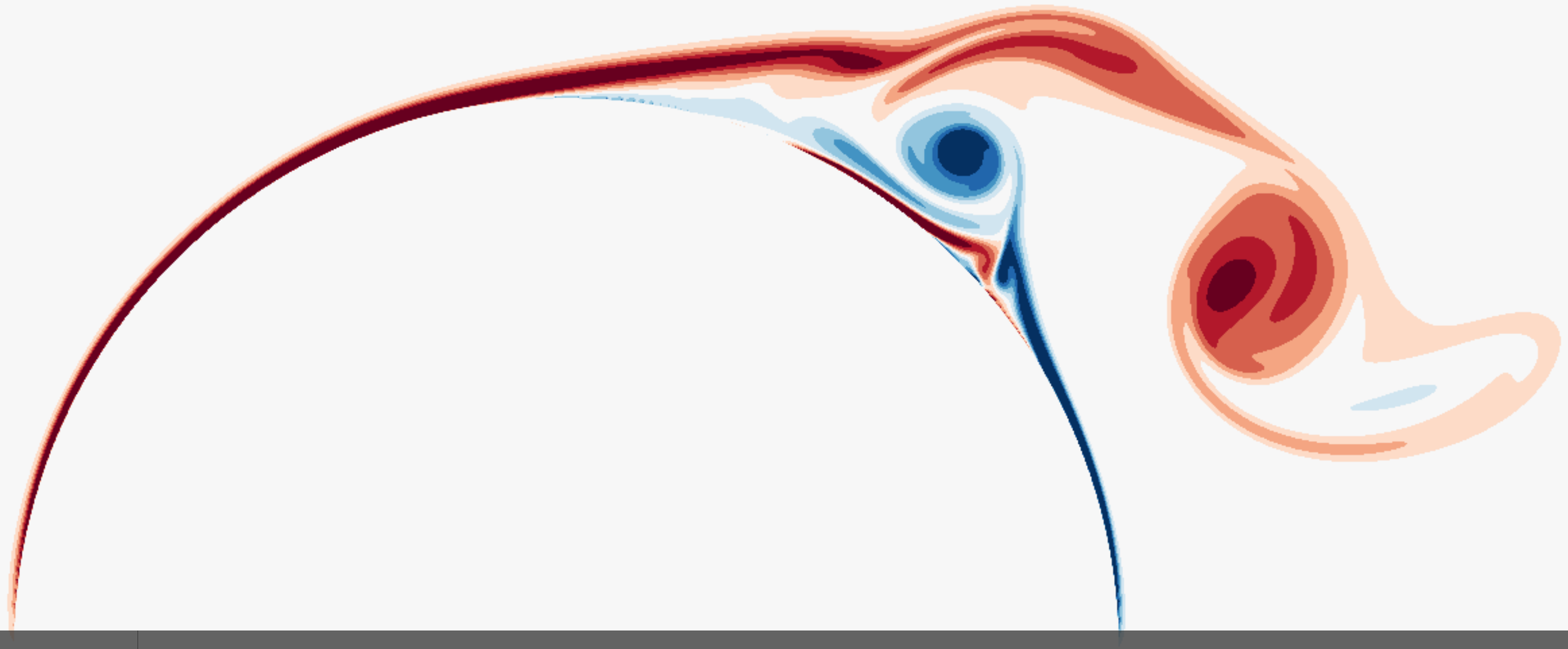
An Efficient **Iterative Penalization Method** Based on **Recycled Krylov Subspaces** and Its Application to **Impulsively Started Flows**



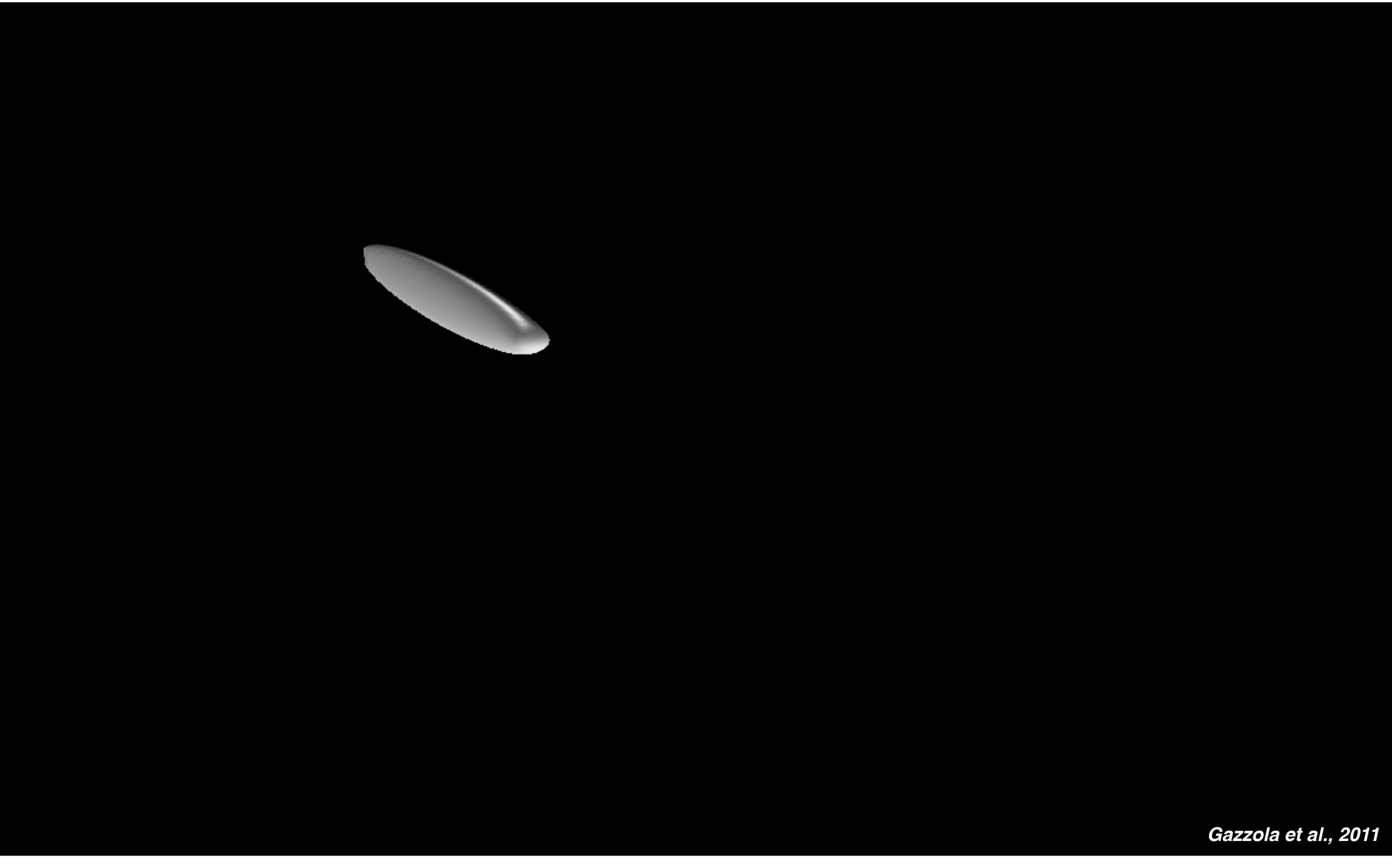
UCL Université catholique de Louvain

iMMC

Thomas Gillis,
Grégoire Winckelmans and Philippe Chatelain

CSE17
Atlanta - 27th Feb

Computational fluid dynamics challenge



Gazzola et al., 2011

Computational fluid dynamics challenge

The addressed challenge:

How to take complex geometries into account?

- Swift accelerating flows
- With low errors
- At low cost

Vortex Particle-Mesh (VPM) methods

- **Vorticity formulation** of the **incompressible Navier-Stokes** equations[1,2]

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial\omega}{\partial t} + \mathbf{u}\cdot\nabla\omega = \nu\,\nabla^2\omega$$

$$\nabla^2\mathbf{u} = -\nabla\times\omega$$

- Material **surface** with **position** and **strength**[1,3]

$$S_p \qquad \mathbf{x}_p \qquad \Gamma_p = \int_{S_p} \omega\; d\mathbf{x} \triangleq \omega_p\; S_p$$

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_p\\ \omega_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \nu\,\nabla^2\omega\end{bmatrix}_p$$

1. *Cottet and Koumoutsakos, 2000*
2. *Winckelmans, 2004*
3. *Cocle and Winckelmans, 2008*

Vortex Particle-Mesh (VPM) methods

- **Vorticity formulation** of the **incompressible Navier-Stokes** equations[1,2]

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial\omega}{\partial t} + \mathbf{u}\cdot\nabla\omega = \nu\,\nabla^2\omega$$

$$\nabla^2\mathbf{u} = -\nabla\times\omega$$ **Poisson equation**

- Material **surface** with **position** and **strength**[1,3]

$$S_p \qquad \mathbf{x}_p \qquad \Gamma_p = \int_{S_p} \omega\; d\mathbf{x} \triangleq \omega_p\; S_p$$

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_p\\ \omega_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \nu\;\nabla^2\omega\end{bmatrix}_p$$

1. *Cottet and Koumoutsakos, 2000*
2. *Winckelmans, 2004*
3. *Cocle and Winckelmans, 2008*

Vortex Particle-Mesh (VPM) methods

- **Vorticity formulation** of the **incompressible Navier-Stokes** equations[1,2]

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial\omega}{\partial t} + \mathbf{u}\cdot\nabla\omega = \nu\,\nabla^2\omega$$

$$\nabla^2\mathbf{u} = -\nabla\times\omega$$ **Poisson equation**

- Material **surface** with **position** and **strength**[1,3]

S_p $\mathbf{x}_p$ $\Gamma_p = \int_{S_p} \omega\; d\mathbf{x} \triangleq \omega_p\; S_p$



$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_p\\ \omega_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \nu\;\nabla^2\omega\end{bmatrix}_p$$

1. *Cottet and Koumoutsakos, 2000*
2. *Winckelmans, 2004*
3. *Cocle and Winckelmans, 2008*

Vortex Particle-Mesh (VPM) methods

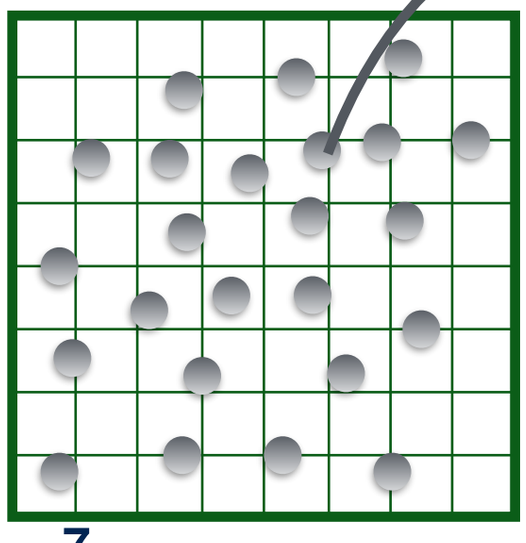
- **Vorticity formulation** of the **incompressible Navier-Stokes** equations[1,2]

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial\omega}{\partial t} + \mathbf{u}\cdot\nabla\omega = \nu\,\nabla^2\omega$$

$$\nabla^2\mathbf{u} = -\nabla\times\omega$$ **Poisson equation**

- Material **surface** with **position** and **strength**[1,3]

S_p $\mathbf{x}_p$ $\Gamma_p = \int_{S_p} \omega\, d\mathbf{x} \triangleq \omega_p\, S_p$



$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_p\\ \omega_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \nu\,\nabla^2\omega\end{bmatrix}_p$$

1. *Cottet and Koumoutsakos, 2000*
2. *Winckelmans, 2004*
3. *Cocle and Winckelmans, 2008*

The Brinkman penalization…

- Adds a **constraint** on the velocity[1,2,3]

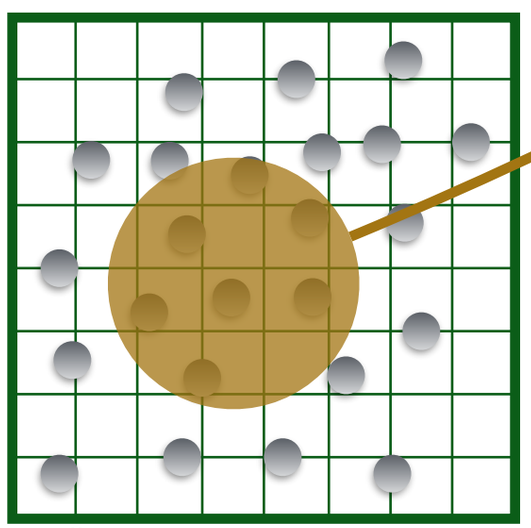
$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right]$$

where the mask function is given by

$$\chi \left(\mathbf{x} \right) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_b, \\ 0 & \text{otherwise.} \end{cases}$$

- The system to integrate then becomes

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right] \end{bmatrix}_p$$



1. *Kevlahan et al., 2001*
2. *Coquerelle et al., 2008*
3. *Hejlesen et al., 2013*

The Brinkman penalization…

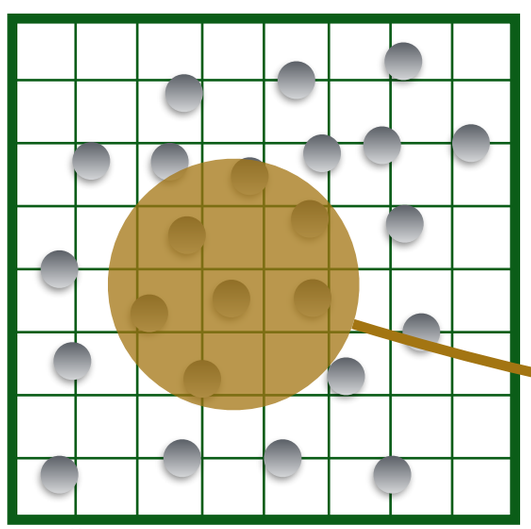
- Adds a **constraint** on the velocity[1,2,3]

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right]$$

where the mask function is given by

$$\chi \left(\mathbf{x} \right) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_b, \\ 0 & \text{otherwise.} \end{cases}$$

- The system to integrate then becomes



$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right] \end{bmatrix}_p$$

1. *Kevlahan et al., 2001*
2. *Coquerelle et al., 2008*
3. *Hejlesen et al., 2013*

The Brinkman penalization…

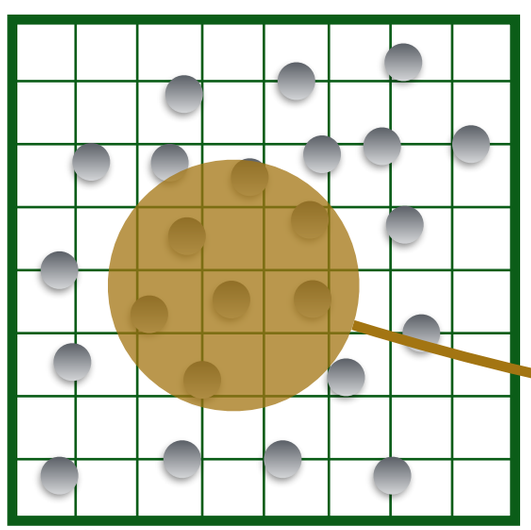
- Adds a **constraint** on the velocity[1,2,3]

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right]$$

where the mask function is given by

$$\chi \left(\mathbf{x} \right) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_b, \\ 0 & \text{otherwise.} \end{cases}$$

- The system to integrate then becomes



$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \, \nabla^2 \omega + \lambda \, \nabla \times \left[\chi \left(\mathbf{u}_b - \mathbf{u} \right) \right] \end{bmatrix}_p$$

1. *Kevlahan et al., 2001*
2. *Coquerelle et al., 2008*
3. *Hejlesen et al., 2013*

… applies a correction

- Computation scheme
  1. Solve convection and diffusion

$$\tilde{\mathbf{u}}^{n+1} \qquad \tilde{\omega}^{n+1}$$

  2. Compute the correction

$$\delta\omega = \mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right)$$

  3. Apply it

$$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$$

- The most frequent ways to compute the correction[1,2,3]:

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\frac{\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t \ \ \chi\, \mathbf{u}_b}{1 + \lambda\Delta t \ \ \chi}\right] - \tilde{\omega}^{n+1}$$

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\tilde{\mathbf{u}}^{n}\left(1 + \lambda\Delta t \ \ \chi\right) + \lambda\Delta t \ \ \chi\, \mathbf{u}_b\right] - \tilde{\omega}^{n+1}$$

1. *Coquerelle and Cottet, 2008*
2. *Gazzola et al., 2011*
3. *Mimeau et al., 2015*

… applies a correction

- Computation scheme

1. Solve convection and diffusion

$$\tilde{\mathbf{u}}^{n+1} \qquad \tilde{\omega}^{n+1}$$

2. Compute the correction

$$\delta\omega = \mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right)$$

3. Apply it

$$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$$

Drawbacks:

1. generate substantial **diffusion errors**[1]
2. **no global relation** between vorticity and velocity

- The most frequent ways to compute the correction[1,2,3]:

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\frac{\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t \ \ \chi\, \mathbf{u}_b}{1 + \lambda\Delta t \ \ \chi}\right] - \tilde{\omega}^{n+1}$$

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\tilde{\mathbf{u}}^{n}\left(1 + \lambda\Delta t \ \ \chi\right) + \lambda\Delta t \ \ \chi\, \mathbf{u}_b\right] - \tilde{\omega}^{n+1}$$

1. *Coquerelle and Cottet, 2008*
2. *Gazzola et al., 2011*
3. *Limet et al., 2015*

1. *Rasmussen et al., 2011*
2. *Hejlesen et al., 2015*

… applies a correction

- Computation scheme

1. Solve convection and diffusion

$$\tilde{\mathbf{u}}^{n+1} \qquad \tilde{\omega}^{n+1}$$

2. Compute the correction

$$\delta\omega = \mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right)$$

3. Apply it

$$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$$

- The most frequent ways to compute the correction[1,2,3]:

Need to iterate![2]

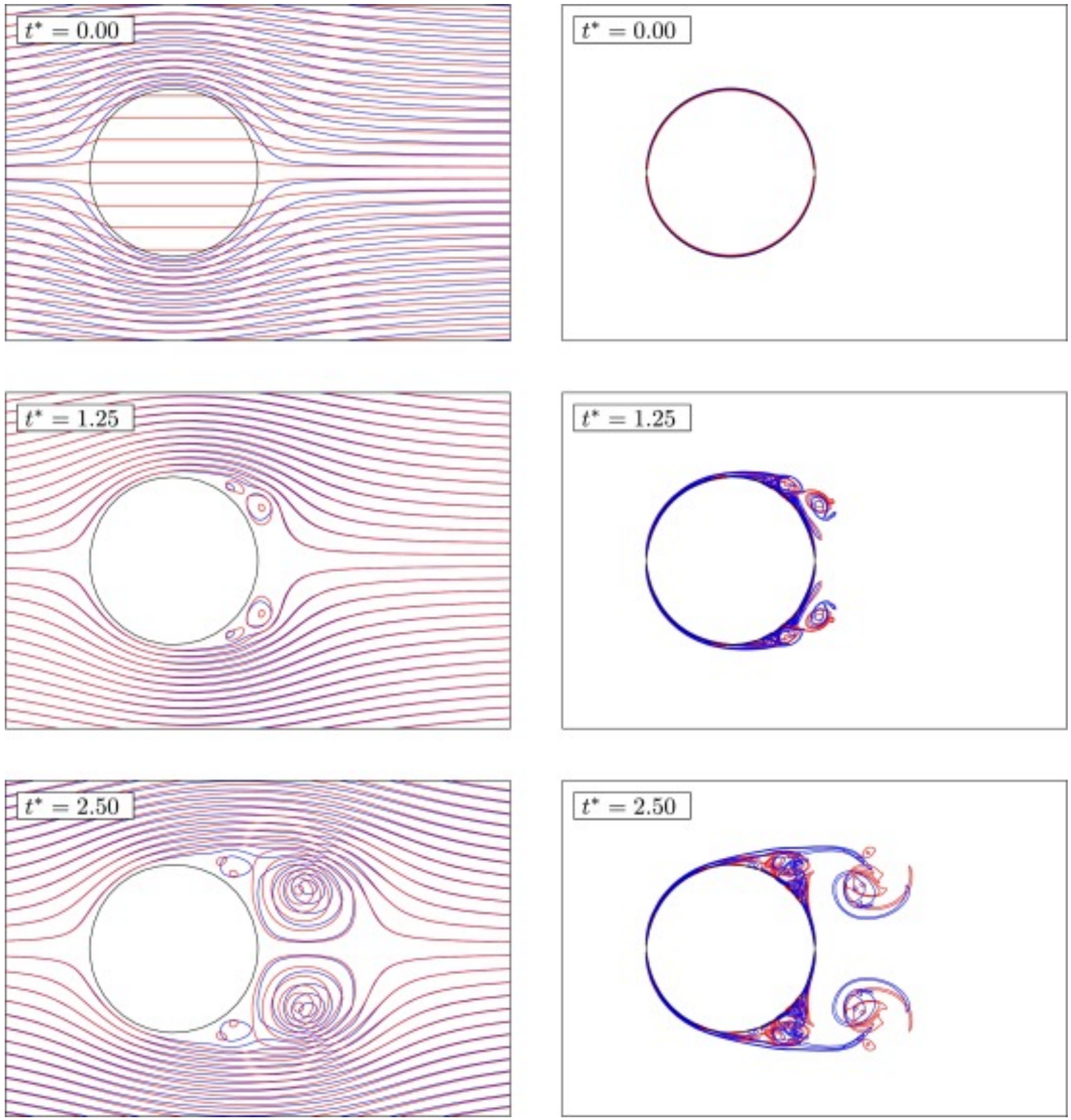
$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\frac{\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t\ \chi\ \mathbf{u}_b}{1 + \lambda\Delta t\ \chi}\right] - \tilde{\omega}^{n+1}$$

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\tilde{\mathbf{u}}^{n}\left(1 + \lambda\Delta t\ \chi\right) + \lambda\Delta t\ \chi\ \mathbf{u}_b\right] - \tilde{\omega}^{n+1}$$

1. *Coquerelle and Cottet, 2008*
2. *Gazzola et al., 2011*
3. *Limet et al., 2015*

1. *Rasmussen et al., 2011*
2. *Hejlesen et al., 2015*

Iterations improve the solution!



Hejlesen et al., 2015

Instead, we solve a system

- We solve the system[3]

$$A\left(\delta\boldsymbol{\omega}\right) = \nabla \times \left[\chi\left(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1}\right)\right]$$

where the linear operator is

$$A : \delta\omega_k \rightarrow \nabla \times \left[\chi\delta\mathbf{u}_k\right] = \nabla \times \left[\chi\left(\left(\nabla^2\right)^{-1}\left(-\nabla \times \delta\boldsymbol{\omega}_k\right)\right)\right]$$

- Drawbacks:
  1. Need to **solve a Poisson equation at each matvec**
  2. A few (or more) **matvec at each time step**
- Jacobi-like approach in the literature[2]
- We use **recycling BiCGStab**[1]

1. *Amritkar et al., 2015*
2. *Hejlesen et al., 2015*
3. *Gillis et al., submitted to JCP, 2016*

Instead, we solve a system

- We solve the system[3]

$$A\left(\delta\boldsymbol{\omega}\right) = \nabla \times \left[\chi\left(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1}\right)\right]$$

where the linear operator is

$$A : \delta\omega_k \rightarrow \nabla \times \left[\chi\delta\mathbf{u}_k\right] = \nabla \times \left[\chi\left(\left(\nabla^2\right)^{-1}\left(-\nabla \times \delta\boldsymbol{\omega}_k\right)\right)\right]$$

- Drawbacks:
  1. Need to **solve a Poisson equation at each matvec**
  2. A few (or more) **matvec at each time step**
- Jacobi-like approach in the literature[2]
- We use **recycling BiCGStab**[1]

1. *Amritkar et al., 2015*
2. *Hejlesen et al., 2015*
3. *Gillis et al., submitted to JCP, 2016*

Recycling previous solutions

Given previous solutions

$$\tilde{C}_{1:l} = \begin{bmatrix} C_{2:l} & b - r_m \end{bmatrix}; \qquad \tilde{U} = \begin{bmatrix} U_{2:l} & x_m \end{bmatrix}$$

$$\tilde{C}_{1:l} = QR$$

$$C_{1:l} = Q \qquad U_{1:l} = UR^{-1}$$

we then satisfy

$$AU = C \qquad \text{and} \qquad C^T C = I$$

Therefore, the optimal guess is

$$x_0 = x_{-1} + UC^T r_{-1} \qquad r_0 = r_{-1} - CC^T r_{-1}$$

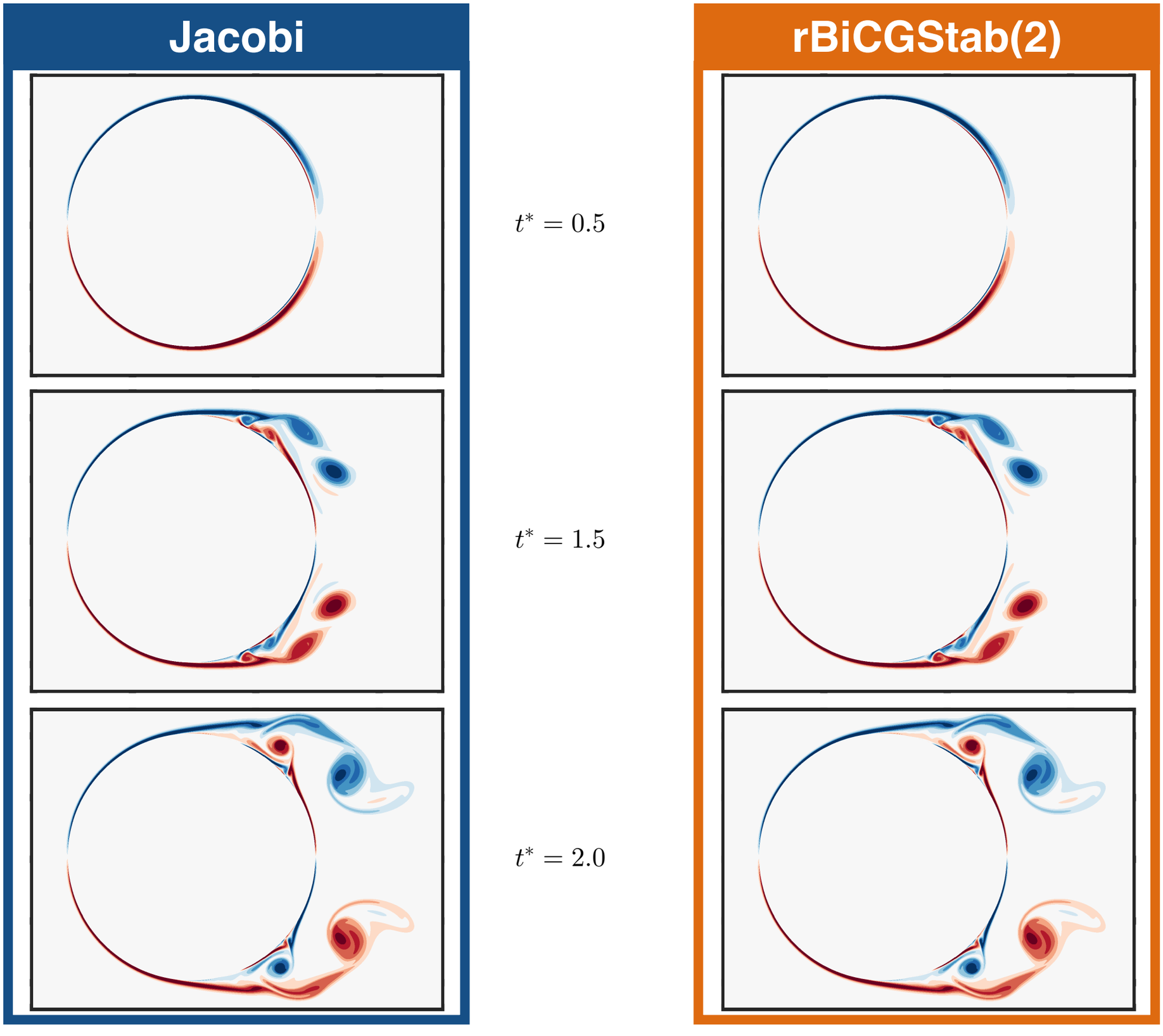
We solve the following linear system

$$A_C \overline{\zeta} = \left[I - CC^T\right] A\overline{\zeta} = \overline{r_0} = r_0$$

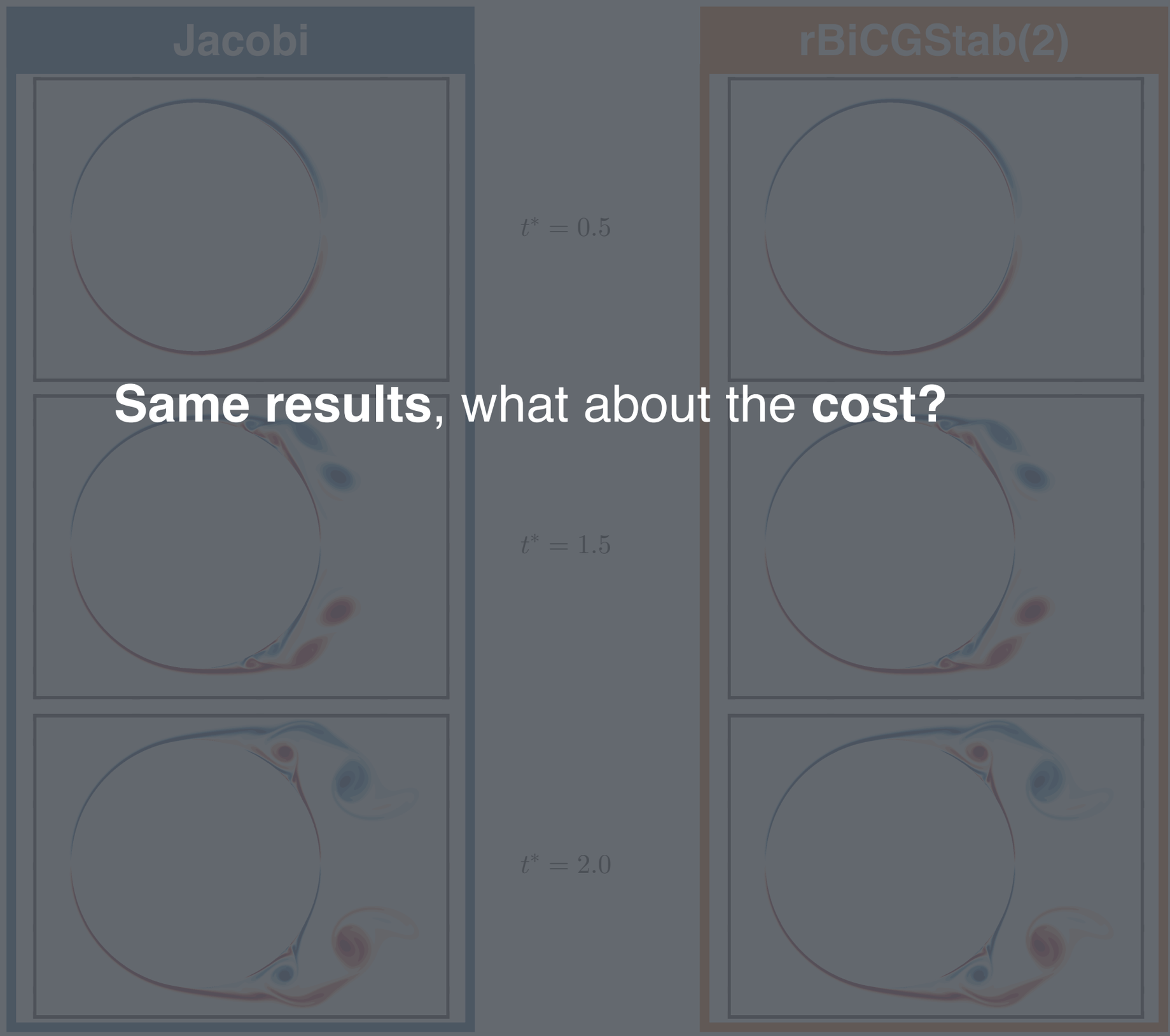
$$x = x_0 + \zeta = x_0 + \left[I - UC^T A\right] \overline{\zeta}$$

1. Parks et al., 2006

Results on a Cylinder - Re_D 9500

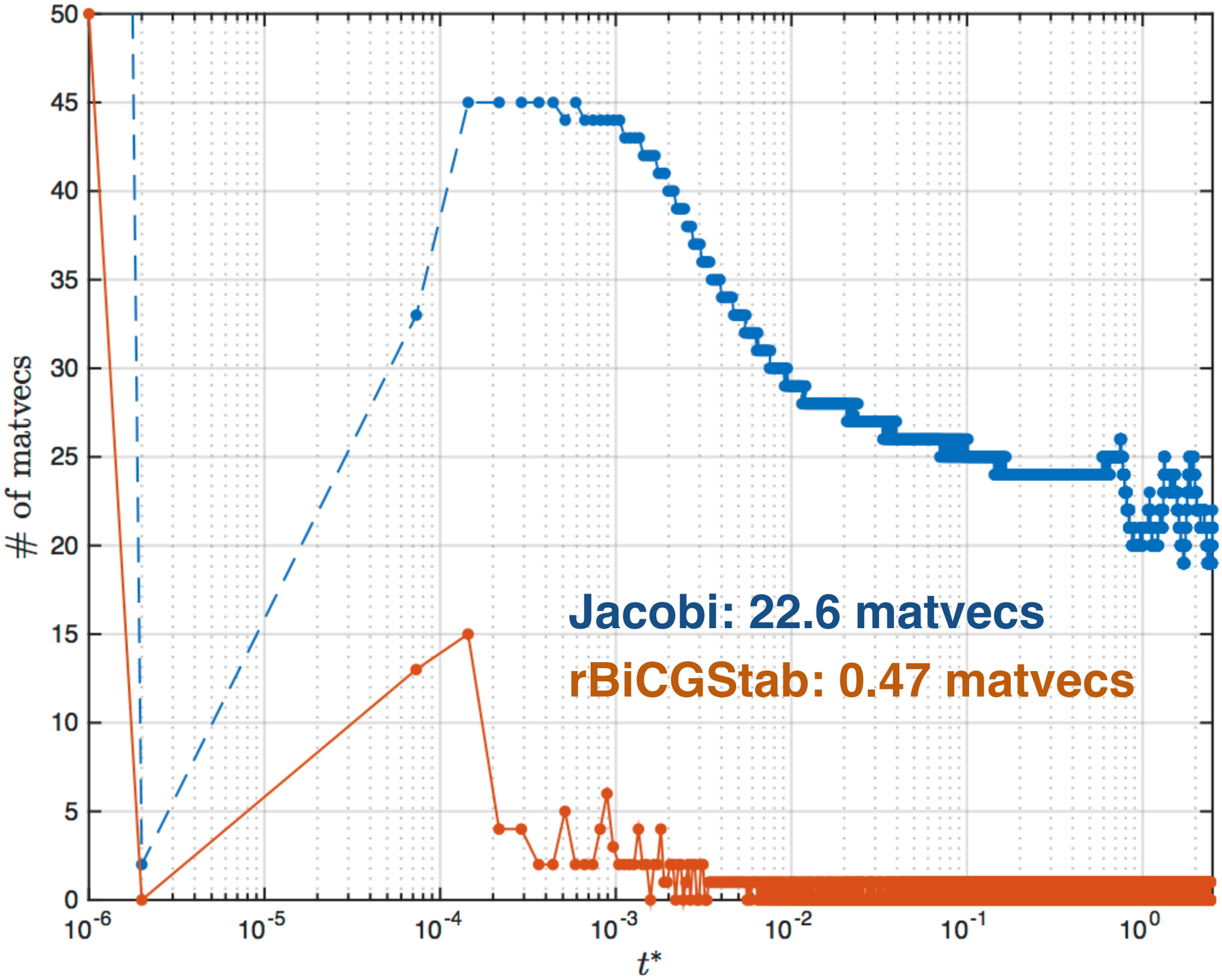


Results on a Cylinder - Re_D 9500

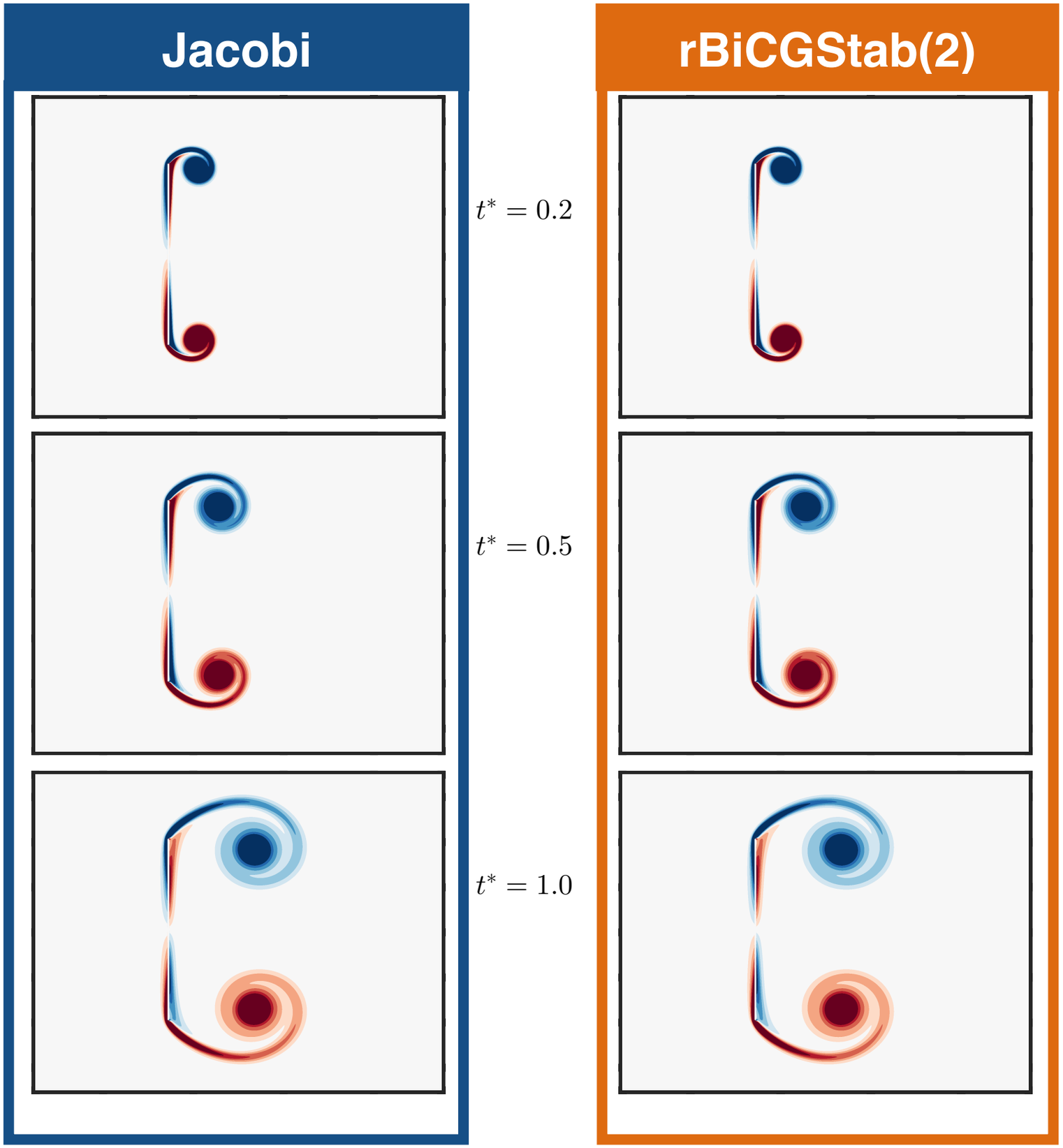


Same results, what about the **cost?**

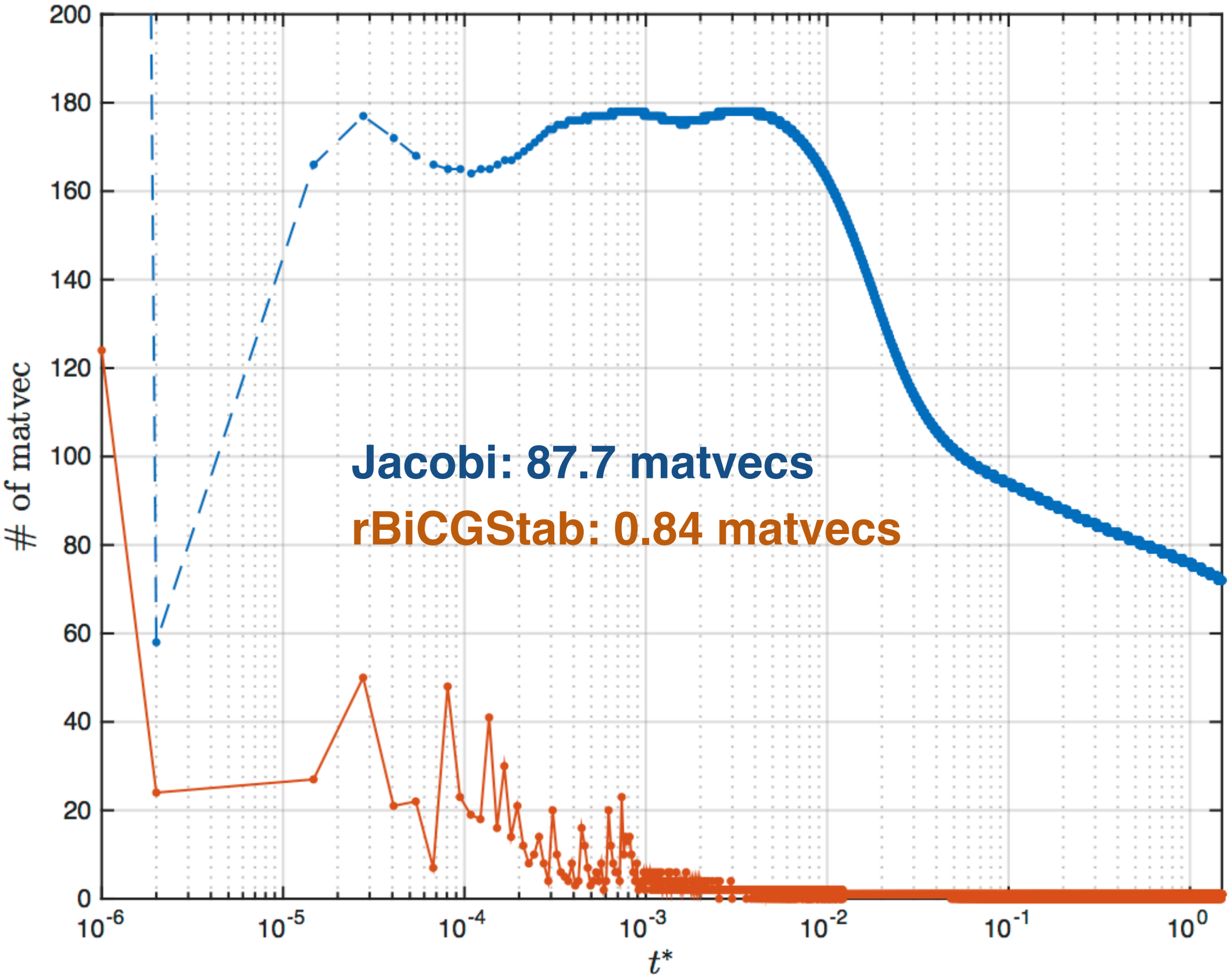
Cost savings: 48 times cheaper!



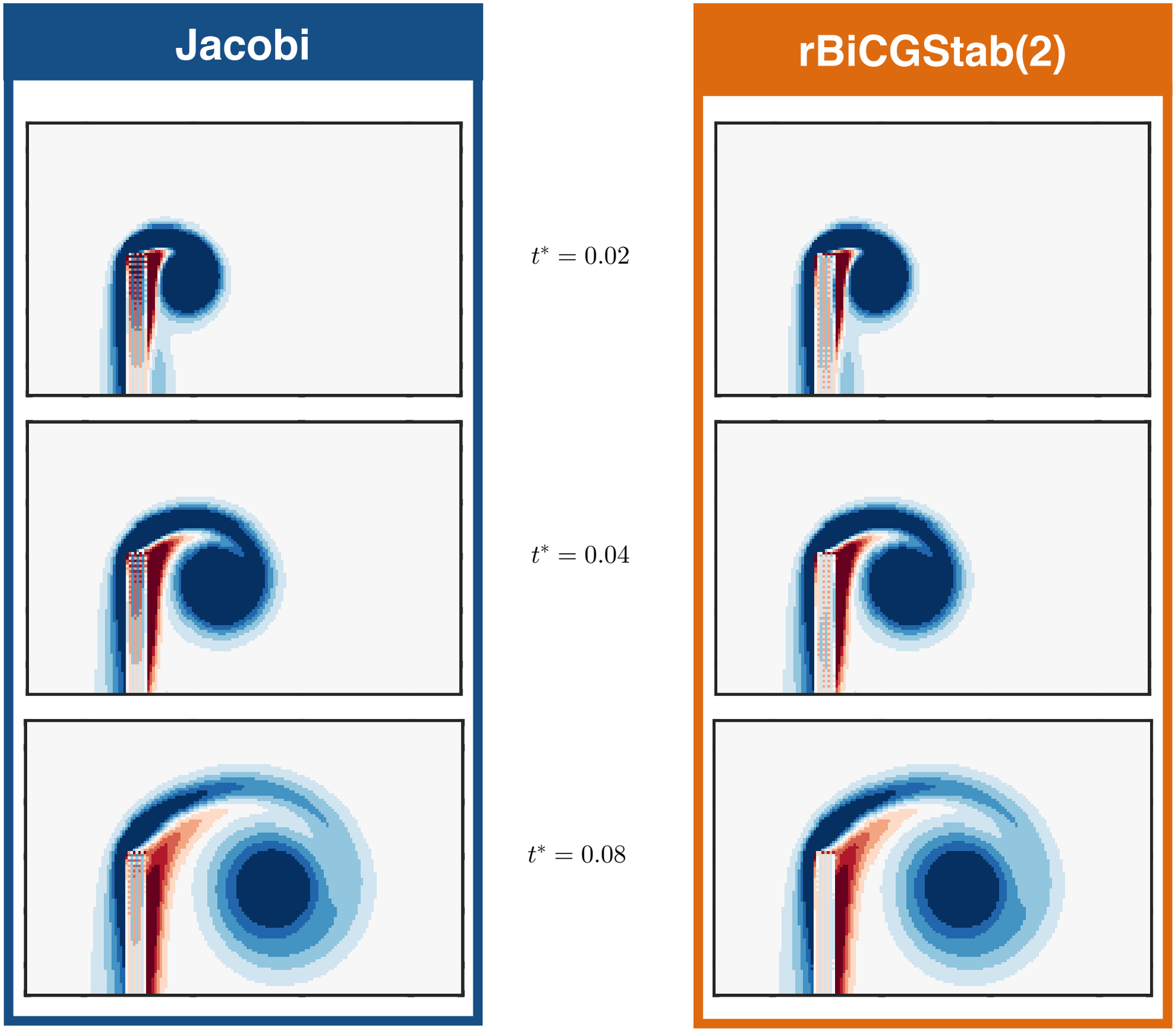
Results on a plate - Re 1000



Cost savings: 104 times cheaper!



Icing on the cake: Spurious vorticity decays



Take-home message

- For the same results quality, using rBiCGStab is up to 100 times cheaper!
- Using previous solution also decreases the number of iterations to perform
- Only two previous solutions are found to be sufficient and keep memory overhead low enough
- What about preconditioning and adaptative size of recycled subspace?

Thank you!

Any question?

T. Gillis, G. Winckelmans and Ph. Chatelain, « *An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows* » , accepted for publication in the Journal of Computational Physics

Acknowledgements
- P. Van Dooren (ICTEAM, Université catholique de Louvain)
- FRIA grant from the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS)
- Tier-1 supercomputer of Federation Wallonie-Bruxelles