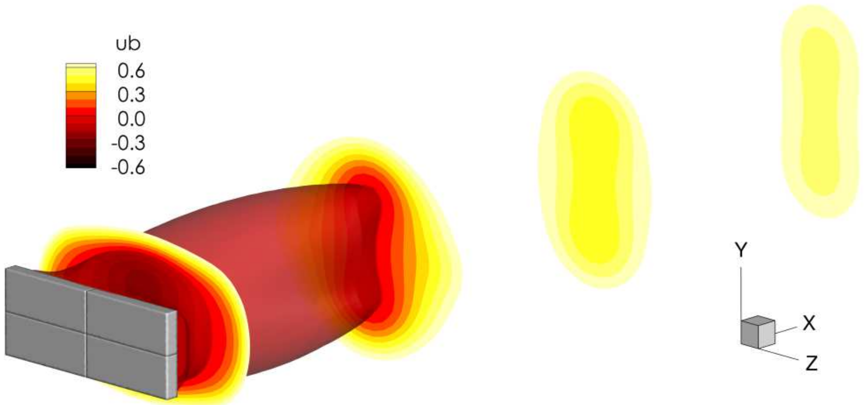
Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]

Test case :

- 1 million tetrahedrons / Taylor-Hood FE pair / 4,8 millions DOFs
- $Re = 100$

Solvers :

- Steady solution : Newton method with FGMRES preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block solved with CG preconditionned by jacobi, tol=10^{-3}
- Eigenvalues : Krylov-Schur + shift-invert + GCRO-DR(100,30) preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block 1 solved with CG preconditionned by jacobi, tol=10^{-3}
  - Schur complement sub-block 2 solved with FMGRES preconditionned by gamg, tol=10^{-3}



Steady solution (axial velocity)

4- Parallel 3D numerical examples

Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]

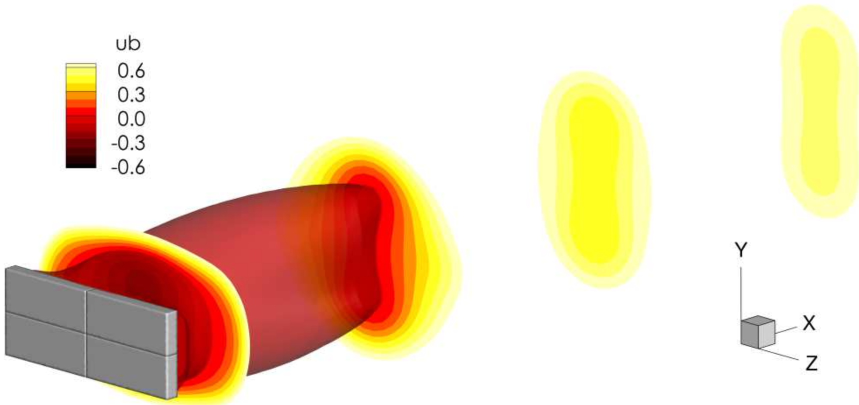
Test case :

- 1 million tetrahedrons / Taylor-Hood FE pair / 4,8 millions DOFs
- $Re = 100$

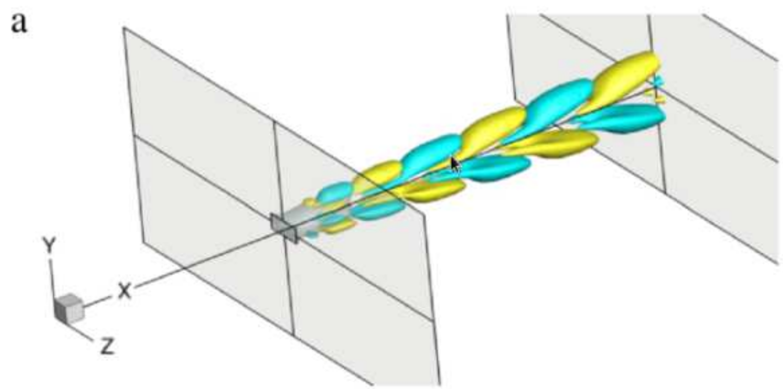
Solvers :

- Steady solution : Newton method with FGMRES preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block solved with CG preconditionned by jacobi, tol=10^{-3}

- Eigenvalues : Krylov-Schur + shift-invert + GCRO-DR(100,30) preconditioned by **Modified Grad-Div** ($\gamma_{optimal} = 0{,}3$)
  - velocity sub-blocks solved with FGMRES preconditionned by ASM, overlap=1, tol=10^{-1}
  - Schur complement sub-block 1 solved with CG preconditionned by jacobi, tol=10^{-3}
  - Schur complement sub-block 2 solved with FMGRES preconditionned by gamg, tol=10^{-3}



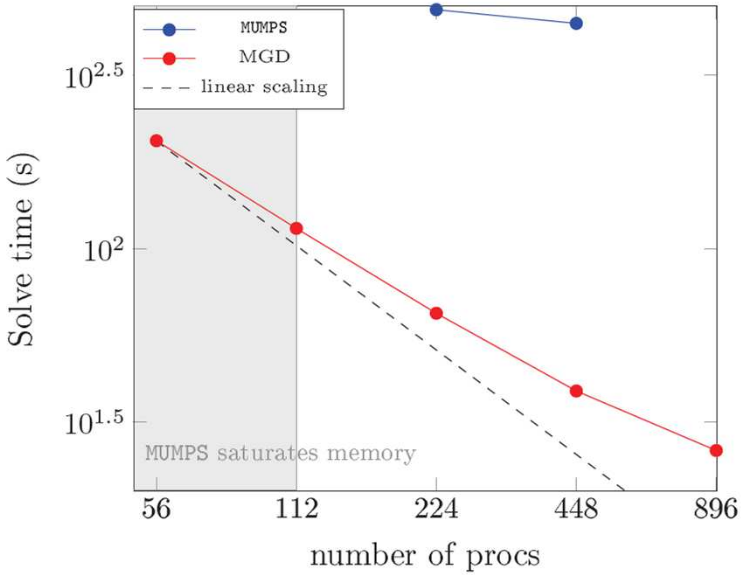
Steady solution (axial velocity)



Marginally stable mode ($\lambda = 0\,; \omega = 0{,}58$)
from [Marquet & Larsson 2015]

4- Parallel 3D numerical examples

Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]

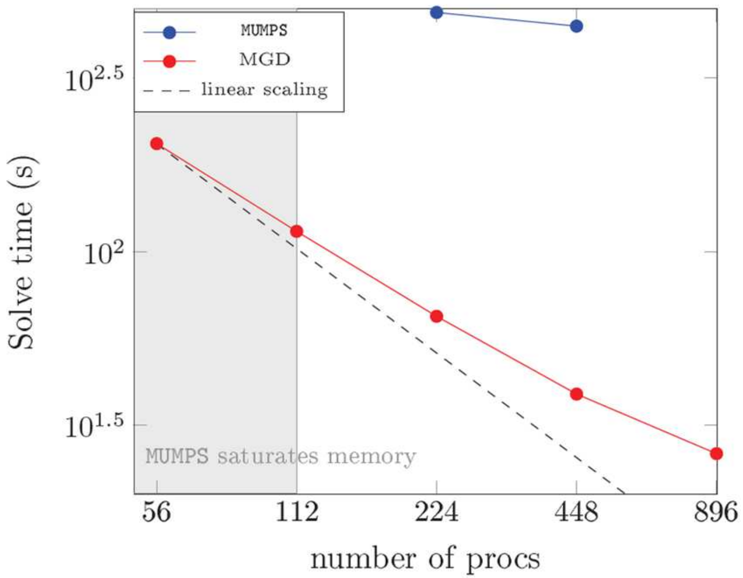


Newton Method

(average iteration time is represented)

4- Parallel 3D numerical examples

Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]

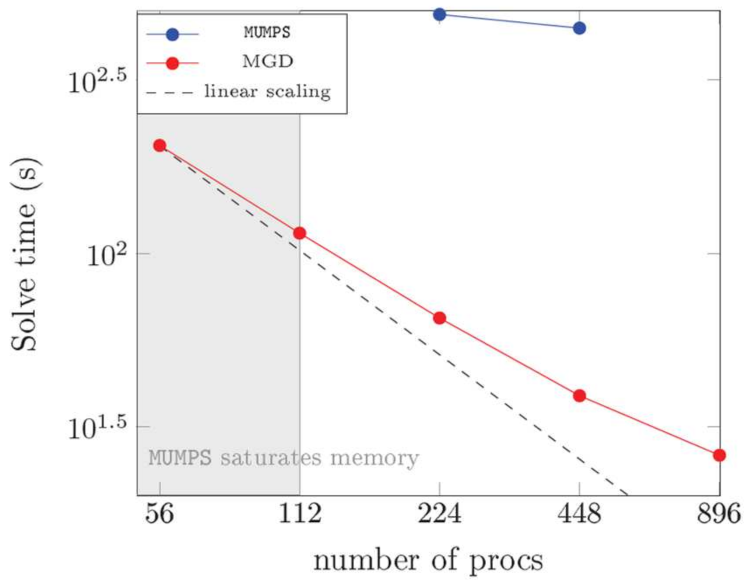


Newton Method
(average iteration time is represented)

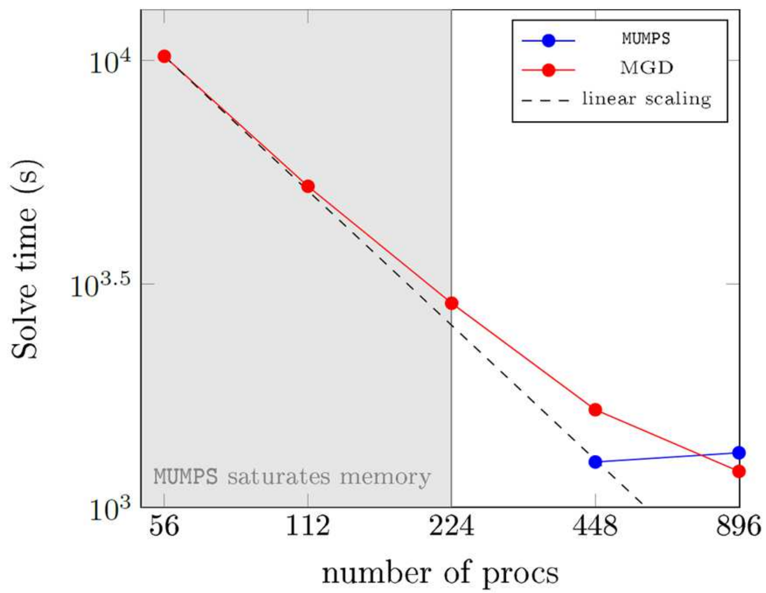
The loss of scaling for high number of procs is mainly due to the non-optimality of ASM w.r.t. number of domains. To be improved …

4- Parallel 3D numerical examples

Flow around low aspect-ratio flat plates [Marquet & Larsson 2015]



Newton Method
(average iteration time is represented)



Eigenvalue computation
(10 ev requested with tolerance 10^{-6})

The loss of scaling for high number of procs is mainly due to the non-optimality of ASM w.r.t. number of domains. To be improved …

5- Some further refinement …

Influence of γ on the solution ?

Grad-Div augmentation (variational augmentation)

$$\int_{\Omega} s\,\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + Re^{-1}\boldsymbol{\nabla}\boldsymbol{u}:\boldsymbol{\nabla}\breve{\boldsymbol{u}} - \mathrm{p}\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}$$

$$+\int_{\Omega} \gamma(\boldsymbol{\nabla}\cdot\boldsymbol{u})(\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}) \;\; = \boldsymbol{0}$$

$$-\int_{\Omega} (\boldsymbol{\nabla}\cdot\boldsymbol{u})\breve{\boldsymbol{q}} = \boldsymbol{0}$$

Grad-Div leaves the **continuous** solution unchanged

But … **changes the discrete solution !**

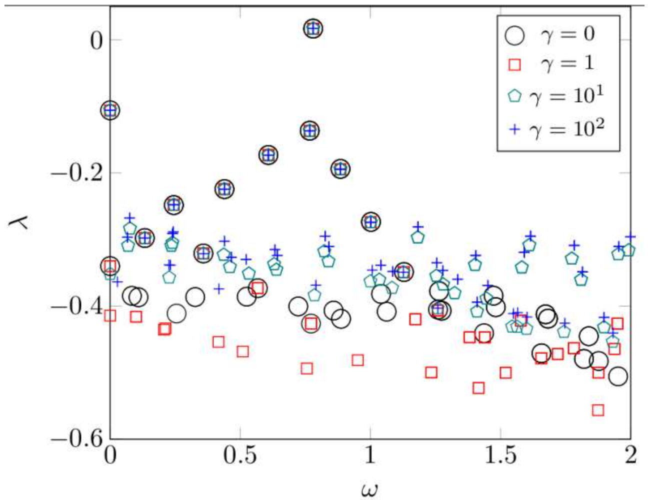


Figure : Eigenvalue spectrum of the flow around a 2D circular cylinder at $Re = 50$

Spatial discretization : Taylor-Hood (P_2, P_1)

5- Some further refinement …

Influence of γ on the solution ?

Grad-Div augmentation (variational augmentation)

$$\int_{\Omega} s\,\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\breve{\boldsymbol{u}} + Re^{-1}\boldsymbol{\nabla u}:\boldsymbol{\nabla}\breve{\boldsymbol{u}} - \mathrm{p}\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}$$

$$+\int_{\Omega} \gamma(\boldsymbol{\nabla}\cdot\boldsymbol{u})(\boldsymbol{\nabla}\cdot\breve{\boldsymbol{u}}) \;\; = \boldsymbol{0}$$

$$-\int_{\Omega} (\boldsymbol{\nabla}\cdot\boldsymbol{u})\breve{\boldsymbol{q}} = \boldsymbol{0}$$

Grad-Div leaves the **continuous** solution unchanged

But … **changes the discrete solution !**

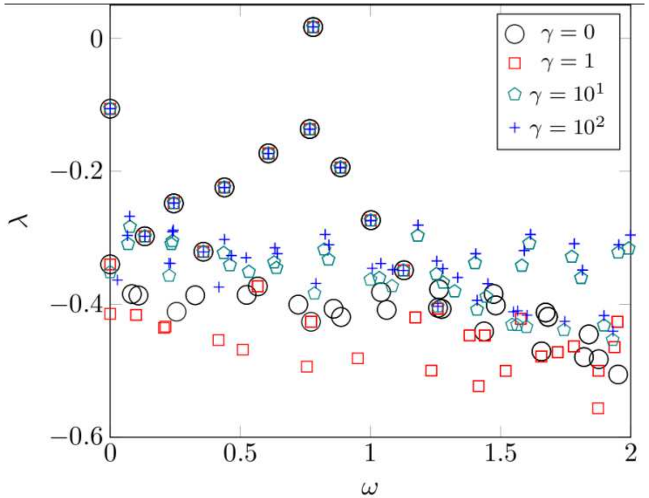


Figure : Eigenvalue spectrum of the flow around a 2D circular cylinder at $Re = 50$

Spatial discretization : Taylor-Hood (P_2, P_1)

Not a divergence free element !!

$\nabla\cdot(P_2) \notin P_1$

5- Some further refinement …

Influence of γ on the solution ?

What if one uses a **divergence-free** element ?

- Scott-Vogelius FE pair : (P_2, P_1^{dc}) $\quad s.t. \quad \nabla \cdot (P_2) \in P_1^{dc}$

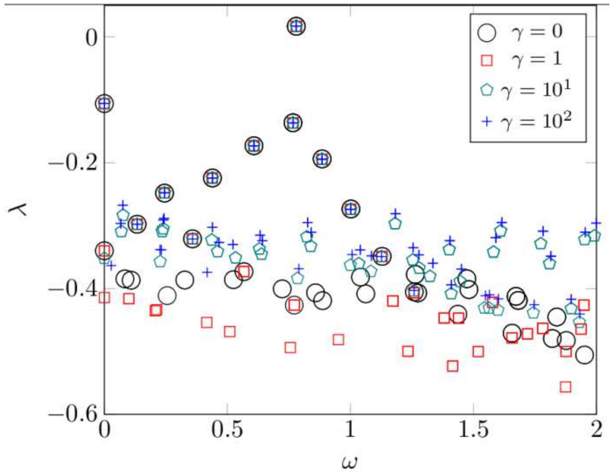


Figure : Eigenvalue spectrum of the flow around a 2D circular cylinder at $Re = 50$

Spatial discretization : Taylor-Hood (P_2, P_1)

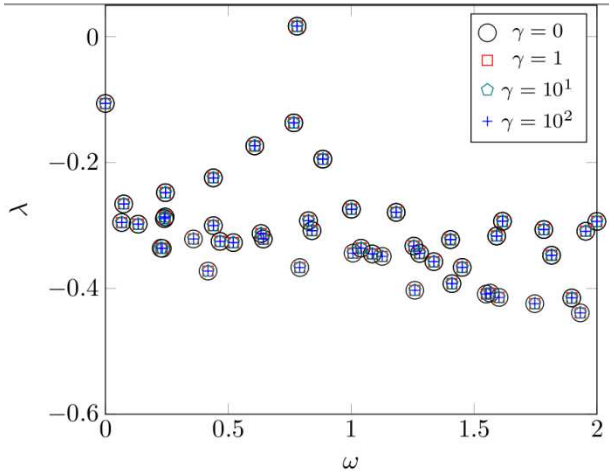


Figure : Eigenvalue spectrum of the flow around a 2D circular cylinder at $Re = 50$

Spatial discretization : Scott-Vogelius (P_2, P_1^{dc})

A few remarks :

- (P_2, P_1^{dc}) is inf-sup stable only on specific types of mesh (Hsieh-Clough-Toucher triangulation)
- We showed that when using **divergence-free elements** the **variational and discrete augmentations are equivalent**
- It is unprcatical to use the discrete augmentation without divergence free elements due to the unsparse nature of the augmentation term …

ONERA THE FRENCH AEROSPACE LAB

Conclusion

Conclusion :

- **Krylov subspaces iterative** method preconditioned by **Modified Grad-Div** where shown to be **efficient** both for finding a steady solution and computing its spectrum
- **Large 3D configurations and large number of processors** accentuate the benefits of using the iterative strategy w.r.t. direct solver.
- **Ritz vector recycling** was shown to provide significant acceleration of the eigenvalue computation when using an iterative strategy for $(J - s\,M)^{-1}$
- **A parallel implementation in FreeFem++**/PETSc/SLEPc was proposed.

Conclusion

Conclusion :

- **Krylov subspaces iterative** method preconditioned by **Modified Grad-Div** where shown to be **efficient** both for finding a steady solution and computing its spectrum
- **Large 3D configurations and large number of processors** accentuate the benefits of using the iterative strategy w.r.t. direct solver.
- **Ritz vector recycling** was shown to provide significant acceleration of the eigenvalue computation when using an iterative strategy for $(J - s\,M)^{-1}$
- **A parallel implementation in FreeFem++**/PETSc/SLEPc was proposed.

Perspectives :

- Scalings must be improved : find an optimal preconditioner for velocity sub-blocks
- Extension for preconditioning turbulence models (RANS equations)
- Towards coupled fluid-structure Linear Stability Analysis on large 3D configurations …

Conclusion

Conclusion :

- **Krylov subspaces iterative** method preconditioned by **Modified Grad-Div** where shown to be **efficient** both for finding a steady solution and computing its spectrum
- **Large 3D configurations and large number of processors** accentuate the benefits of using the iterative strategy w.r.t. direct solver.
- **Ritz vector recycling** was shown to provide significant acceleration of the eigenvalue computation when using an iterative strategy for $(J - s\,M)^{-1}$
- **A parallel implementation in FreeFem++**/PETSc/SLEPc was proposed.

Perspectives :

- Scalings must be improved : find an optimal preconditioner for velocity sub-blocks
- Extension for preconditioning turbulence models (RANS equations)
- Towards coupled fluid-structure Linear Stability Analysis on large 3D configurations …

Fluid-structure Jacobian matrix = $\begin{pmatrix} J_{ff} & J_{fs} \\ J_{sf} & J_{ss} \end{pmatrix}$

Conclusion

Conclusion :

- **Krylov subspaces iterative** method preconditioned by **Modified Grad-Div** where shown to be **efficient** both for finding a steady solution and computing its spectrum
- **Large 3D configurations and large number of processors** accentuate the benefits of using the iterative strategy w.r.t. direct solver.
- **Ritz vector recycling** was shown to provide significant acceleration of the eigenvalue computation when using an iterative strategy for $(J - s\,M)^{-1}$
- **A parallel implementation in FreeFem++**/PETSc/SLEPc was proposed.

Perspectives :

- Scalings must be improved : find an optimal preconditioner for velocity sub-blocks
- Extension for preconditioning turbulence models (RANS equations)
- Towards coupled fluid-structure Linear Stability Analysis on large 3D configurations …

Fluid-structure Jacobian matrix = $\begin{pmatrix} J_{ff} & J_{fs} \\ J_{sf} & J_{ss} \end{pmatrix}$

Modified Grad-Div

Questions

2- Performances

Memory requirements in Newton method

Mesh	Velocity DOFs	Pressure DOFs	Memory direct MUMPS (Mb)	Memory Modified Grad-Div (Mb)	Memory gain (%)
16x16	2600	340	5	2x2	20
32x32	9900	1300	16	2x4	50
64x64	39000	5000	75	2x17	55
96x96	88000	11000	191	2x41	57
256x256	623400	78200	1862	2x353	62

Memory requirements (**2D** lid-driven cavity, $Re = 100$, $\gamma = 0{,}1$)

Mesh	Velocity DOFs	Pressure DOFs	Memory direct MUMPS (Mb)	Memory Modified Grad-Div (Mb)	Memory gain (%)
8x8x8	14700	729	143	3x21	56
16x16x16	107800	4900	2565	3x260	70

Memory requirements (**3D** lid-driven cavity, $Re = 100$, $\gamma = 0{,}1$)

*All sub-systems are solved with MUMPS