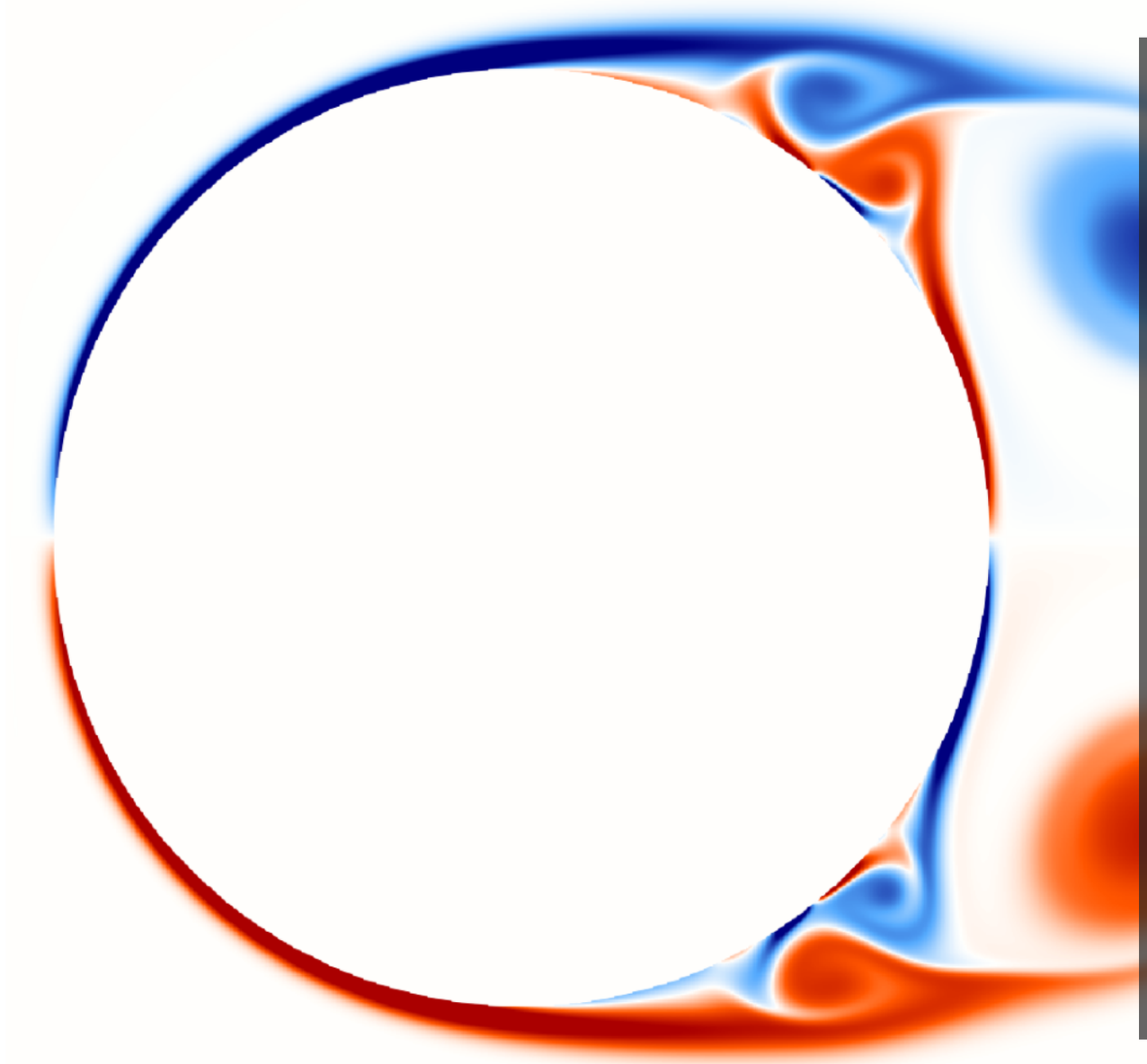


Vortex particle-mesh methods: accurate and efficient handling of solid boundaries

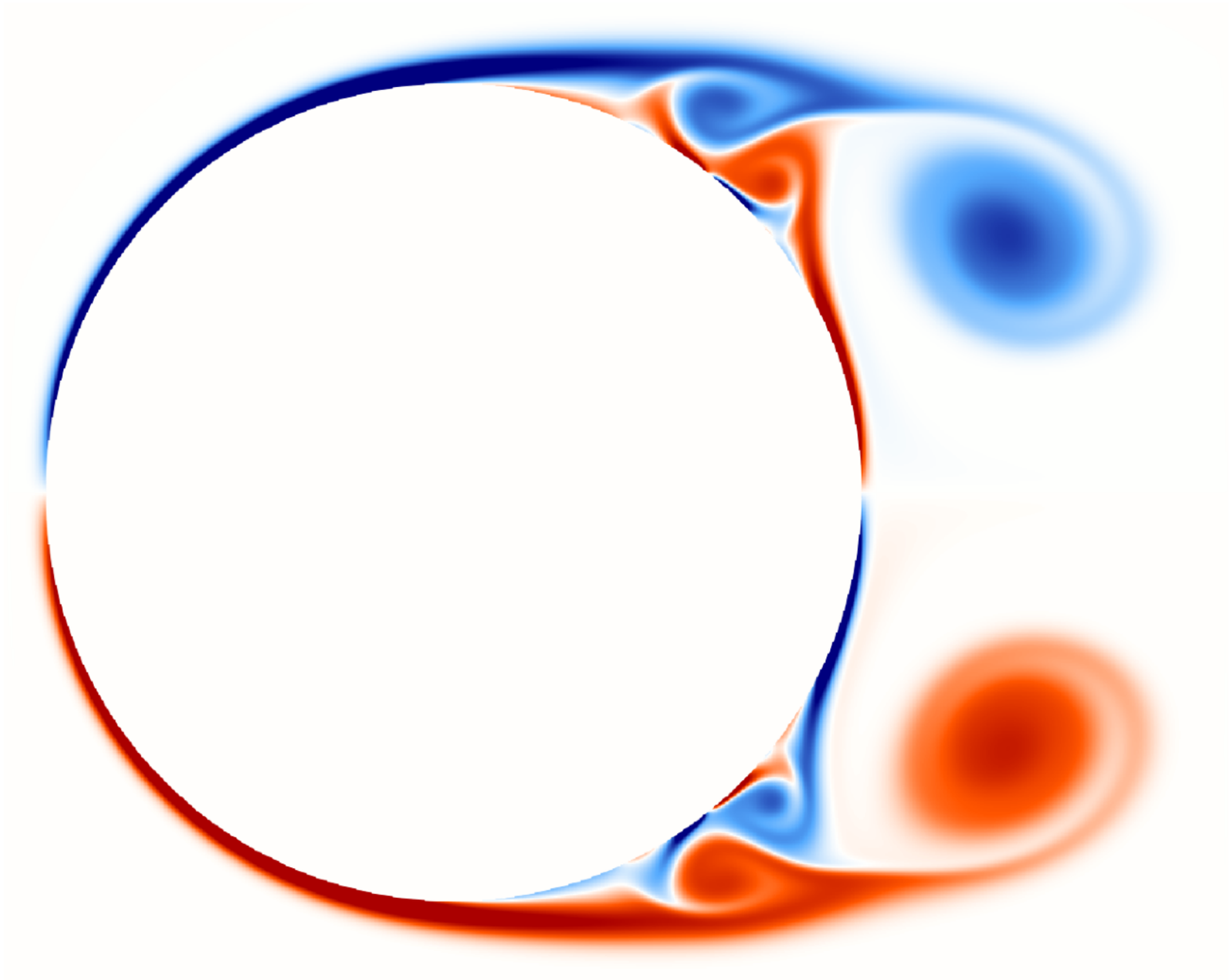
UCL Université catholique de Louvain

iMMC

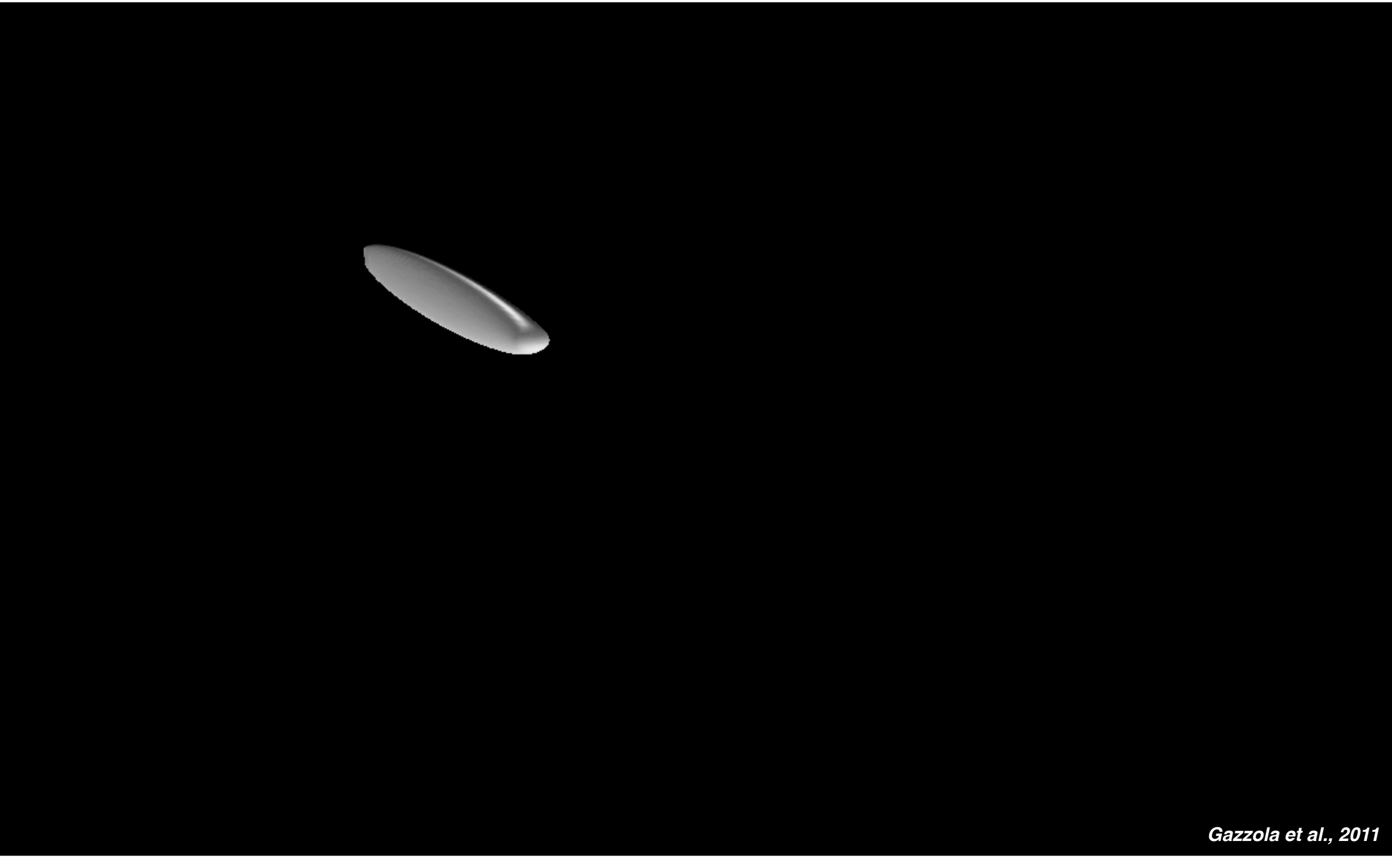
Thomas Gillis, Yves Marichal, Caroline Bernier, Mattia Gazzola, Matthieu Duponcheel, Denis-Gabriel Caprace, Grégoire Winckelmans and Philippe Chatelain

CANUM 2018
Cap d'Agde
May 28

Bluff body flows



Not-so-bluff body flows



Gazzola et al., 2011

Not-so-bluff body flows

The addressed challenge:

How to take complex geometries into account?

- Accelerating flows
- Low errors
- Low cost
- Complex scenarios

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]

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

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

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

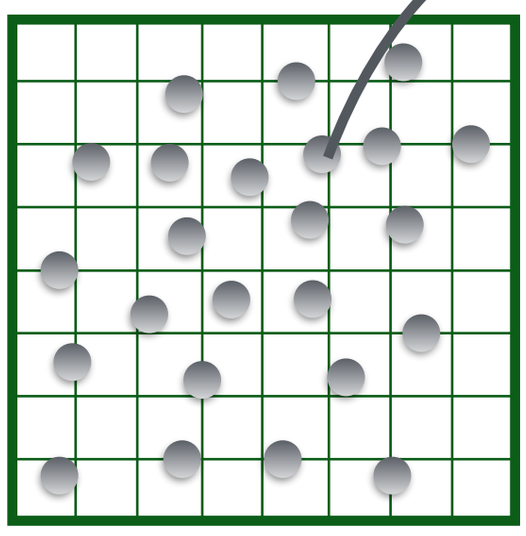
- Vorticity formulation of the incompressible Navier-Stokes equations[1,2]

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

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

Boundaries

- Helmholtz decomposition of velocity field

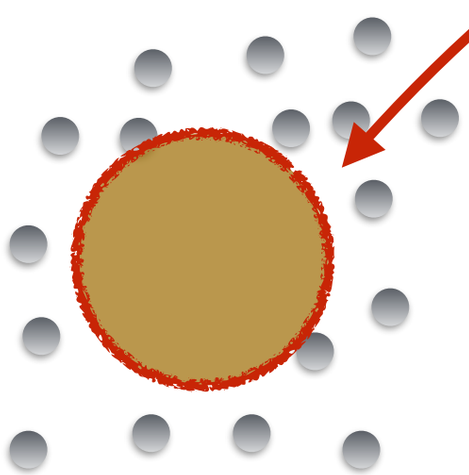
$$\mathbf{u} = \nabla\varphi + \nabla \times \boldsymbol{\psi}$$

- Boundaries -> Generalized Helmholtz decomposition[1]

$$\mathbf{u}(\mathbf{x}) = \int_\Omega \nabla G(\mathbf{x}-\mathbf{x}') \times \boldsymbol{\omega}\, d\mathbf{x}' + \int_\Omega \nabla G(\mathbf{x}-\mathbf{x}')\sigma\, d\mathbf{x}'$$
$$+ \int_{\partial\Omega} \nabla G(\mathbf{x}-\mathbf{x}') \times (\mathbf{n}\times\mathbf{u})\, d\mathbf{x}' + \int_{\partial\Omega} \nabla G(\mathbf{x}-\mathbf{x}')(\mathbf{n}\cdot\mathbf{u})\, d\mathbf{x}'$$

Vortex sheet

- Integral equation -> "panel"[1,2] or "sources"[3] methods
  - Lighthill's model of vorticity generation : Velocity jump provides flux of vorticity over a time increment



1. *Ploumhans, Winckelmans, 2002*
2. *Chatelain, 2004*
3. *Poncet, 2009*

The Brinkman penalization...

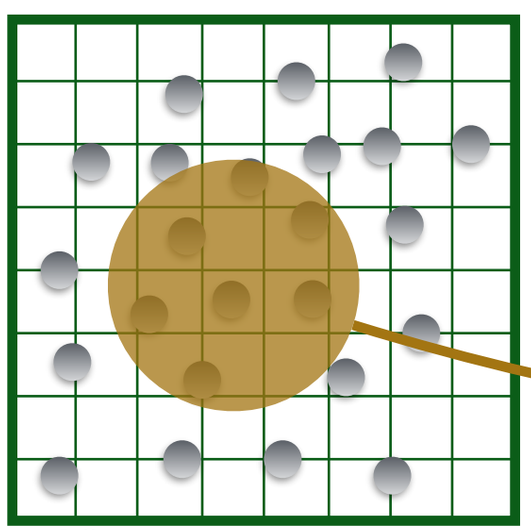
- Adds a **constraint** on the velocity[1,2,3]

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

where the mask function is given by

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

- 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

Drawbacks:

1. generate substantial **diffusion errors**[1]
2. misses **elliptic character** of 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., 2017*

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]

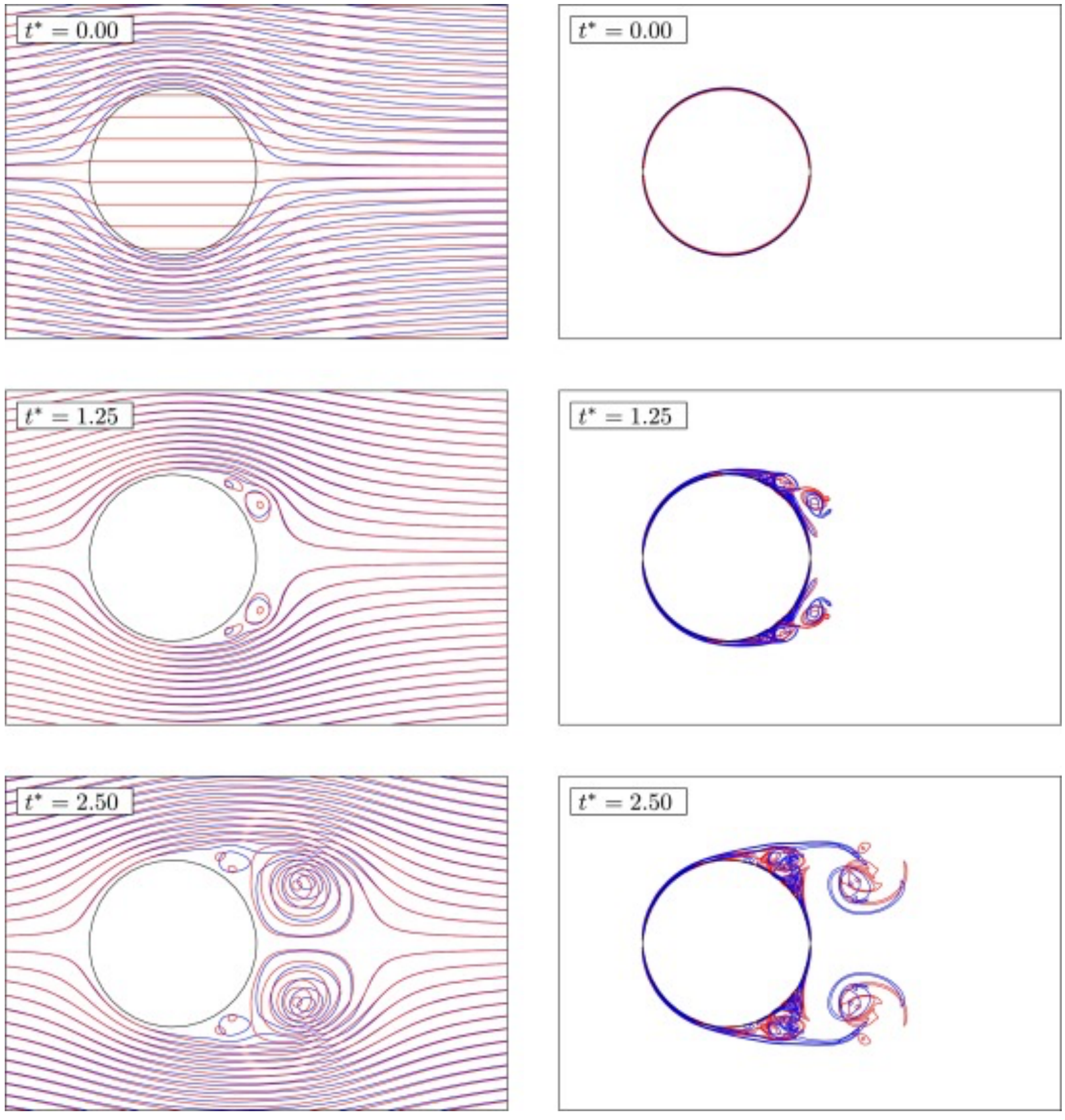
$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla \times \left[\frac{\tilde{\mathbf{u}}^{n+1}\left(1+\lambda\Delta t\ \chi\right) - \lambda\Delta t\ \chi\ \mathbf{u}_b}{1+\lambda\Delta t\ \chi}\right] - \tilde{\omega}^{n+1}$$

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

1. *Coquerelle and Cottet, 2008*
2. *Gazzola et al., 2011*
3. *Limeida et al., 2012*

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., JCP, 2017*

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., JCP, 2017*

Recycling iterative linear solver: Ax=b

Given previous solutions, we build U and C to satisfy

$$AU = C \qquad C^T C = I$$

$[\ x_1\ x_2\ \dots\ x_l]$ $[\ b_1-r_1\ b_2-r_2\ \dots\ b_l-r_l]$

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

Recycling iterative linear solver: Ax=b

Given previous solutions, we build U and C to satisfy

$$AU = C \qquad C^T C = I$$

[x_1 x_2 … x_l] [b_1-r_1 b_2-r_2 … b_l-r_l]

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

Orthogonalized linear system

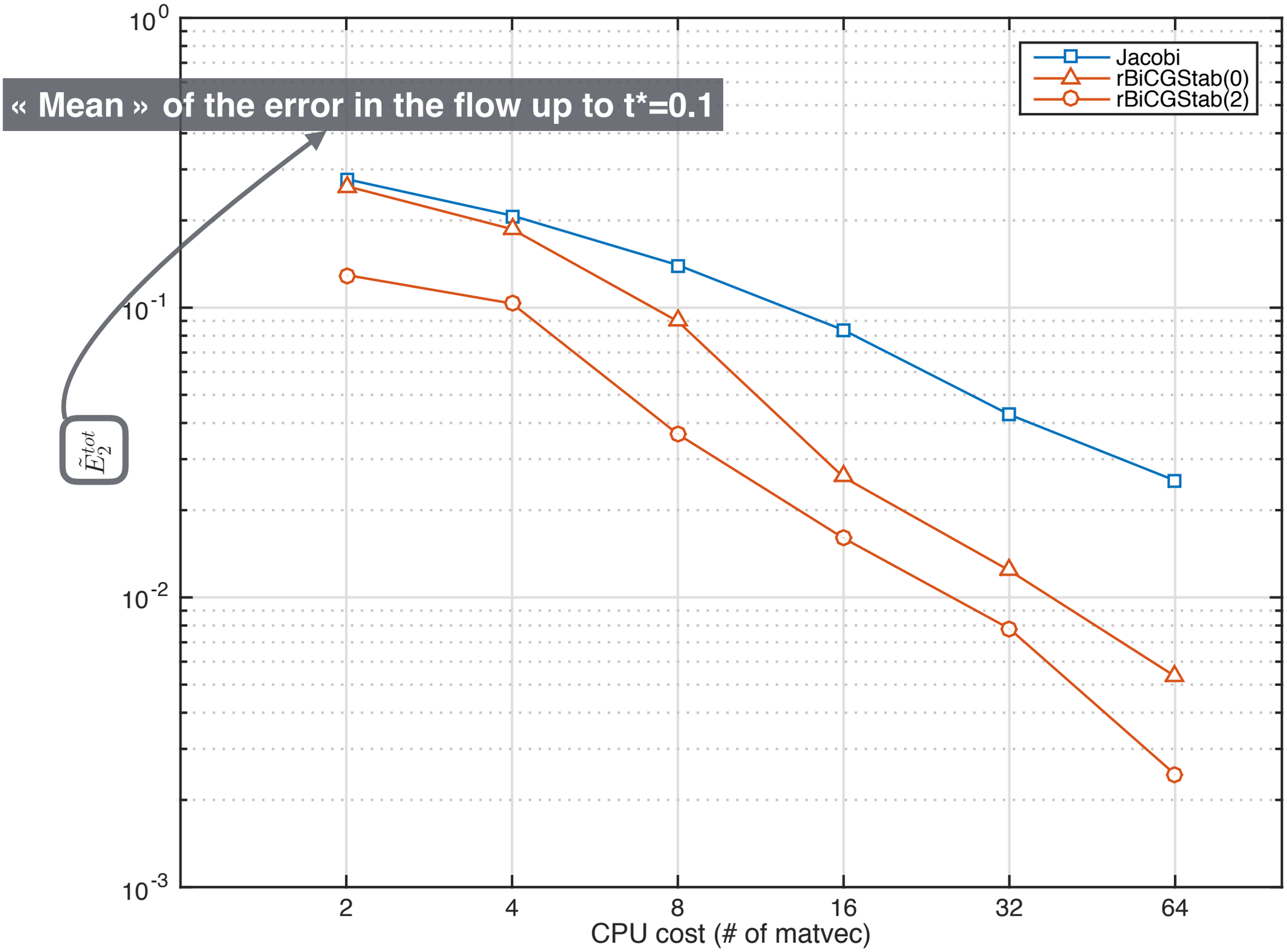
$$A_C \overline{\zeta} = \left[I - CC^T\right] A\overline{\zeta} = \overline{r_0} = r_0$$

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

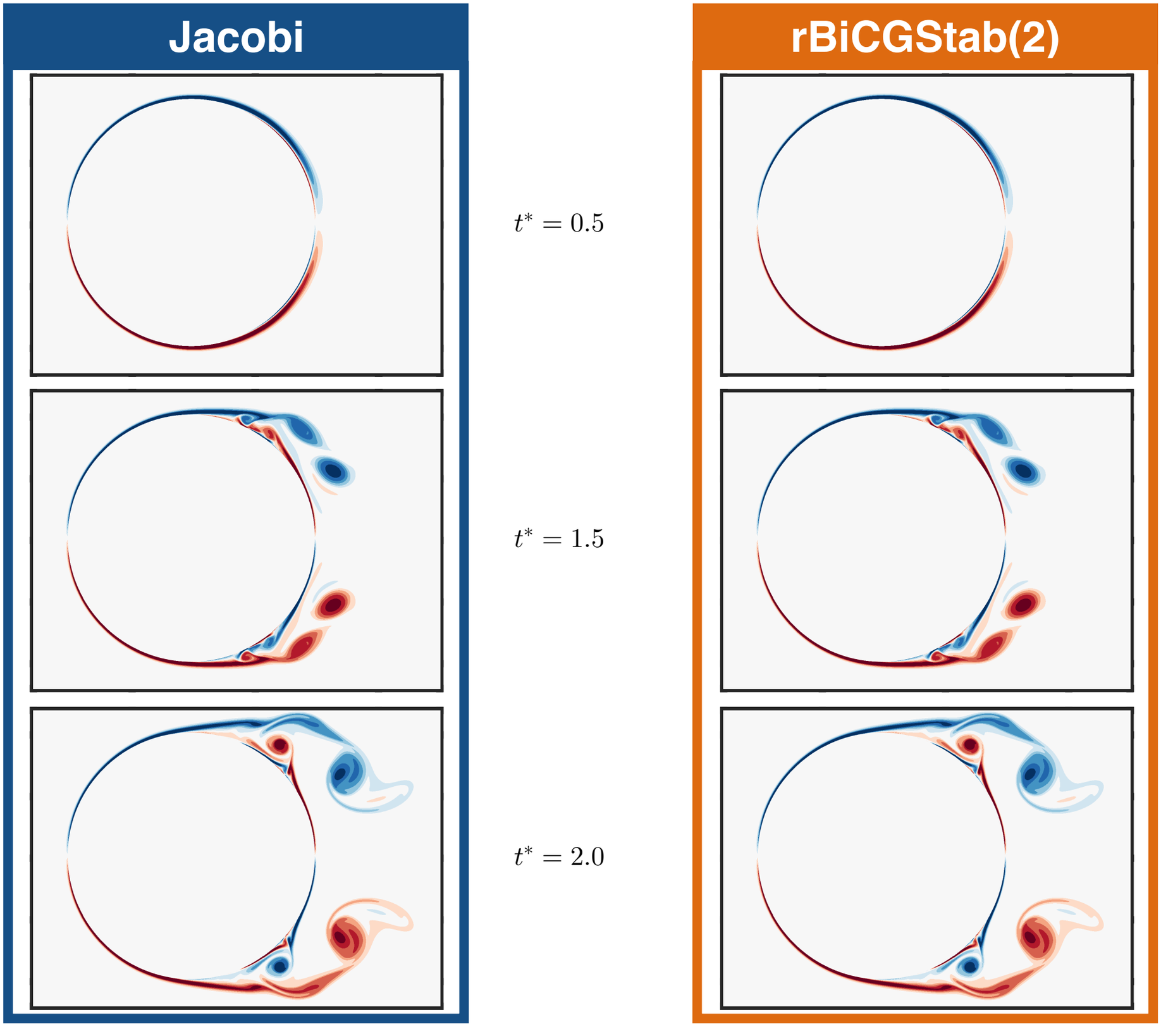
1. Parks et al., 2006

... and use previous solutions

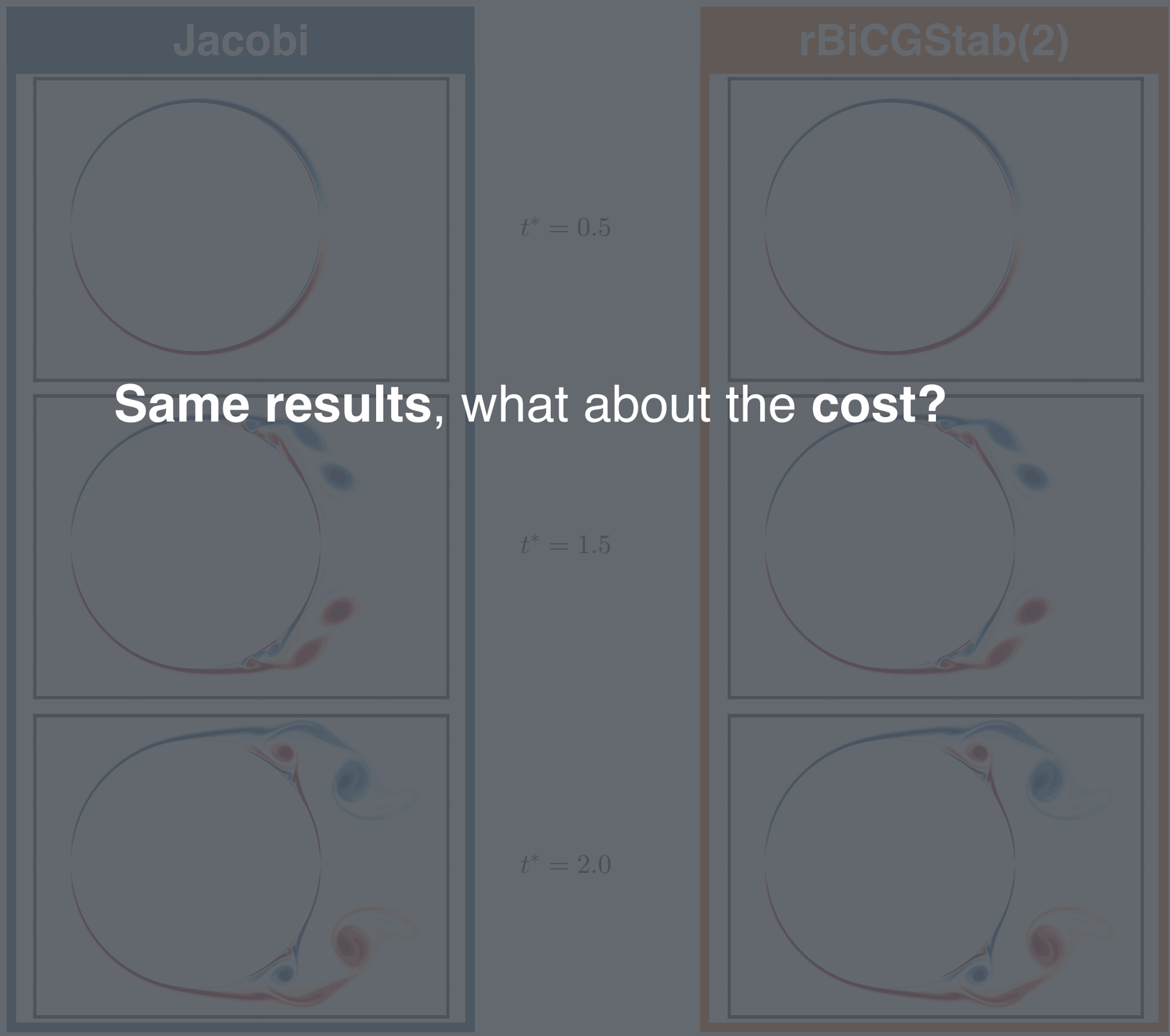
Therefore, we use recycling BiCGStab



Results on a Cylinder - Re_D 9500

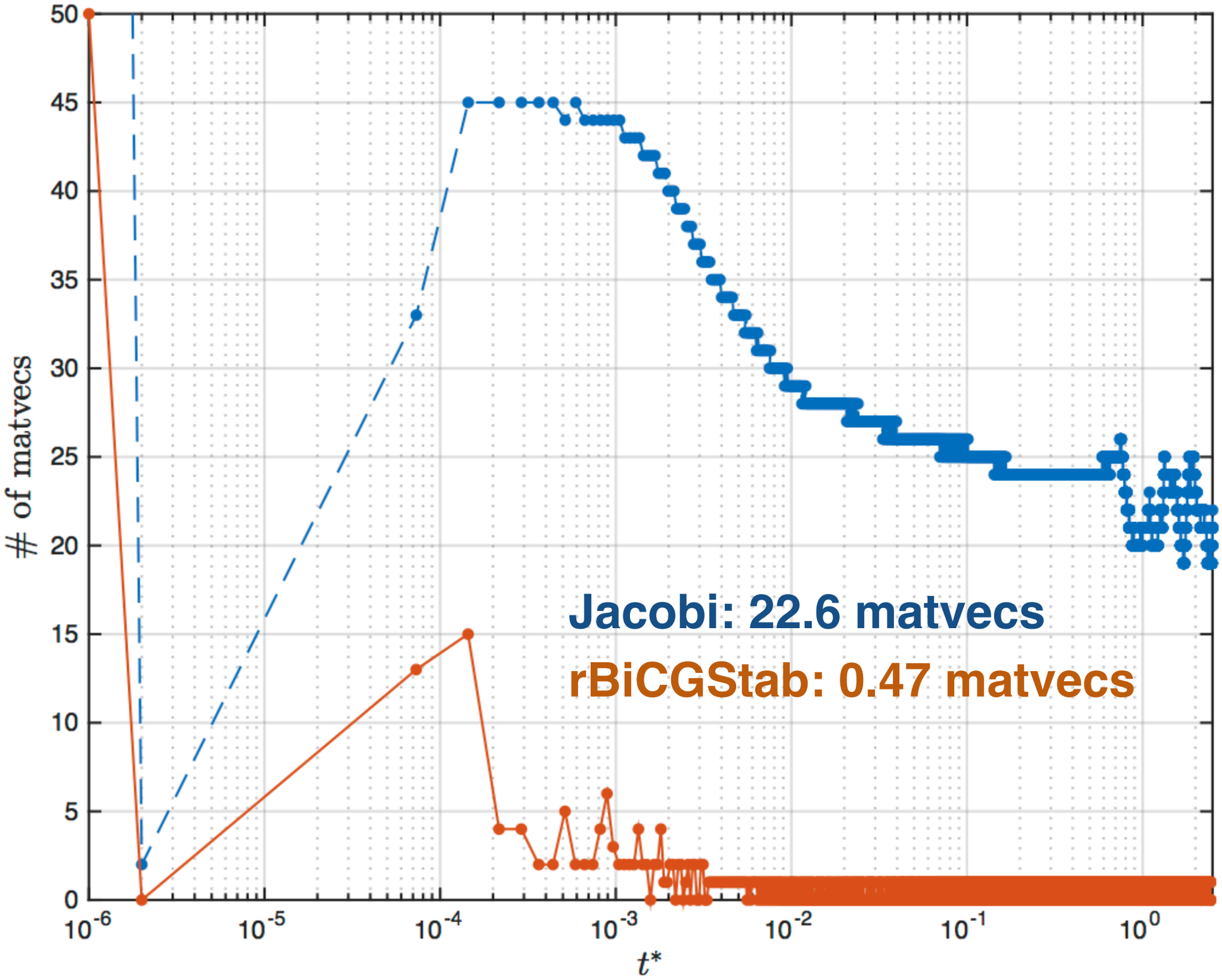


Results on a Cylinder - Re_D 9500

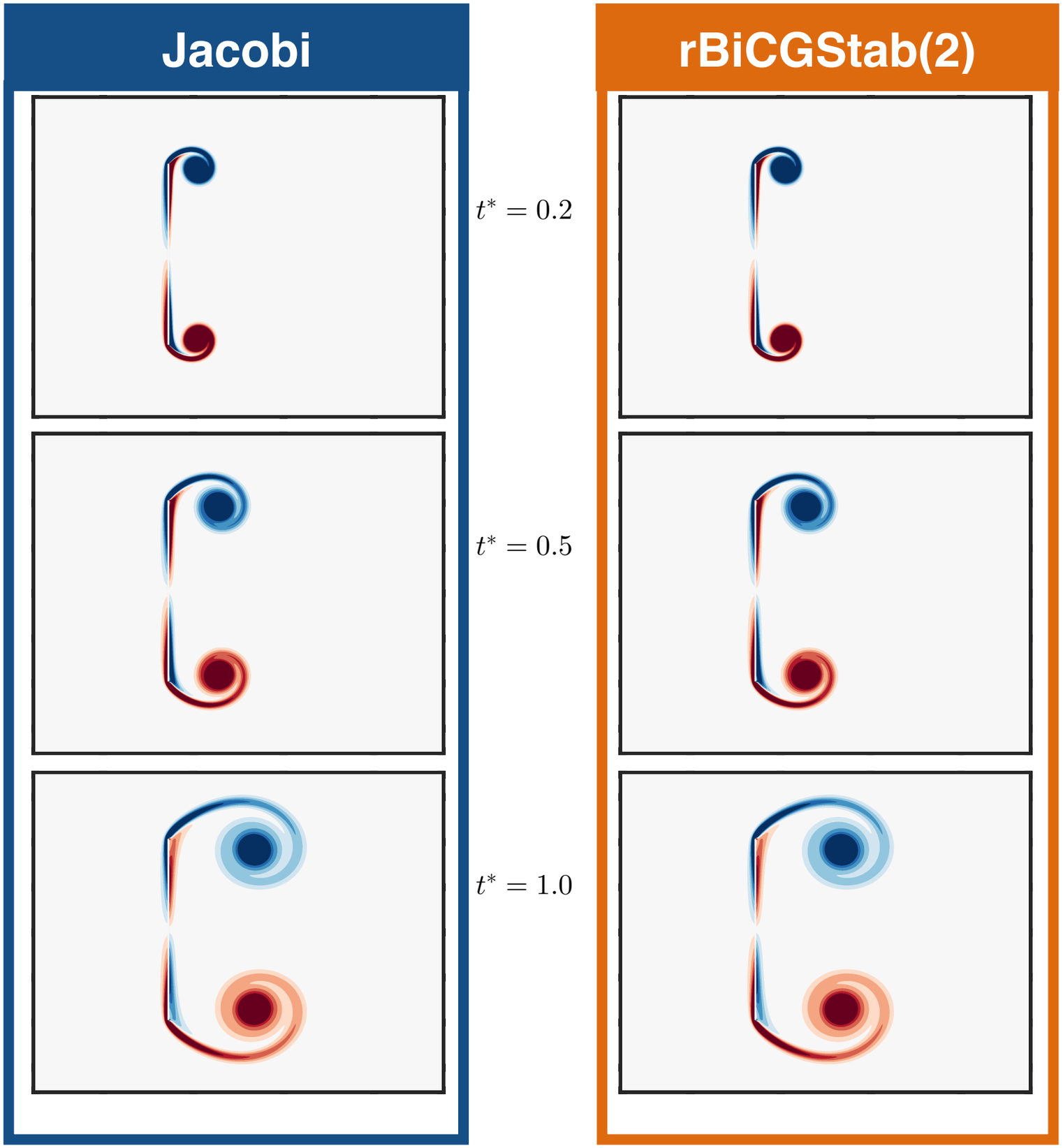


Same results, what about the **cost?**

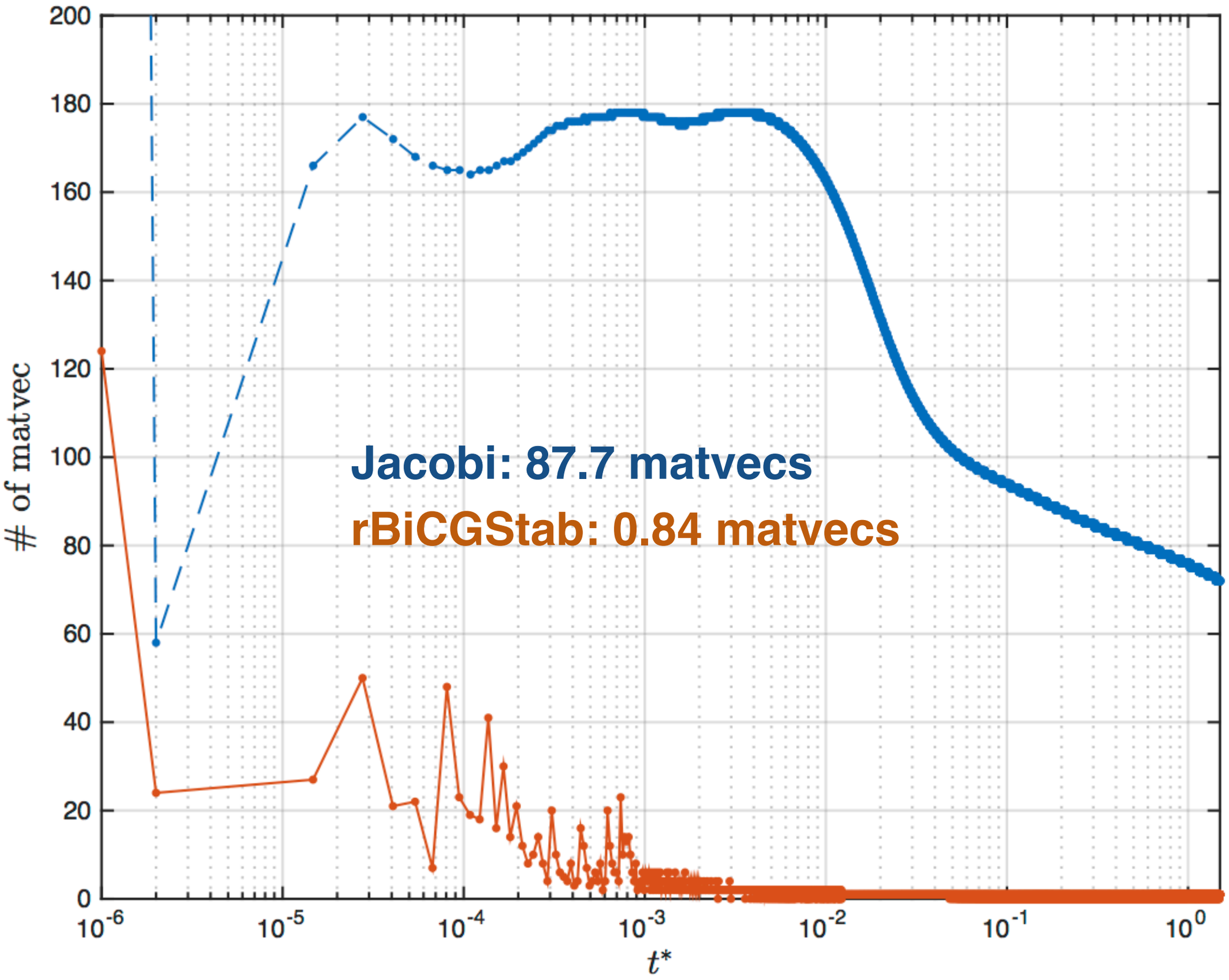
Cost savings: 48 times cheaper!



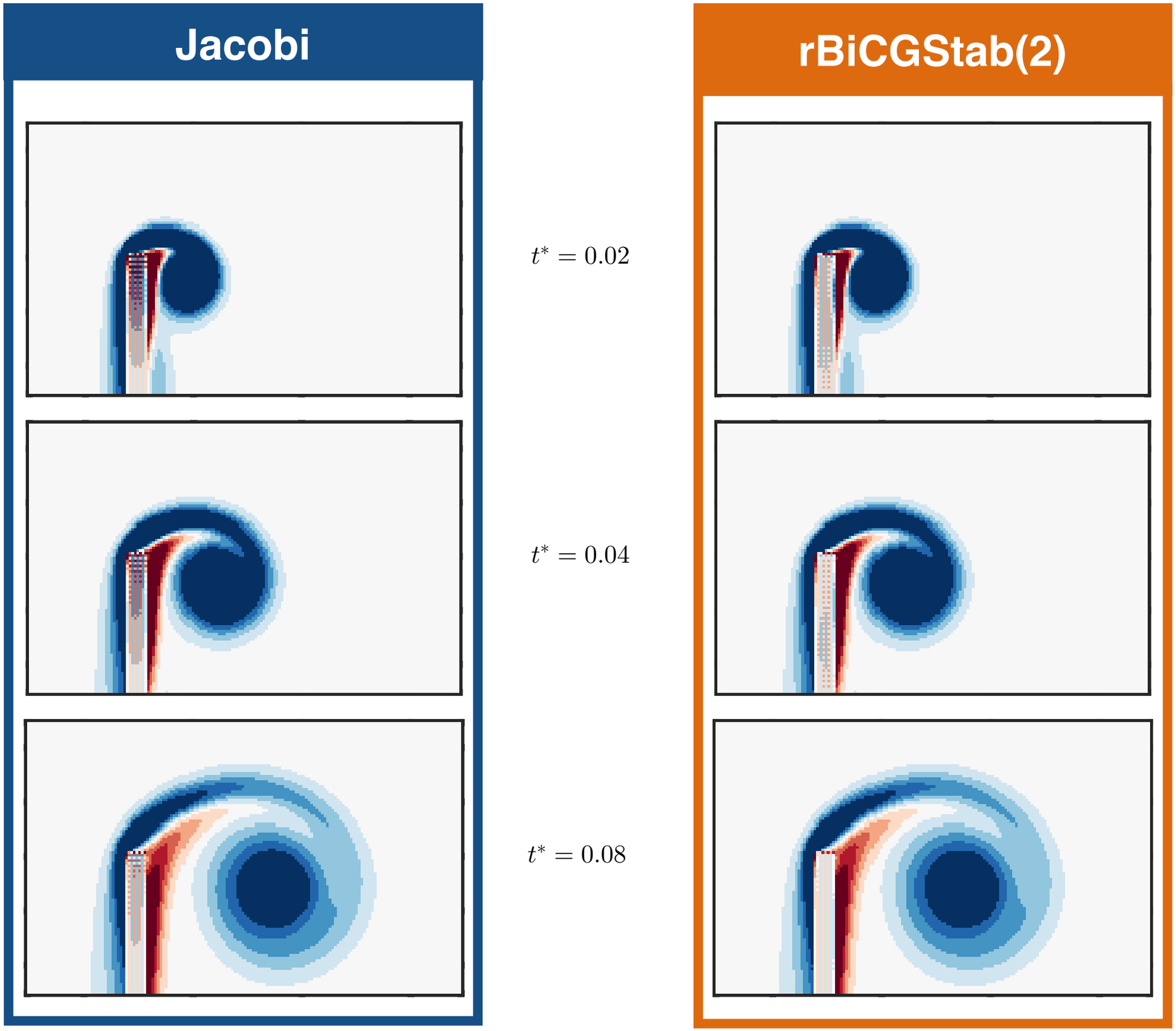
Results on a plate - Re_D 1000



Cost savings: 104 times cheaper!



Icing on the cake: Spurious vorticity reduced

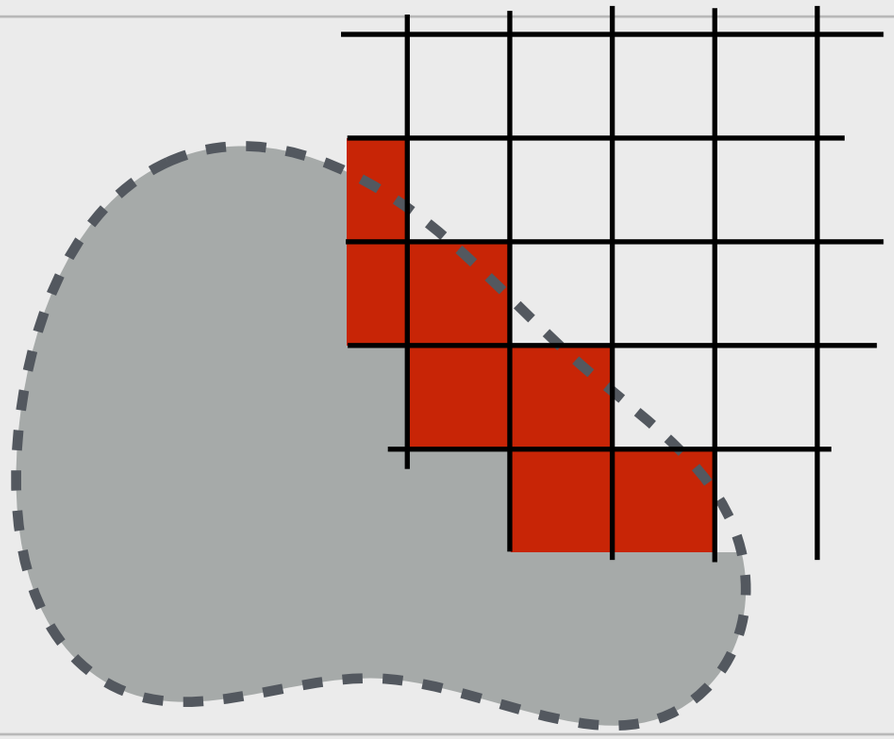


Take-home messages (1)

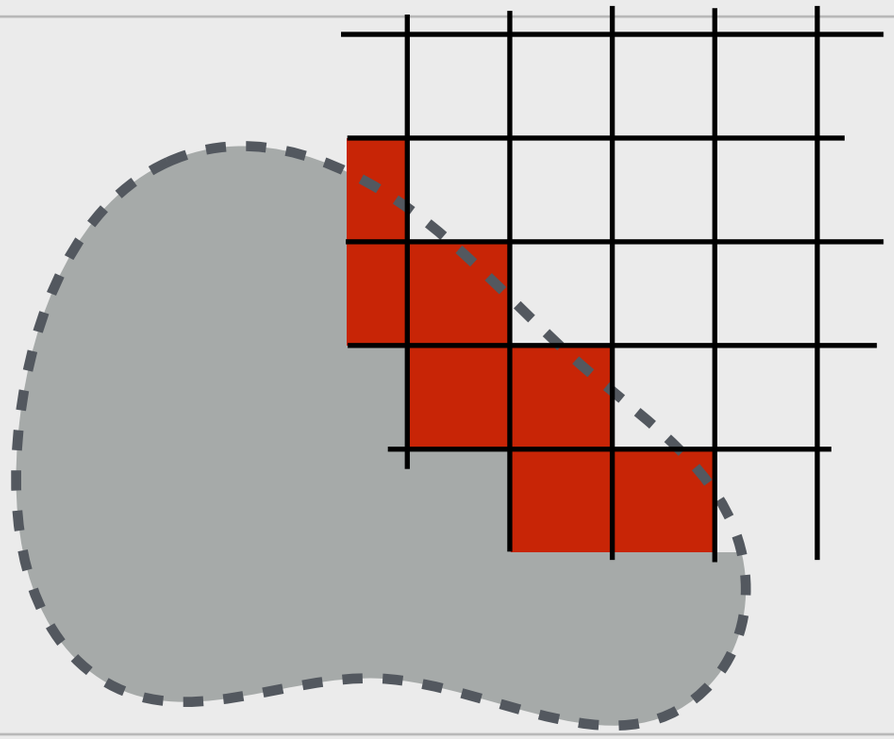
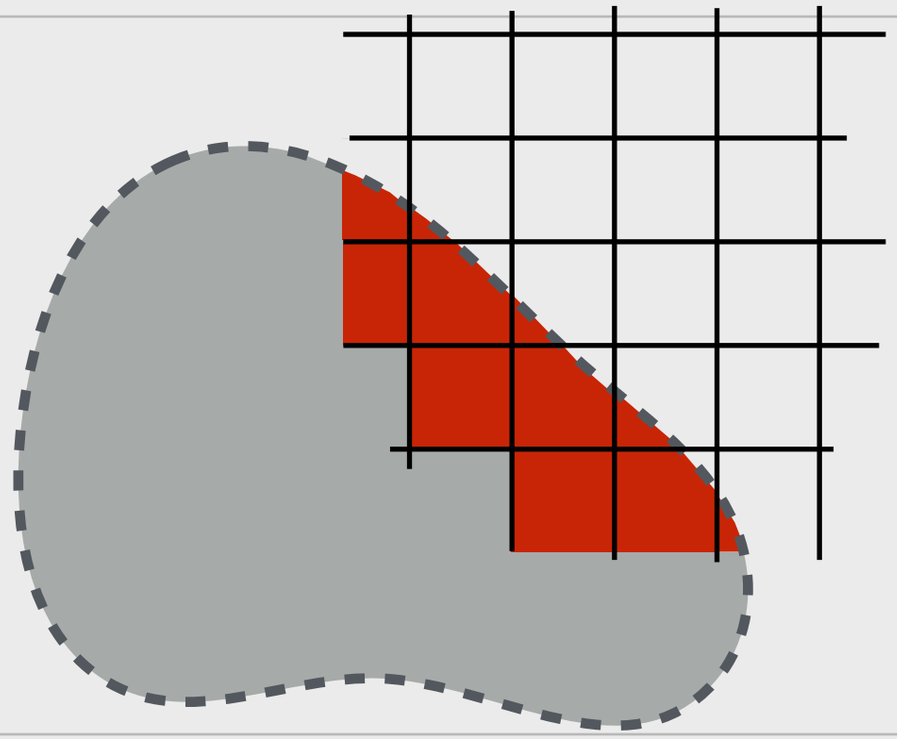
- For the same results quality, using rBiCGStab is up to 100 times cheaper!
- Using previous solution also decreases the number of iterations to perform
- Only two previous solutions are found to be sufficient and keep memory overhead low enough
- What about preconditioning and adaptative size of recycled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

- but ...

Immersed Interfaces with VPM

Iterative Penalization Vortex Particle-Mesh Method (IPenVPM)	Immersed Interface Vortex Particle-Mesh Method (IIVPM)
• Moderately intrusive • Low accuracy	
	
Efficient Iterative Penalization, Gillis et al., JCP (347), 2017	

Immersed Interfaces with VPM

Iterative Penalization Vortex Particle-Mesh Method (IPenVPM)	Immersed Interface Vortex Particle-Mesh Method (IIVPM)
• Moderately intrusive • Low accuracy	• Highly intrusive • High accuracy
	
Efficient Iterative Penalization, Gillis et al., JCP (347), 2017	Imm Interf solver for Potential flows, Marichal et al., C&F, 2014 Imm Interf particle-mesh interpolation, Marichal et al., JCP, 2016 Fast Imm Interf Poisson solver, Gillis et al., JCP, 2018

Immersed Interfaces with VPM

Iterative Penalization Vortex Particle-Mesh Method (IPenVPM)

- Moderately intrusive
- Low accuracy



Efficient Iterative Penalization, Gillis et al., JCP (347), 2017

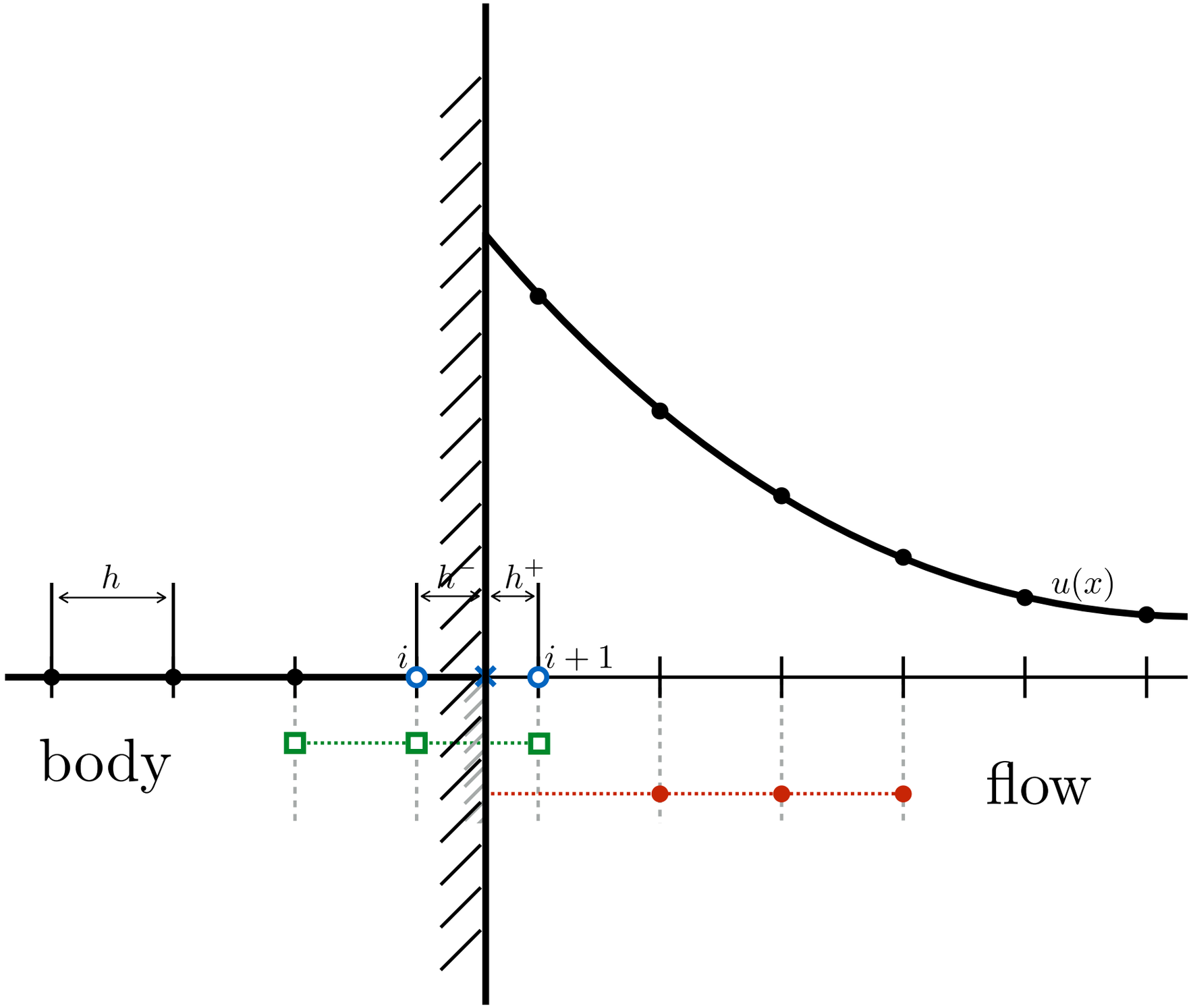
Immersed Interface Vortex Particle-Mesh Method (IIVPM)

- Highly intrusive
- High accuracy



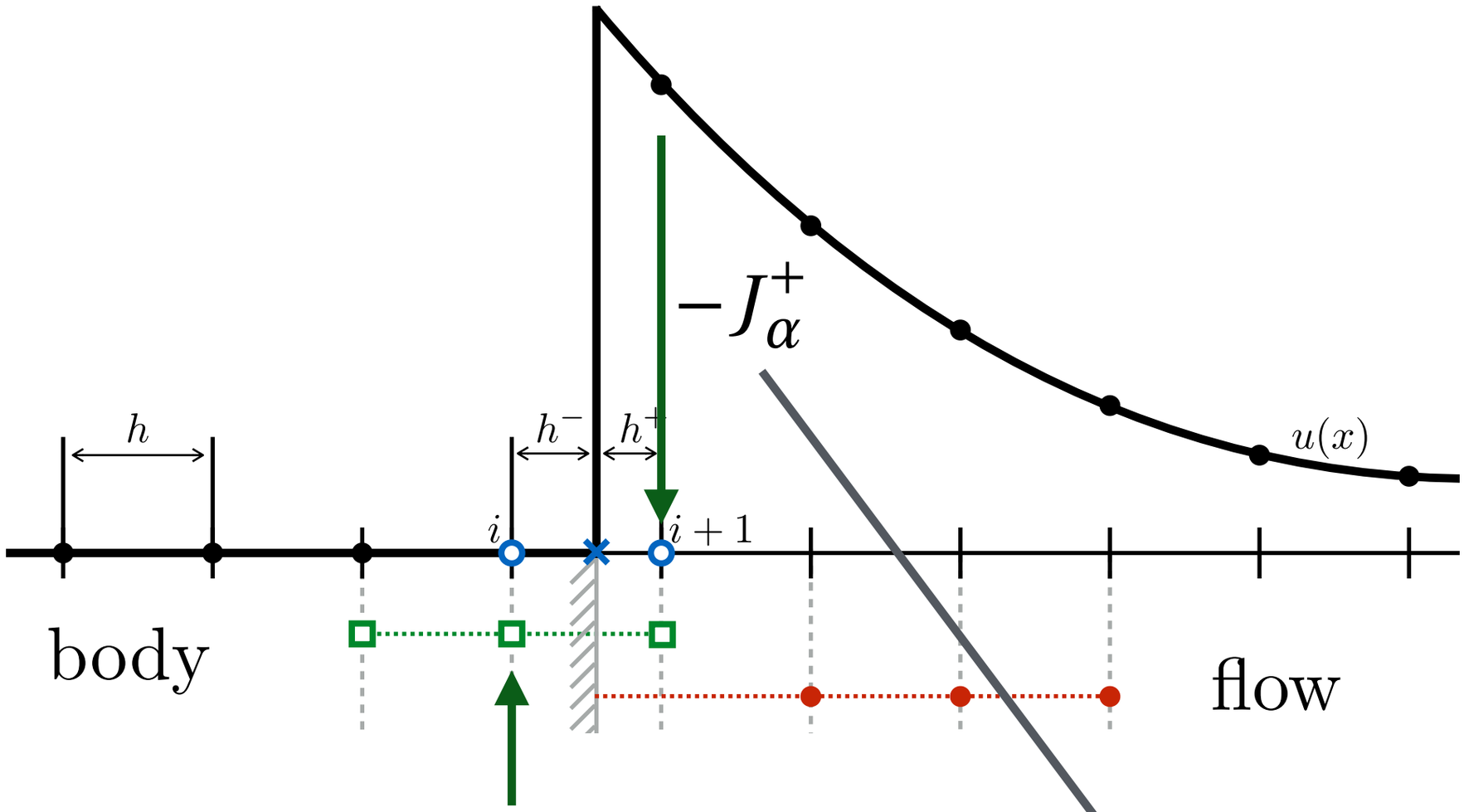
Imm Interf potential flow solver, Marichal et al., C&F, 2014
Imm Interf particle-mesh interp, Marichal et al., JCP, 2016
Fast Imm Interf Poisson solver, Gillis et al., JCP, 2018

The immersed interface meth. (IIM)



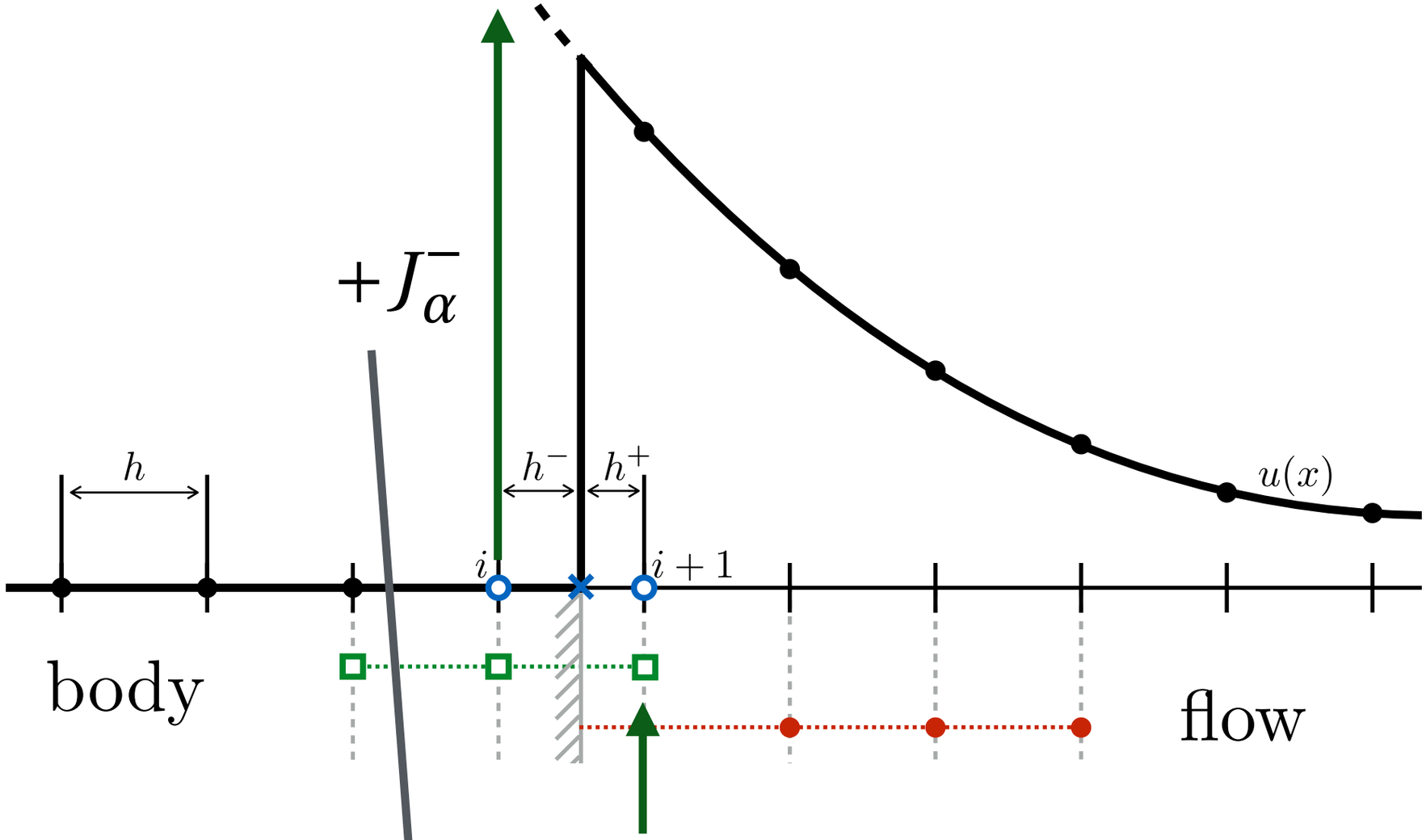
1. Leveque and Li (1995)
2. Marichal et al. (2014)

IIM for Poisson equation



$$u^{(2)}(x_i) = \frac{1}{h^2}u(x_{i-1}) - \frac{2}{h^2}u(x_i) + \frac{1}{h^2}\left[u(x_{i+1}) - J_\alpha^+\right] + \mathcal{O}(h^2)$$

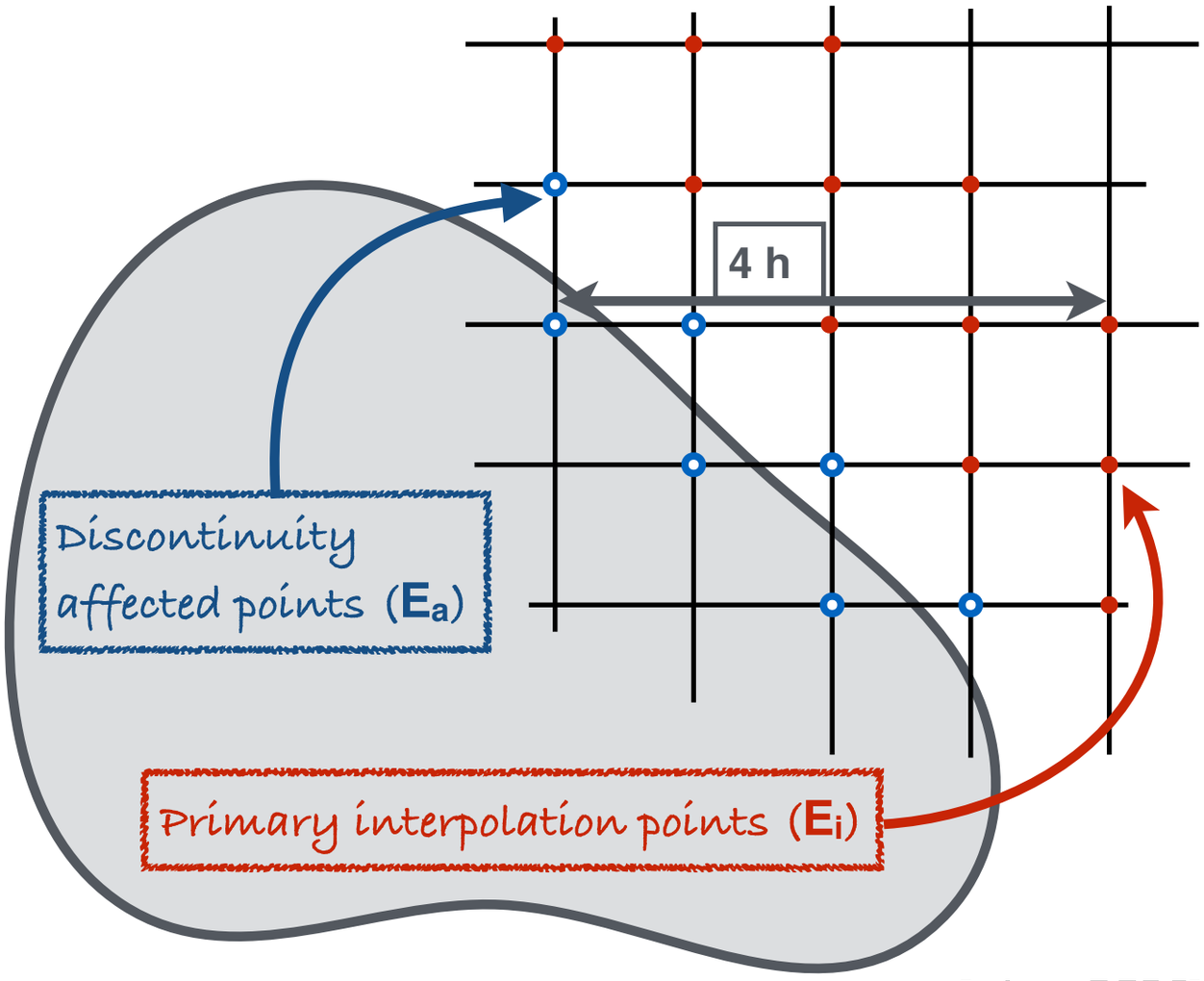
IIM for Poisson equation



$$u^{(2)}(x_i) = \frac{1}{h^2}u(x_{i-1}) - \frac{2}{h^2}u(x_i) + \frac{1}{h^2}\left[u(x_{i+1}) - J_\alpha^+\right] + \mathcal{O}\left(h^2\right)$$

$$u^{(2)}(x_{i+1}) = \frac{1}{h^2}\left[u(x_i) + J_\alpha^-\right] - \frac{2}{h^2}u(x_{i+1}) + \frac{1}{h^2}u(x_{i+2}) + \mathcal{O}\left(h^2\right)$$

IIM from an algebra point of view



IIM is a **lower rank modification to the original operator**

We use **Sherman-Morrison-Woodbury**[1] **(SMW)** formula to invert the operator

$$[A+UV]^{-1} = A^{-1} - A^{-1}U\left[I+VA^{-1}U\right]^{-1}VA^{-1}$$

$$\left(\nabla_h^2 + E_a C E_i^T\right)^{-1} = \nabla_h^{-2}\left\{I - E_a C\left[I + E_i^T\nabla_h^{-2}E_a C\right]^{-1}E_i^T\nabla_h^{-2}\right\}$$

1. *Chatelin et Poncet, 2014*

Sherman-Morrison-Woodbury

$$\left(\nabla_h^2 + E_a C E_i^T\right)^{-1} = \nabla_h^{-2}\left\{I - E_a C\left[I + E_i^T \nabla_h^{-2} E_a C\right]^{-1} E_i^T \nabla_h^{-2}\right\}$$

Already existing unbounded

« Body-free » Poisson solver

- FFT based: Chatelain et al.,...
- FMM based: Colonius et al.,...

Sherman-Morrison-Woodbury

$$\left(\nabla_h^2 + E_a C E_i^T\right)^{-1} = \nabla_h^{-2}\left\{I - E_a C\left[I + E_i^T \nabla_h^{-2} E_a C\right]^{-1} E_i^T \nabla_h^{-2}\right\}$$

Already existing unbounded

« Body-free » Poisson solver

- FFT based: Chatelain et al.,...
- FMM based: Colonius et al.,...

Linear system to solve

- on the interpolation points
- using rGMRes or rBiCGStab

Sherman-Morrison-Woodbury

$$\left(\nabla_h^2 + E_a C E_i^T\right)^{-1} = \nabla_h^{-2}\left\{I - E_a C\left[I + E_i^T \nabla_h^{-2} E_a C\right]^{-1} E_i^T \nabla_h^{-2}\right\}$$

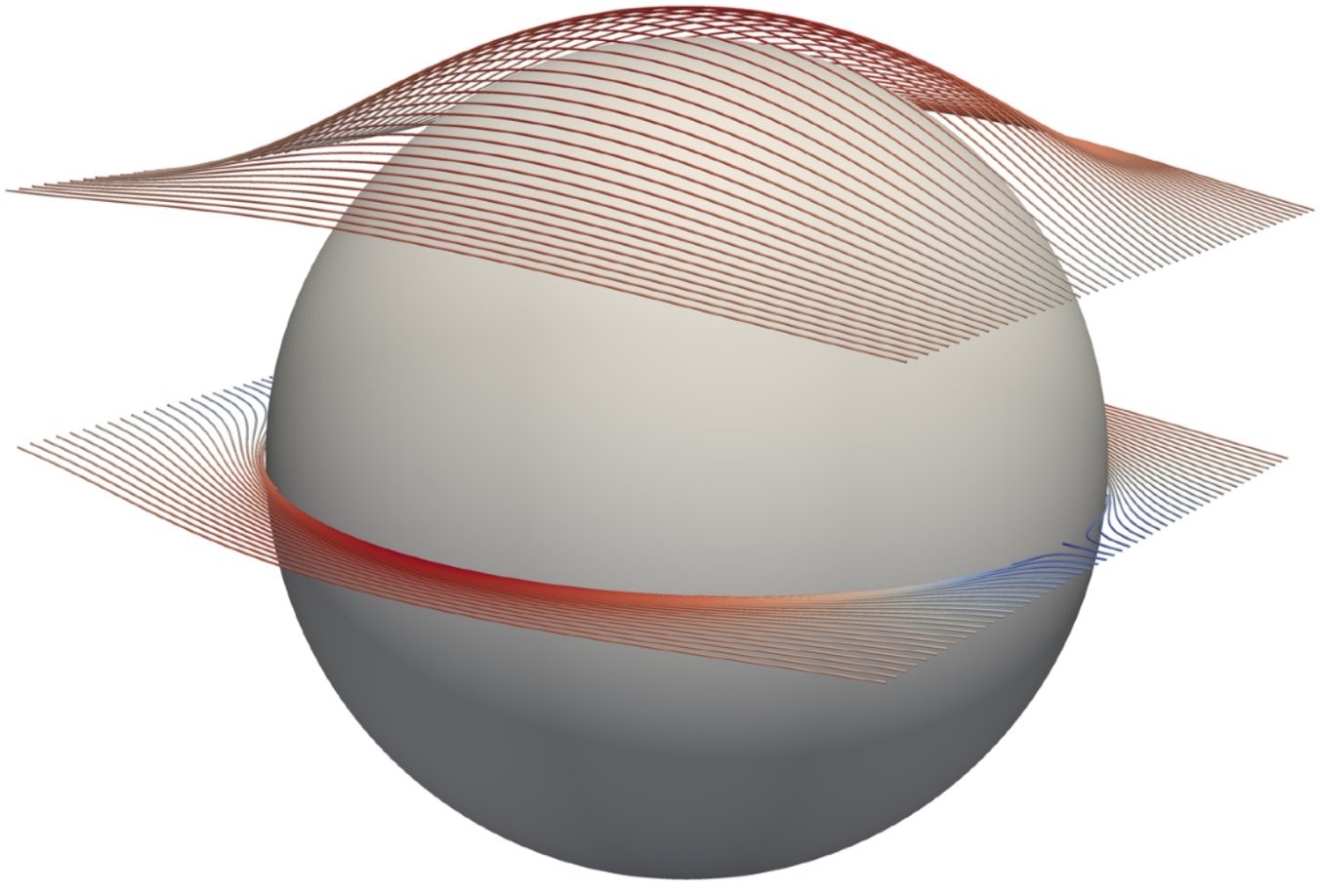
Already existing « Body-free » Poisson solver

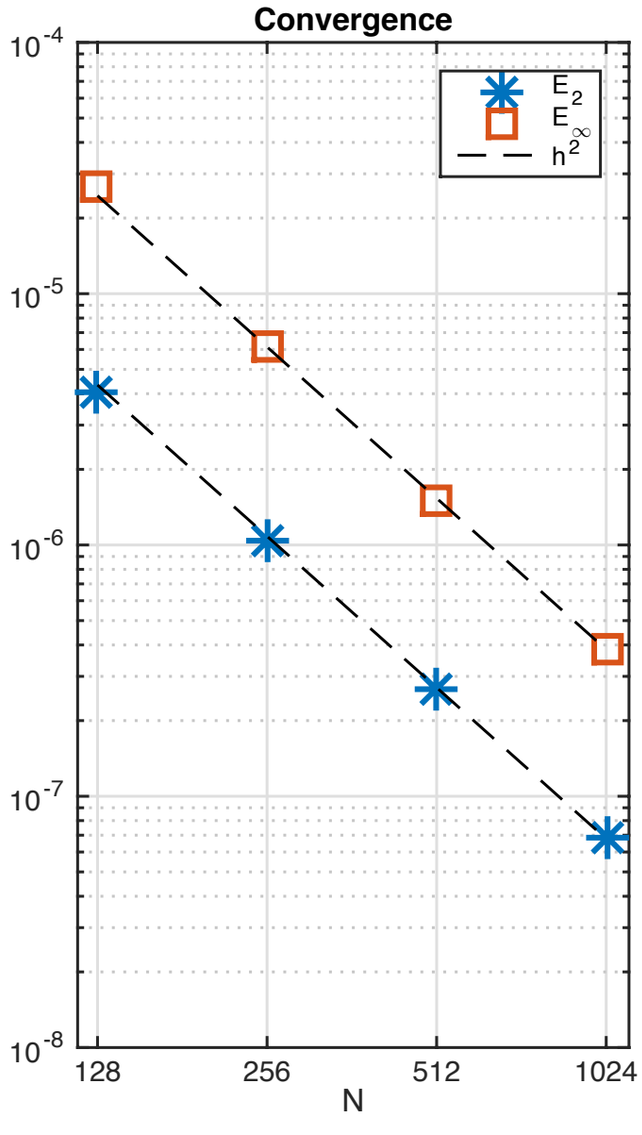
- FFT based: Chatelain et al.,...
- FMM based: Colonius et al.,...

Linear system to solve

- on the interpolation points
- using rGMRes or rBiCGStab

Validation on potential flow past a sphere





Can again use recycling solver

Given previous solutions, we build U and C to satisfy

$$AU = C \qquad C^T C = I$$

[x_1 x_2 … x_l] [b_1-r_1 b_2-r_2 … b_l-r_l]

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

Orthogonalized 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

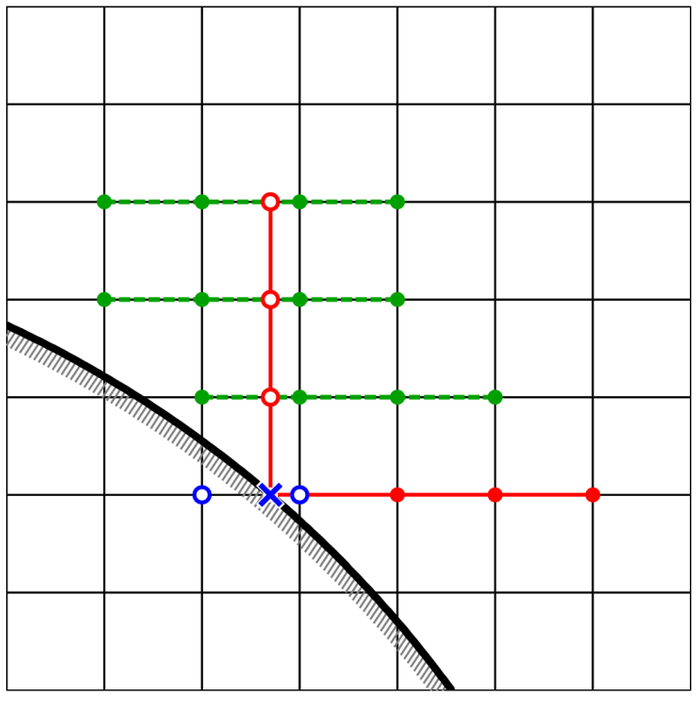
Full II treatment: more consequences

- Parabolic solver: diffusion

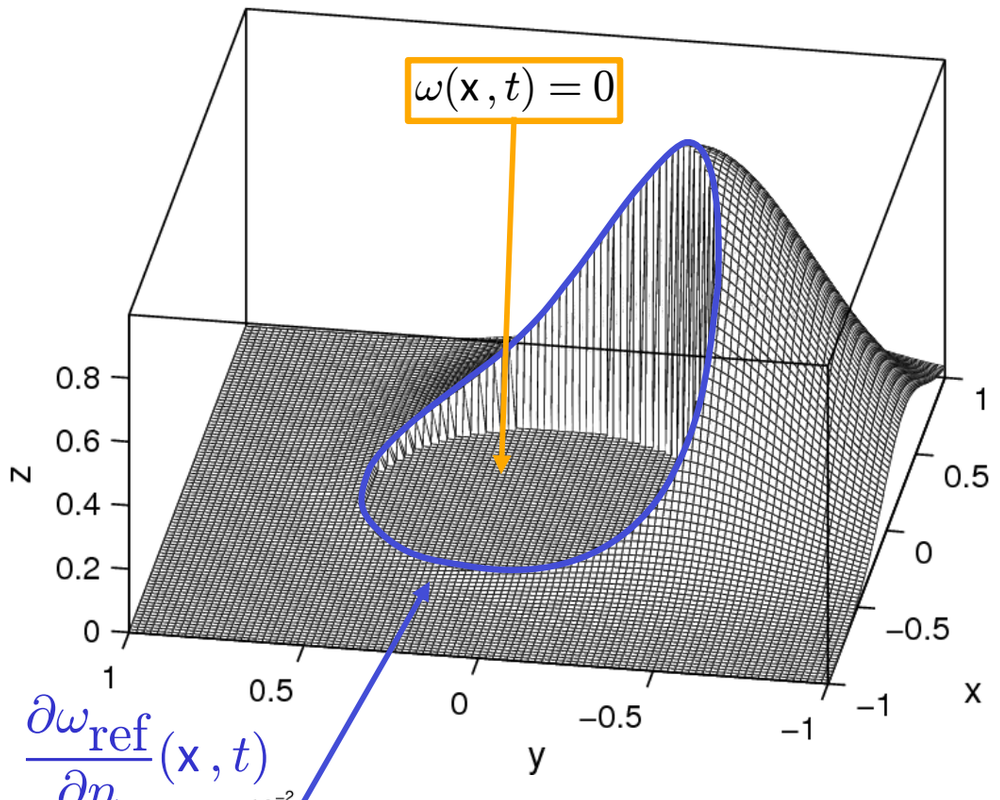
$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega$$

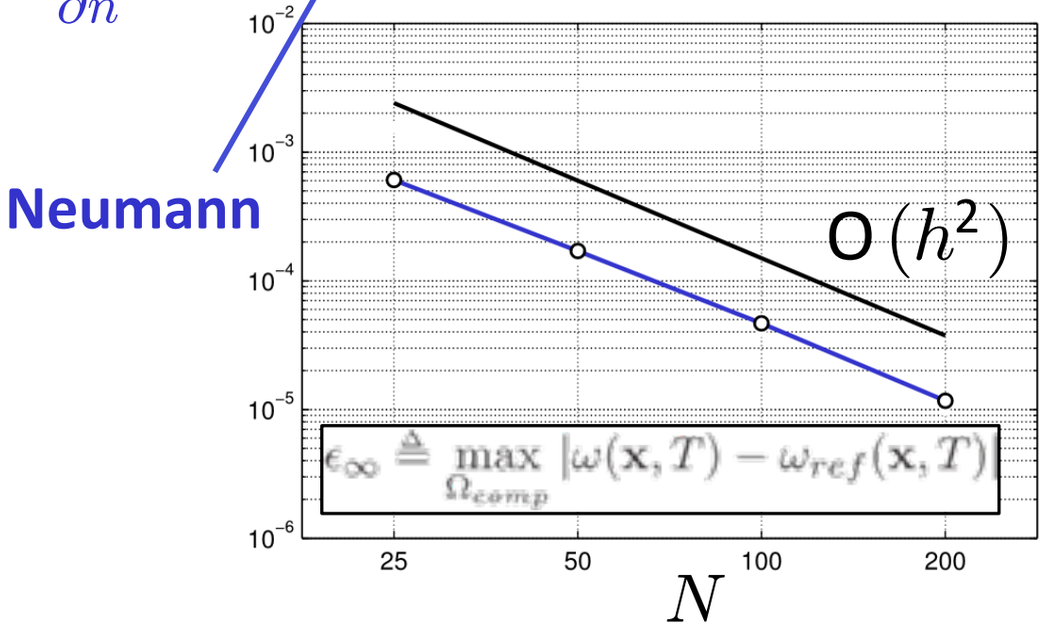
Compatible extrapolation :

enforce a Neumann BC.



Stability constraint nearly unchanged wrt case without interface.

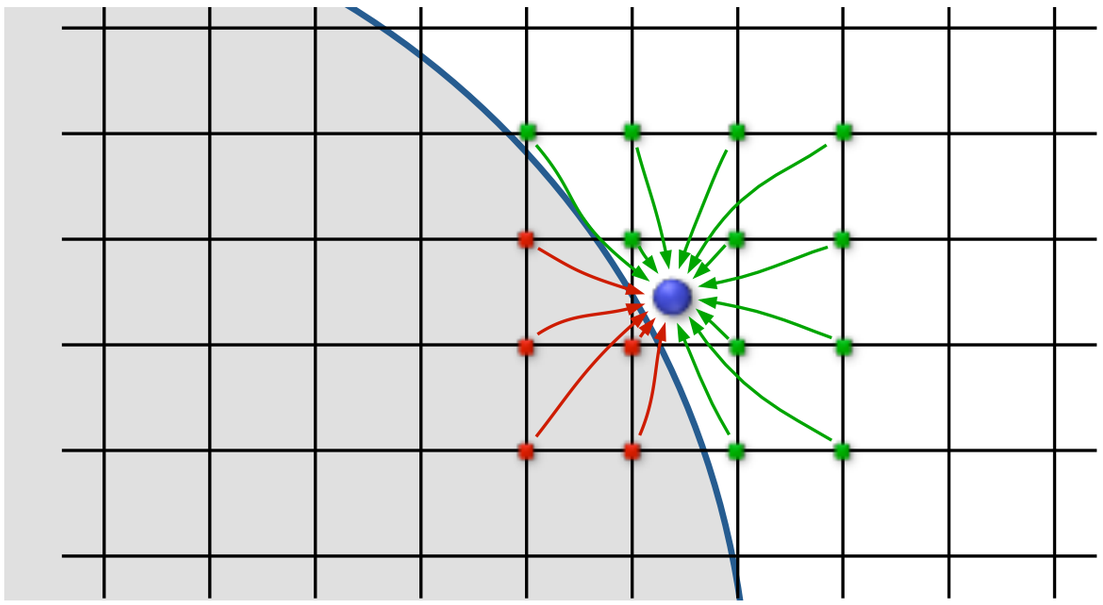


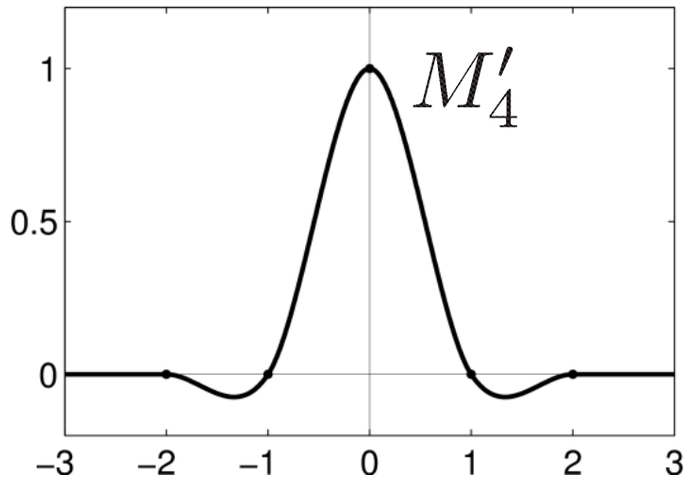


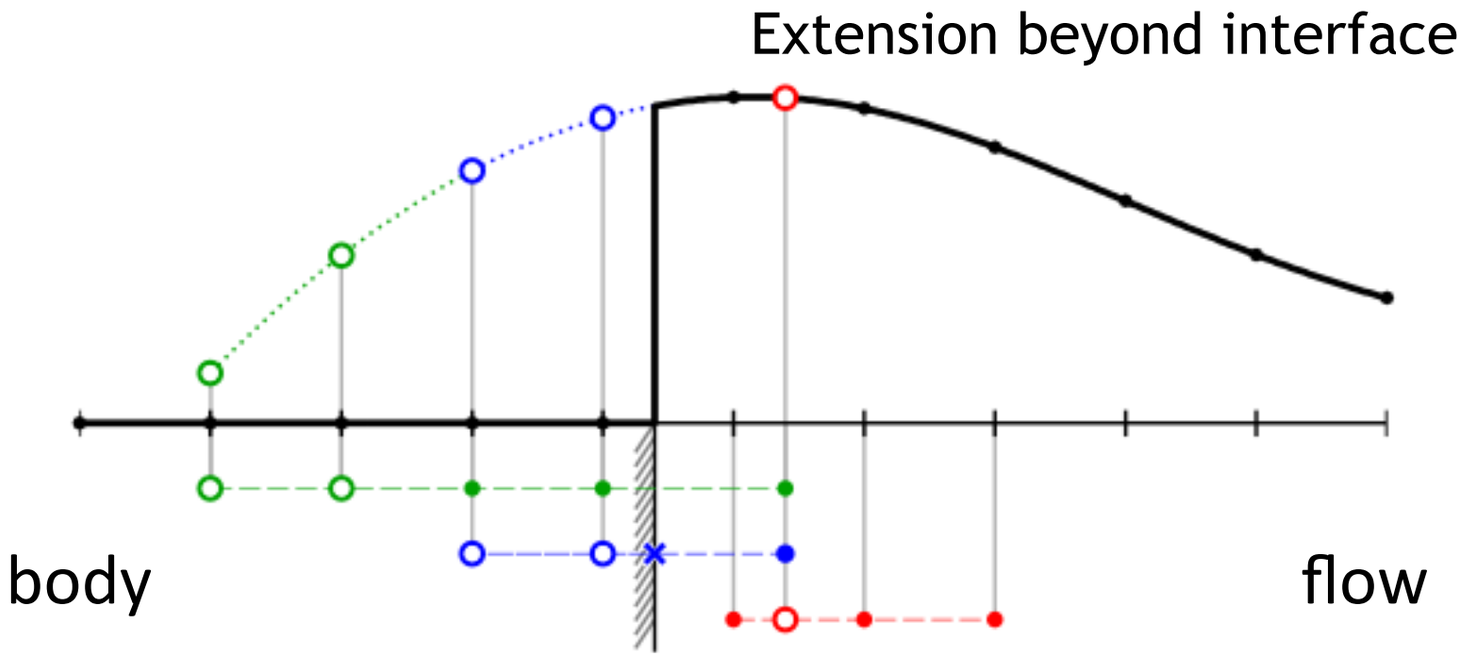
Full II treatment: more consequences

- Jump-corrected Mesh-Particle interpolation

Grid ghosts → Standard interpolation







Based on point in flow and wall information

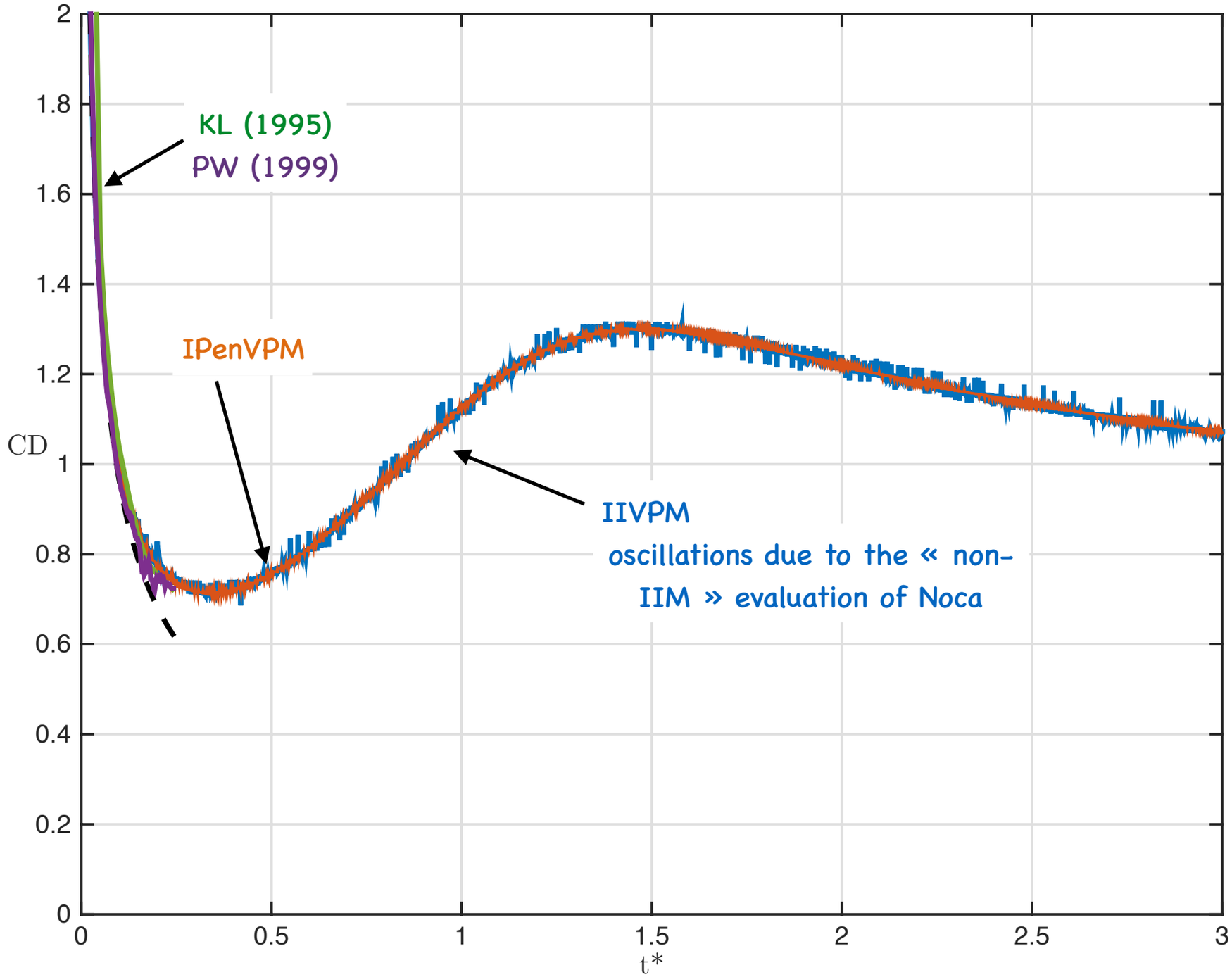
1D stencils

Blending

Results: Cylinder at $Re_D = 550$

Results: Cylinder at $Re_D = 550$

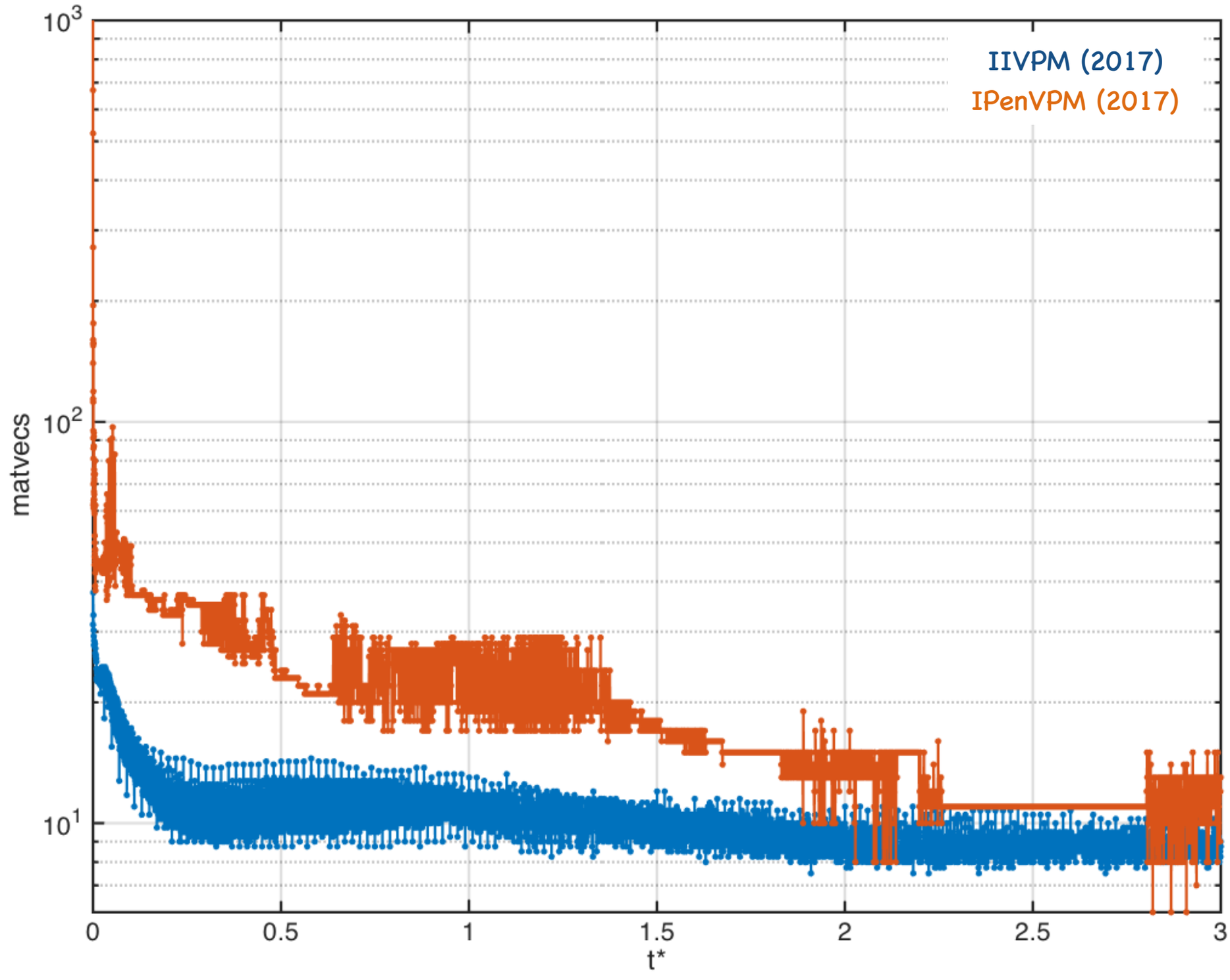
- Drag coefficient C_D



R/h = 204

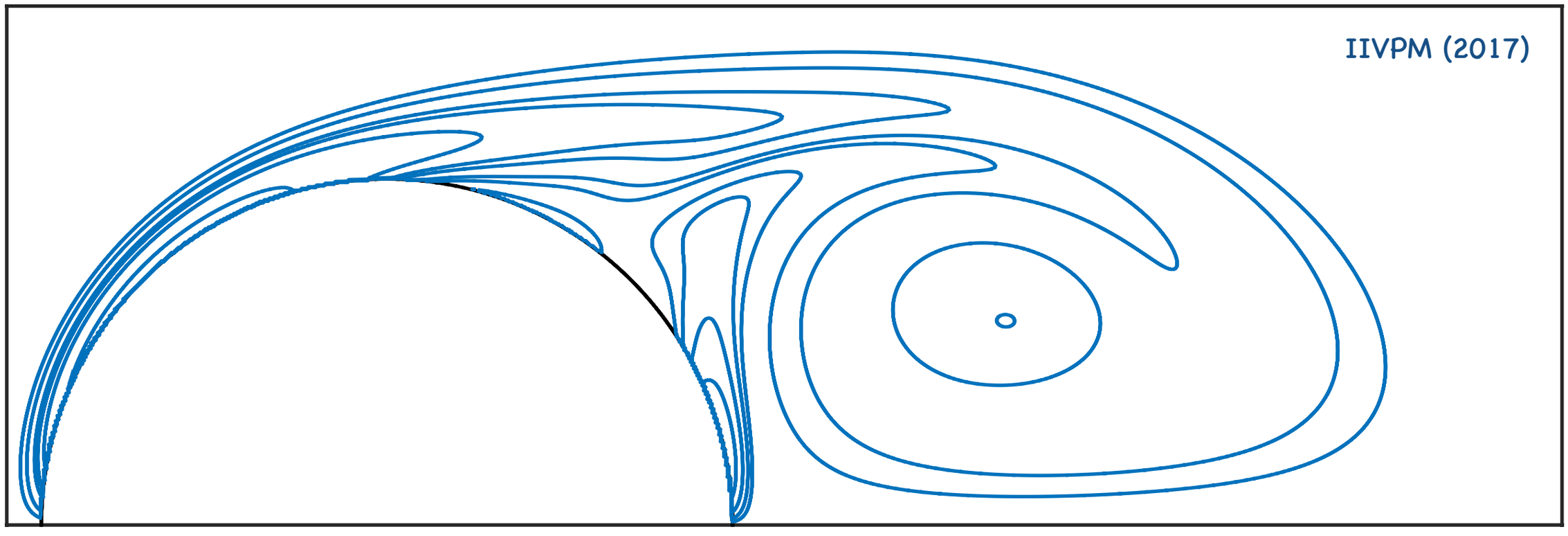
Results: Cylinder at $Re_D = 550$

- Cost: number of Poisson solver calls



Results: Cylinder at $Re_D = 550$

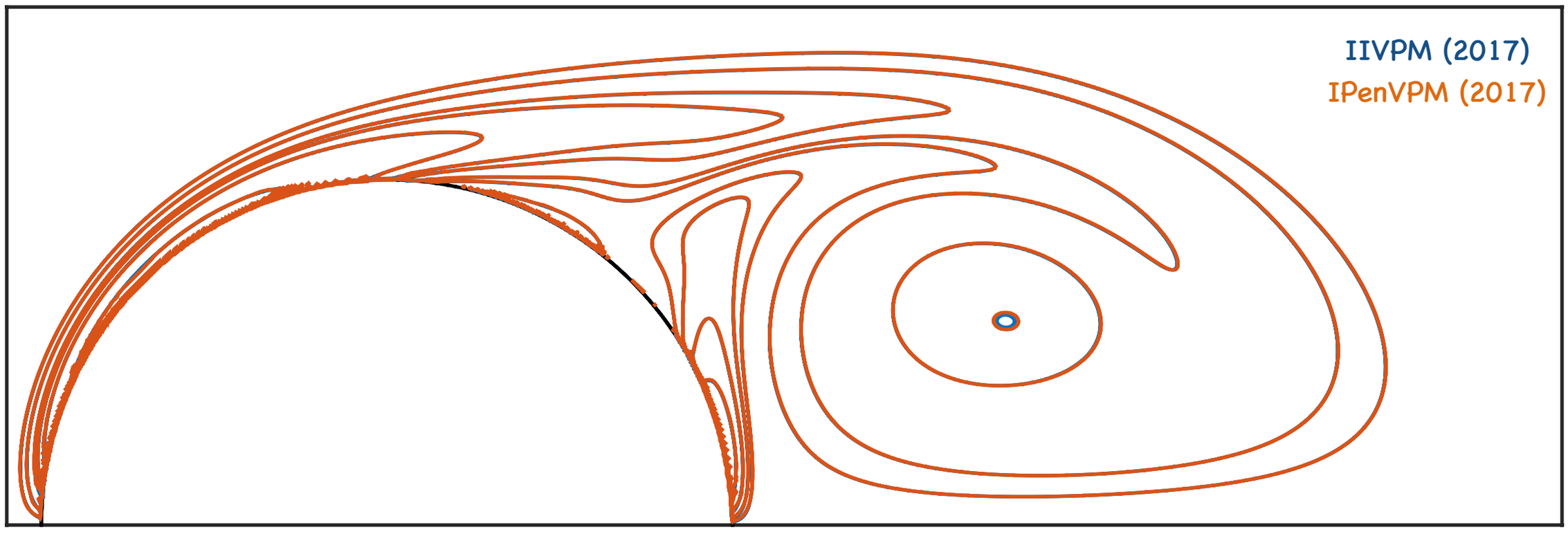
- Flow comparison: iso-vorticity



R/h = 204, t* = 3.0

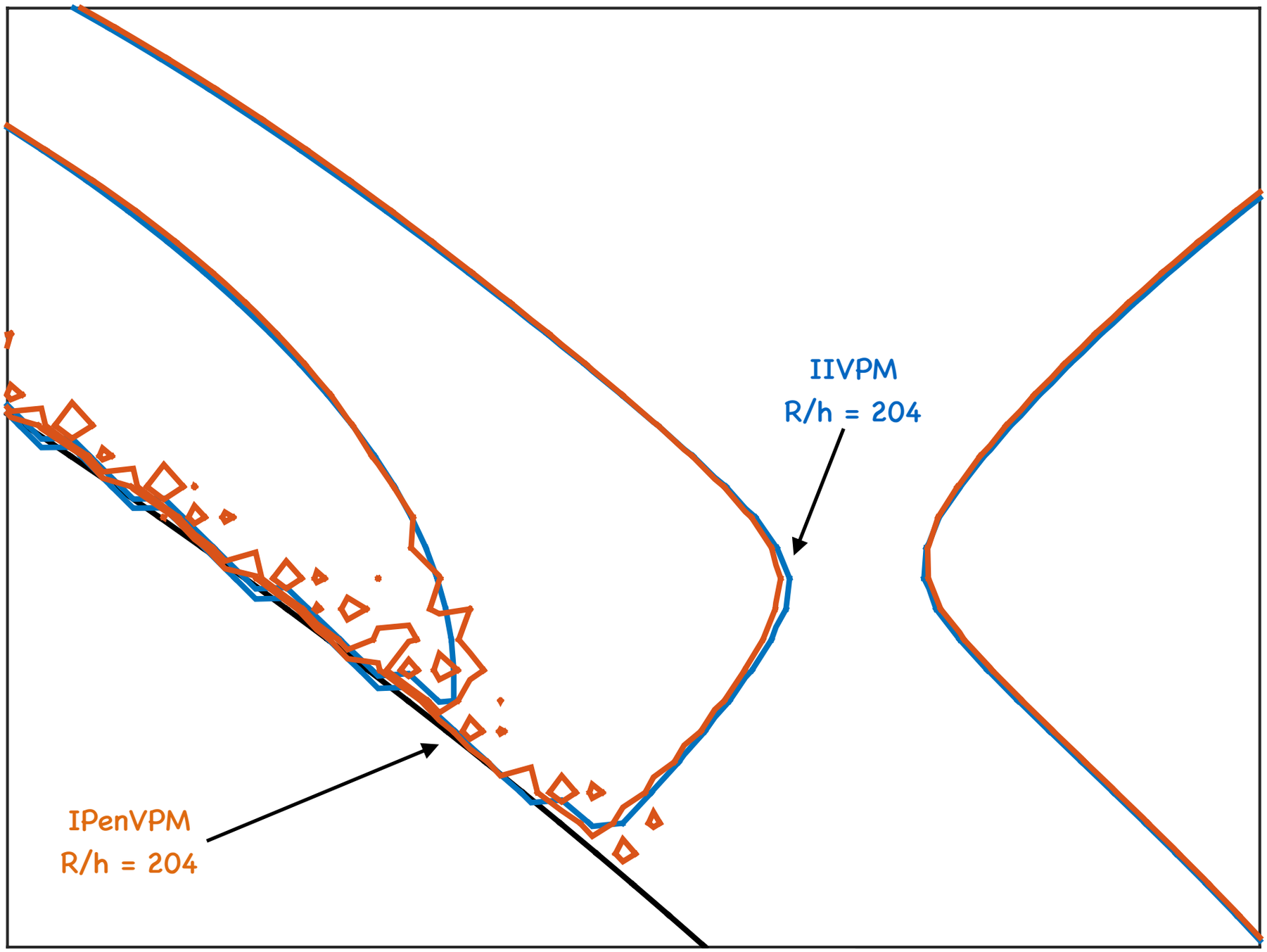
Results: Cylinder at $Re_D = 550$

- Flow comparison: iso-vorticity



R/h = 204, t* = 3.0

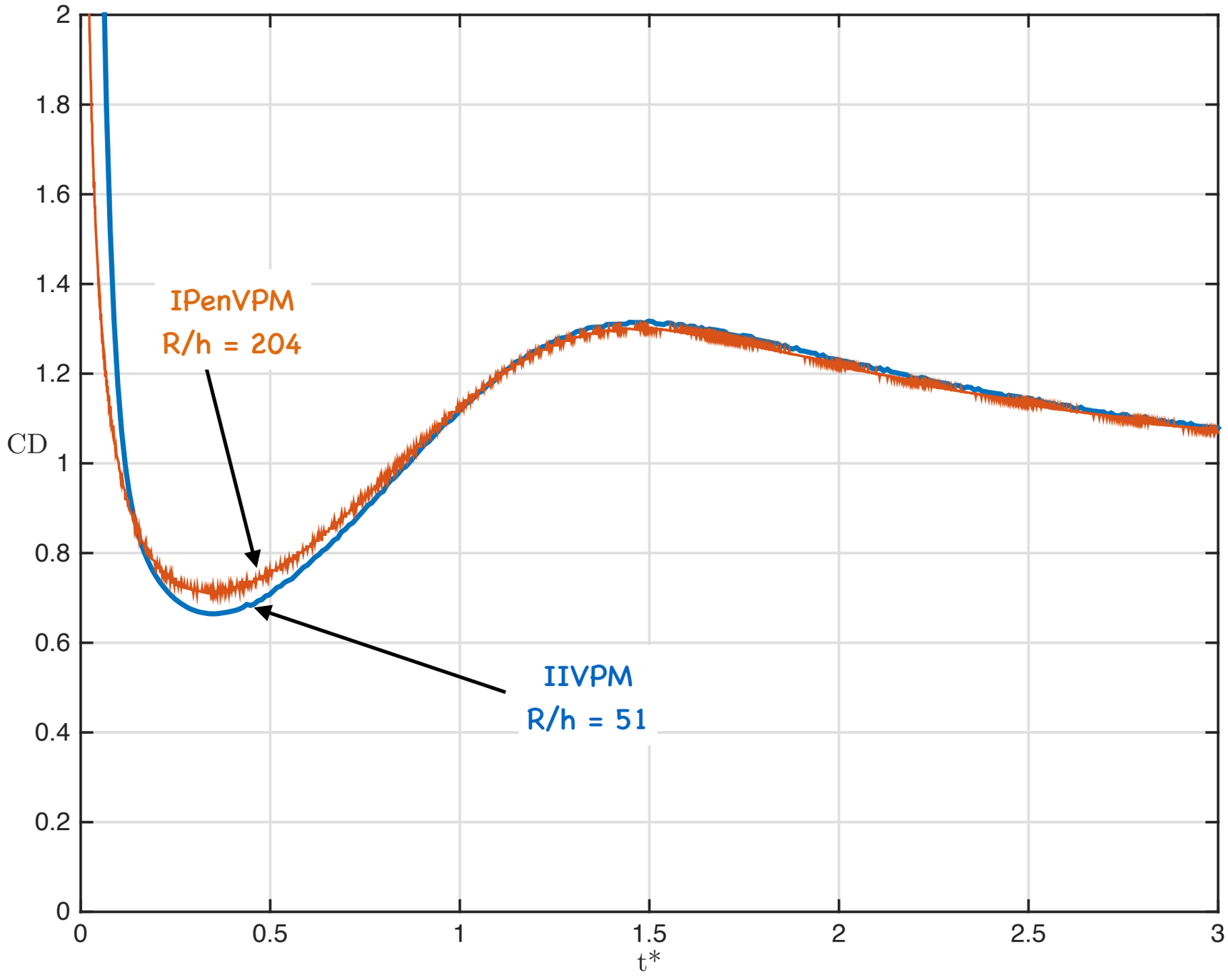
Results: Cylinder at $Re_D = 550$



R/h = 204, t* = 3.0

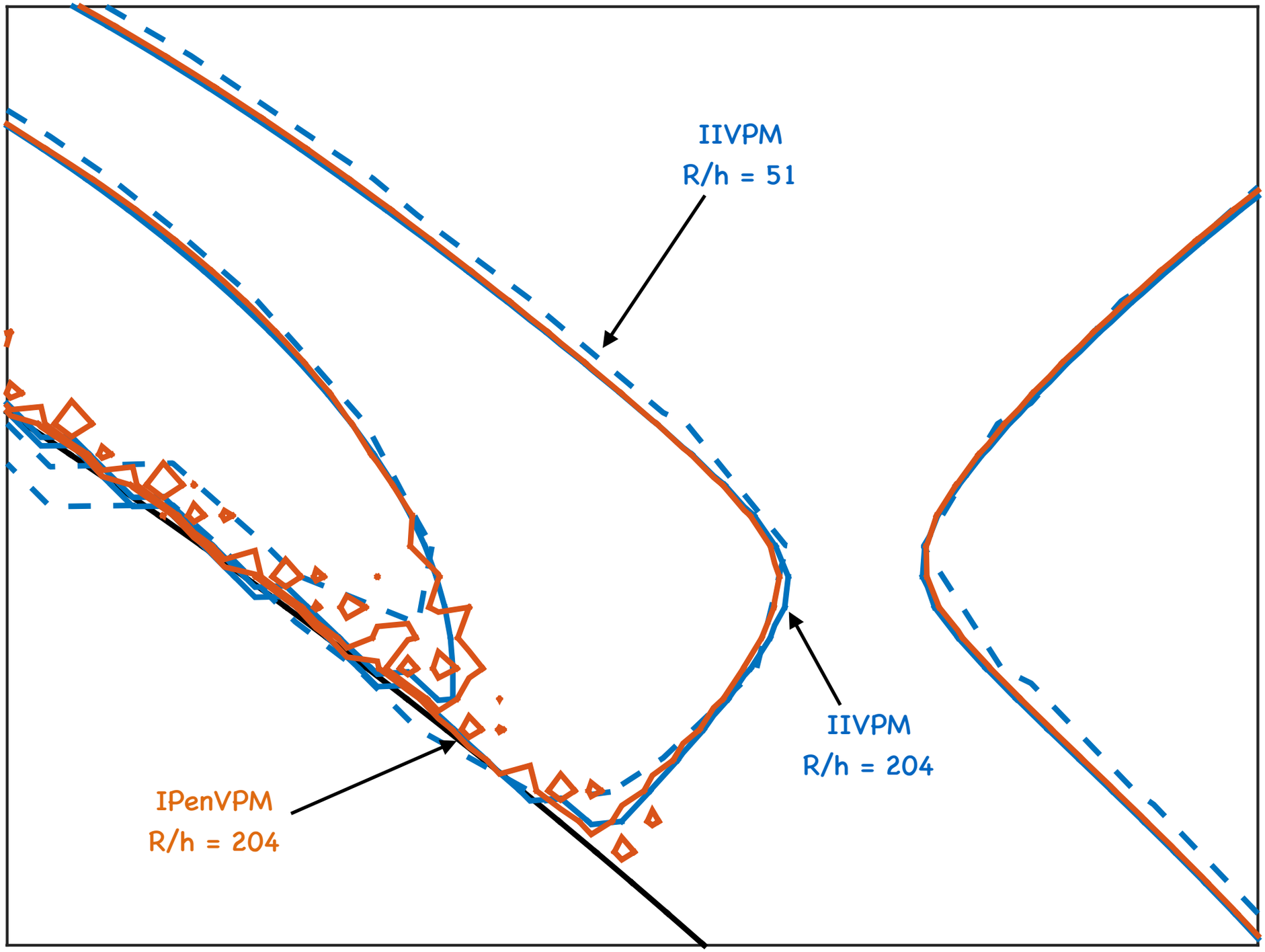
Results: Cylinder at $Re_D = 550$

- Accurate interfaces enables (much) lower resolution



Results: Cylinder at $Re_D = 550$

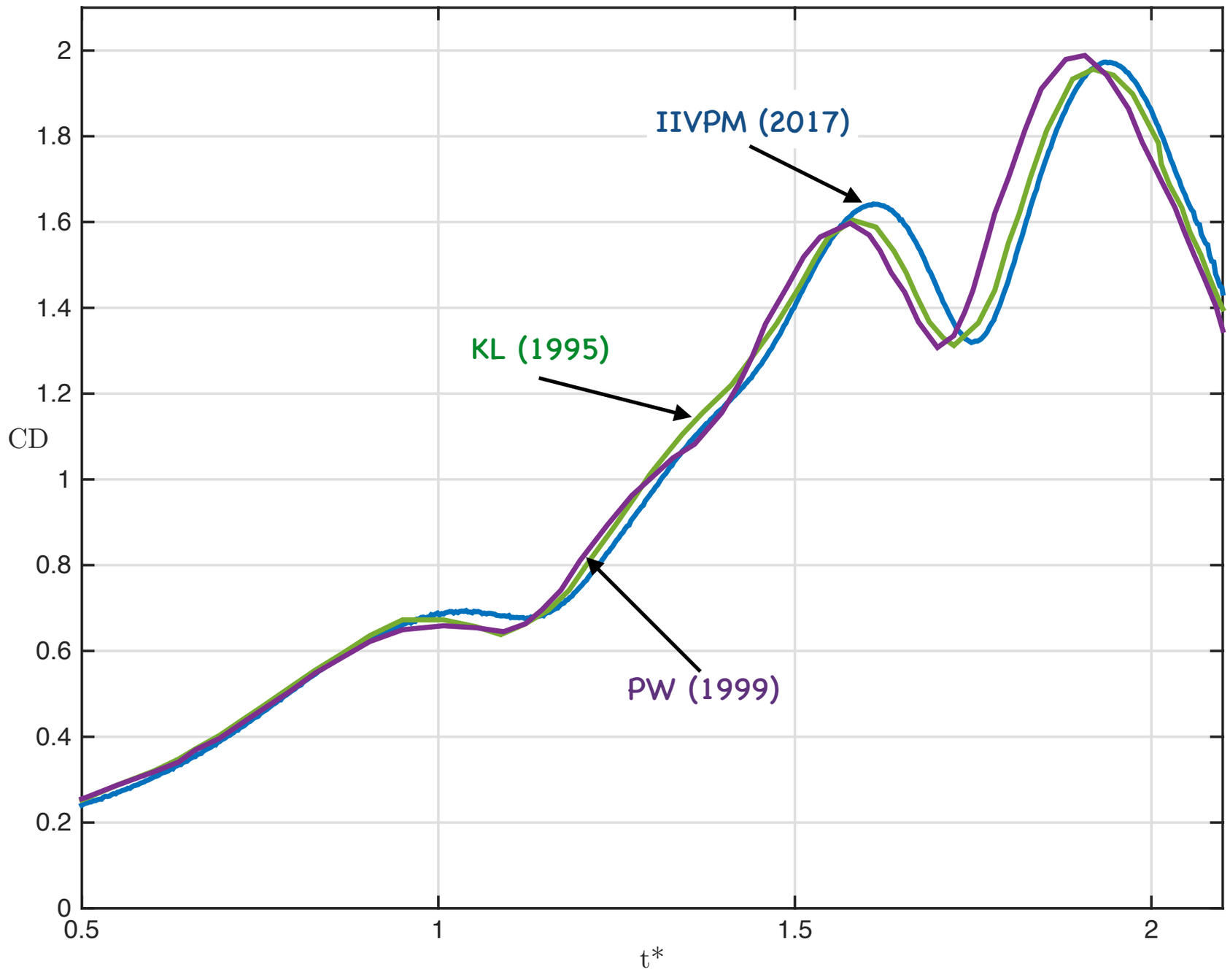
- Flow comparison iso-vorticity



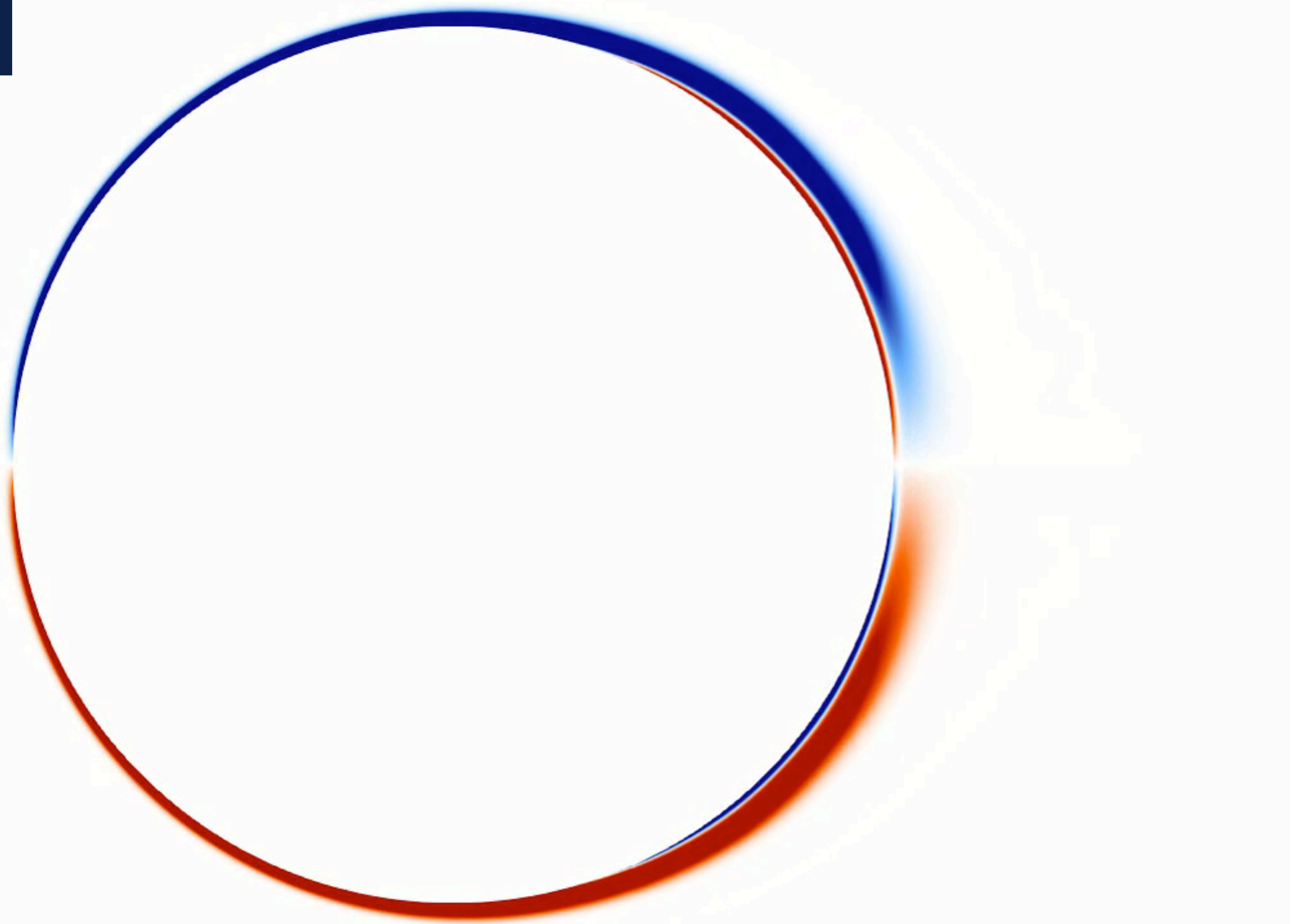
Results: Cylinder at $Re_D = 9500$

Results: Cylinder at $Re_D = 9500$

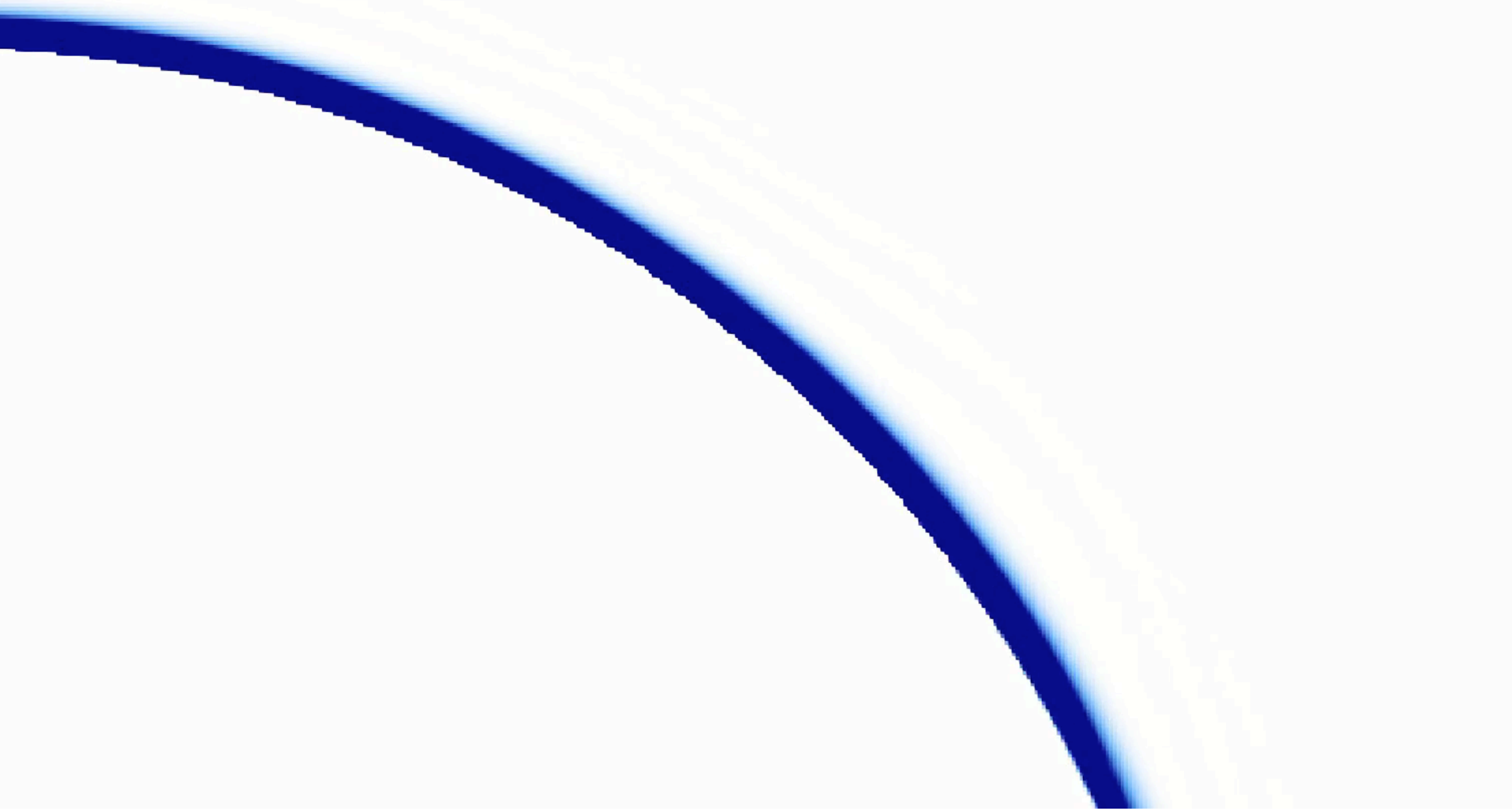
- Drag coefficient C_D



R/h = 341



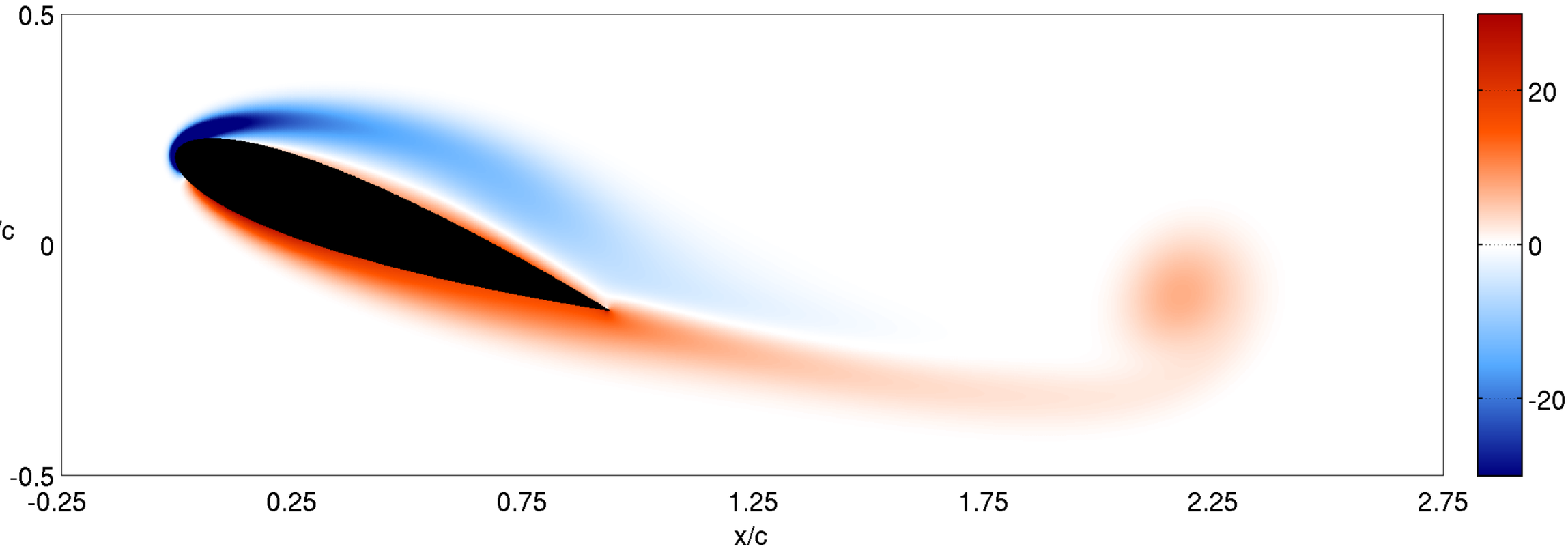
R/h = 341, Re_D = 9500



R/h = 341, Re_D = 9500

Results: Airfoil at Re = 500

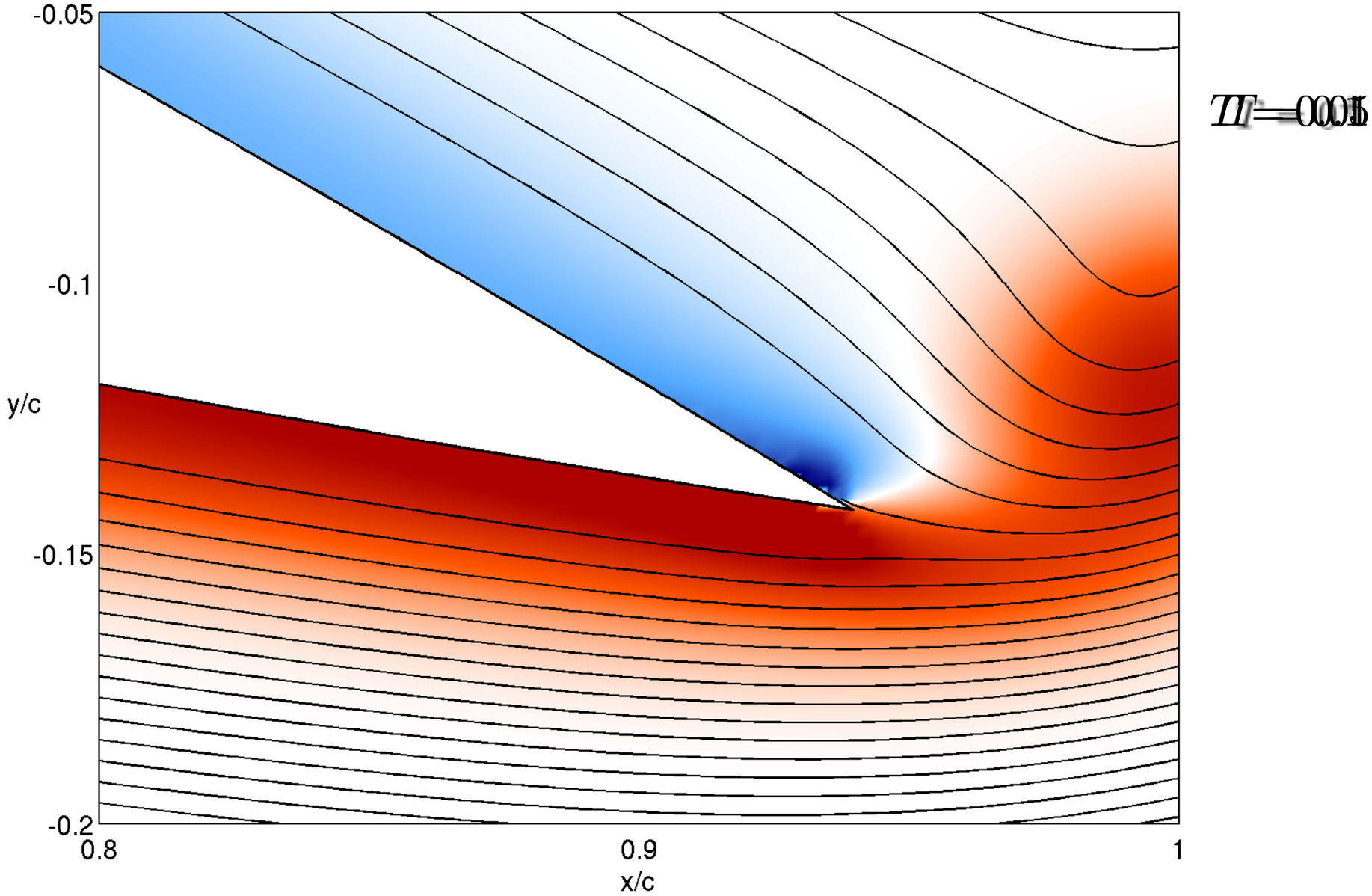
$$Re = \frac{U_\infty c}{\nu} = 500$$



Results: Airfoil at Re = 500

- Starting vortex

$$Re = \frac{U_\infty c}{\nu} = 500$$



Take-home messages (2)

- IIVPM method with interesting features
  - ⋆ Poisson solver
    - ✳ Couples IIM with efficient Poisson solvers (FFT, FMM, GMG,...)
    - ✳ iterates on the boundary instead of the volume (as in IPenVPM)
    - ✳ Reduces the cost using recycling in the iterative solver
  - ⋆ Provides sharp (2nd order) boundary treatment
- At a cost
  - ⋆ pervasive methodological modifications
  - ⋆ implementation complexity
- Next steps
  - ⋆ Extract wall quantities
  - ⋆ Preconditioning of the iterative solver

- but ...

Fluid-structure interaction

- What if $\mathbf{u}_s$ is part of the problem?[1,2,3]

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

- FSI through combination of

  - ⋆ Brinkman penalization :

    - ∗ $\lambda \chi_s (\mathbf{u}_s - \mathbf{u})$ with $\mathbf{u}_s = \overbrace{\mathbf{u}_T}^{\text{Translation}} + \overbrace{\mathbf{u}_R}^{\text{Rotation}} + \overbrace{\mathbf{u}_{def}}^{\text{Deformation}}$

    - ∗ $\nabla \cdot \mathbf{u}_{def} \neq 0$, deformation not necessarily solenoidal

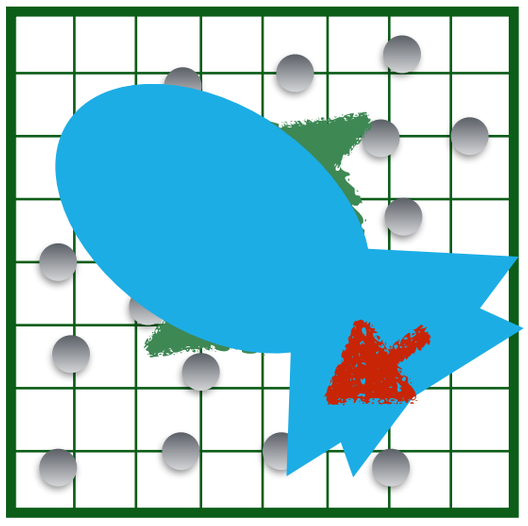
  - ⋆ Projection method

    - ∗ Evaluation of the flux of linear and angular momentum over one time step

    - ∗ Force-free approach

    - ∗ Find $\mathbf{u}_T$ and $\mathbf{u}_R$

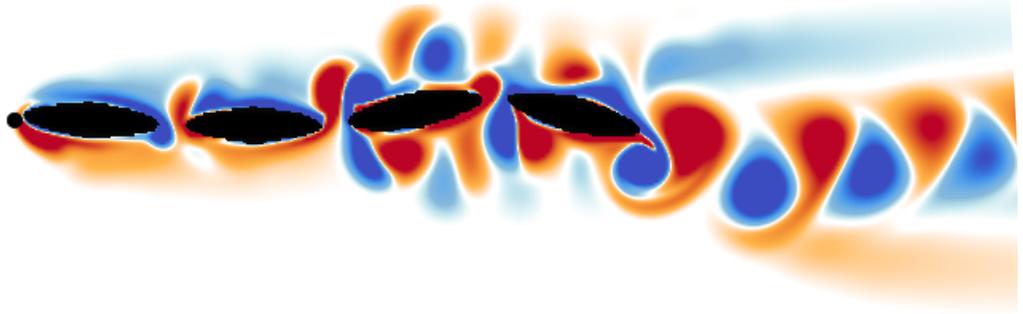
    - ∗ For **given** deformation velocities $\mathbf{u}_{def}$



1. *Patankar et al., CNME, 2005*
2. *Coquerelle et al., 2008*
3. *Gazzola et al., JCP, 2011*

Fluid-structure-actuation interaction

- What if u_{def} is part of the problem?[1]
- i.e. solve for dynamics of structure under actuation and hydrodynamic forces



  - ⋆ Robotic/articulated swimmers
  - ⋆ Simulate gait control and generation … all the way to flow
- Get the hydrodynamic stresses back!

$$\mathbf{F}_{hyd} = \int_{\delta\Omega} \sigma \cdot \mathbf{n}\, dS = \int_{\Omega} \nabla \cdot \sigma\, dV$$

$$\rho_f \frac{D\mathbf{u}}{Dt} = \nabla \cdot \sigma + \rho_f \lambda(\mathbf{u}_s - \mathbf{u})$$

Projection **Penalization**

- = extra Post-processing step to obtain forces and torques on structure joints

1. *Bernier et al, arXiv + JCP (submitted)*

Fluid-structure-actuation interaction

- Hydrodynamic forces and moments

Projection

$$\mathbf{F}_{proj} = \frac{D}{Dt}\int_{\Omega} \rho_f \chi_s \mathbf{u}\, dV$$

$$M_{proj} = \frac{D}{Dt}\int_{\Omega} \rho_f \chi_s (\mathbf{x} \times \mathbf{u})\, dV - \mathbf{x}_{cm} \times \mathbf{F}_{proj}$$

Penalization

$$\mathbf{F}^n_{pen} = \int_{\Omega} \rho_f \lambda \chi_s^{n+1} (\mathbf{u}^n_\lambda - \mathbf{u}^n_s) d\mathbf{x}$$

$$M^n_{pen} = \int_{\Omega} \rho_f \lambda \chi_s^{n+1} (\mathbf{x}_{cm} \times (\mathbf{u}^n_\lambda - \mathbf{u}^n_s))\, d\mathbf{x}$$

- passed to a Multi-Body System solver

  to find new velocities and positions of deforming structure

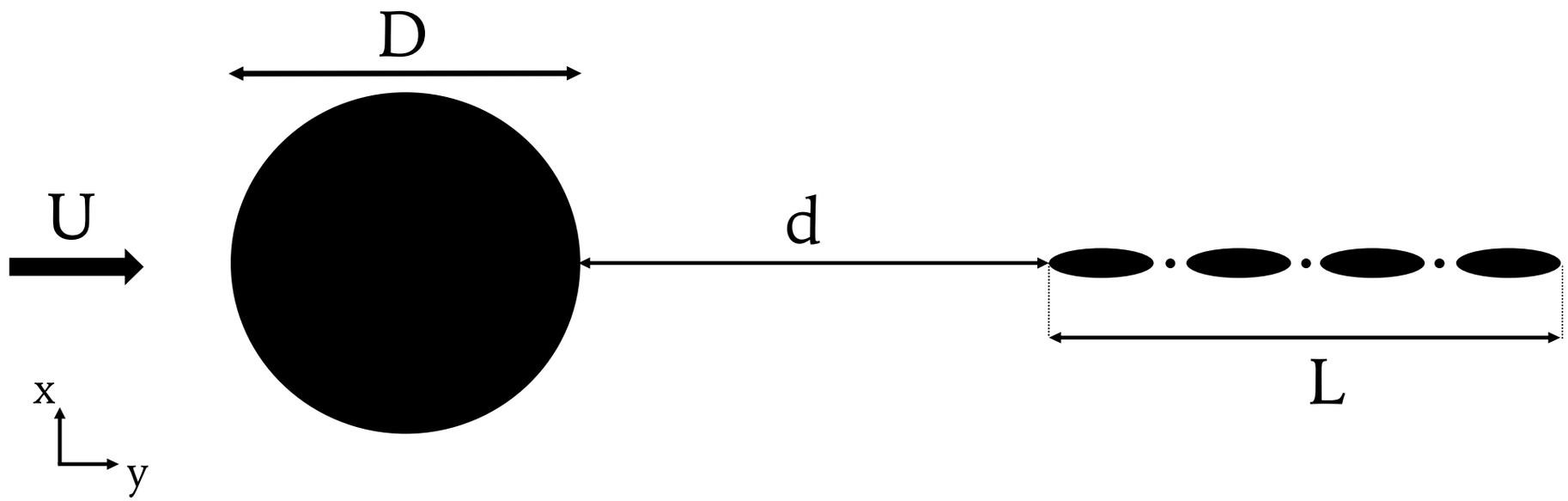
- Not force-free anymore!

  ⋆ Forces = time-averaged quantities, volume integrals computed from pseudo-time step

  ⋆ ... rather than surface integrals of challenging quantities, e.g. pressure

1. *Bernier et al, arXiv + JCP (submitted)*

Drafting strategies for swimmers[1]

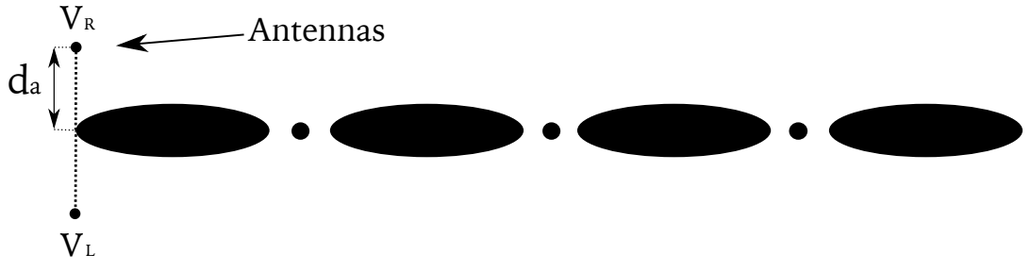


- Input data :
  - ⋆ Upstream velocity : U
  - ⋆ Reynolds number : $Re = \frac{UD}{\nu} = 100$
- 4 segments swimmer evolving in the wake of a cylinder
- Several situations tested: position hold, wake tracking…
- Motion generated by PI controller on joint torques
  - ⋆ speed control through gait frequency
  - ⋆ steering by addition of mean curvature

1. *Bernier et al, Biorob 2018*

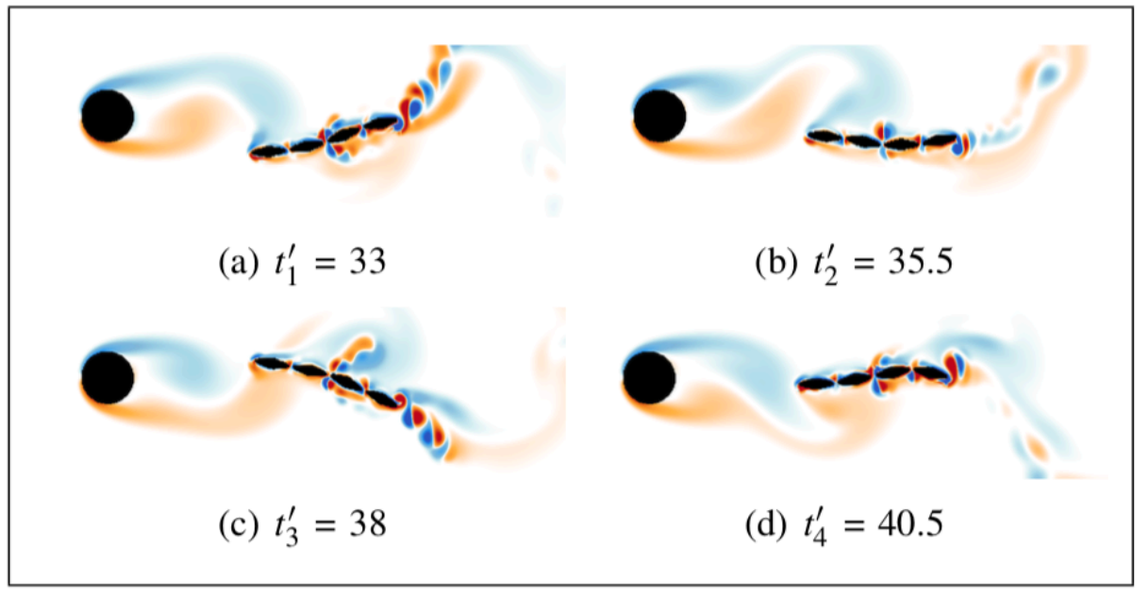
Drafting strategies for swimmers[1]

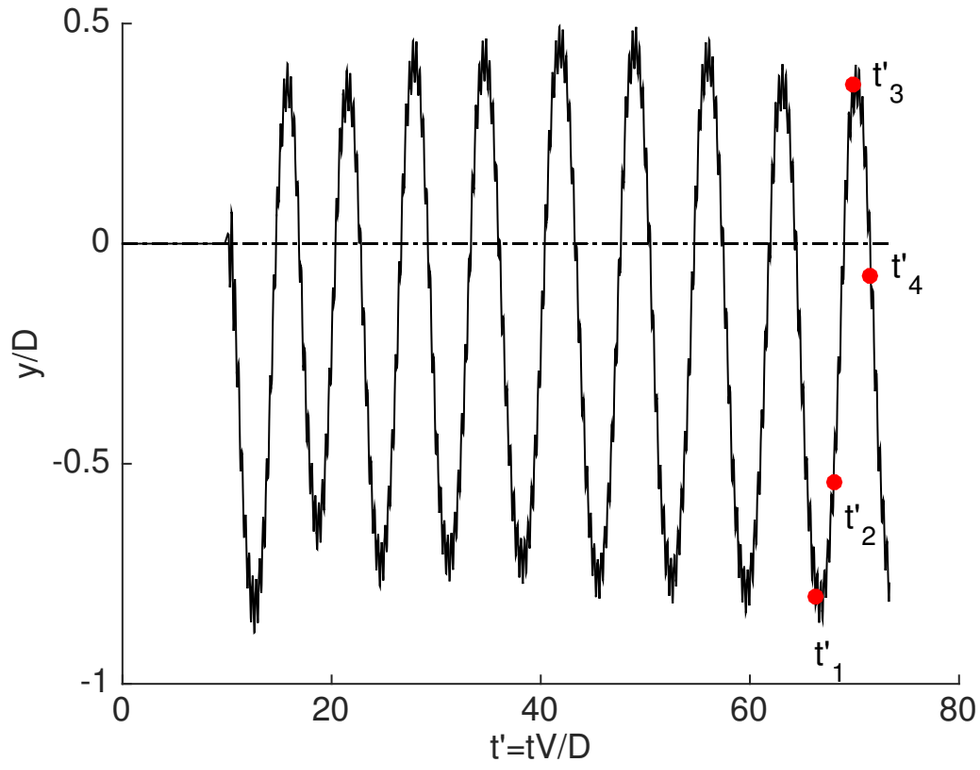
- Stationary cylinder wake tracking
  - ⋆ Streamwise position controller
  - ⋆ Steering controller
    - ✳ Swimmer senses the vorticity via 2 virtual antennas



    - ✳ Steering curvature proportional to $(V_R - V_L)$

  - ⋆ Mean propulsion power $\overline{P}^* = 12.94$

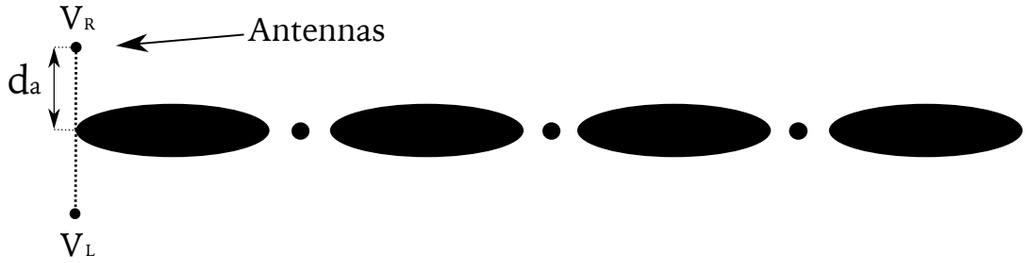




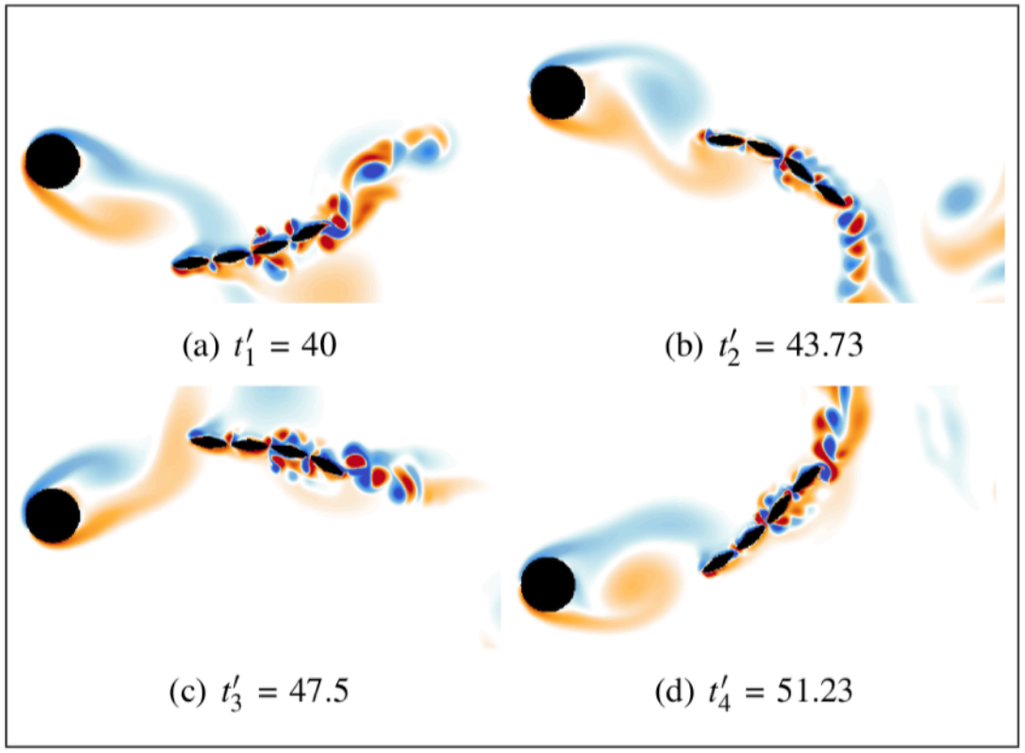
1. *Bernier et al, Biorob 2018*

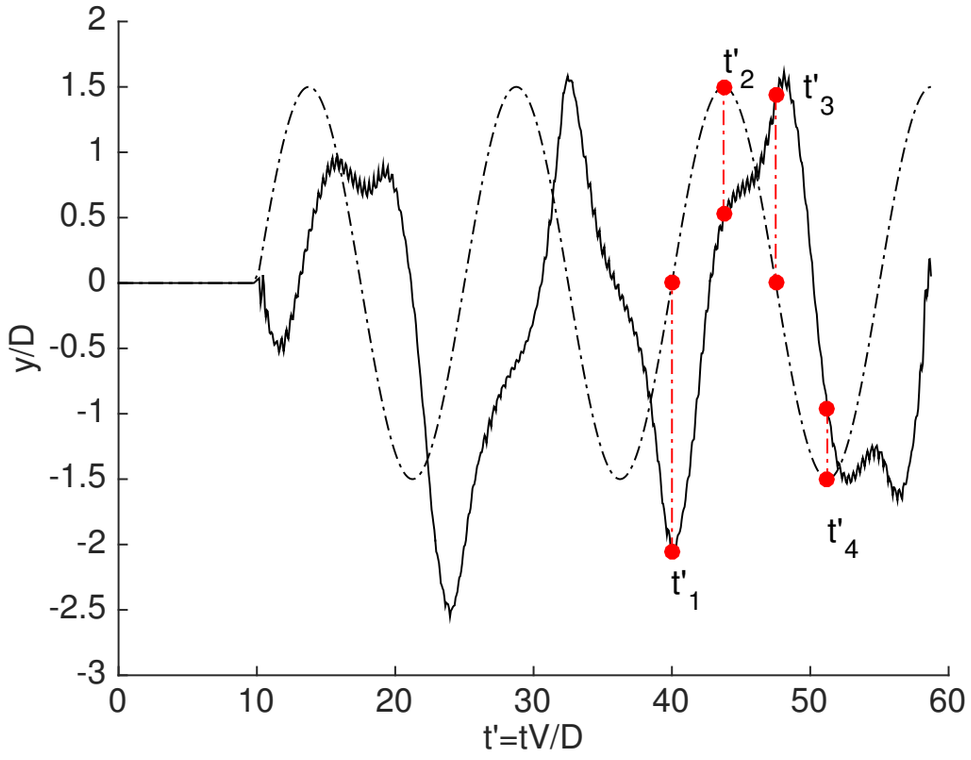
Drafting strategies for swimmers[1]

- Moving cylinder wake tracking
  - ⋆ Streamwise position controller
  - ⋆ Steering controller
    - ✳ Swimmer senses the vorticity via 2 virtual antennas



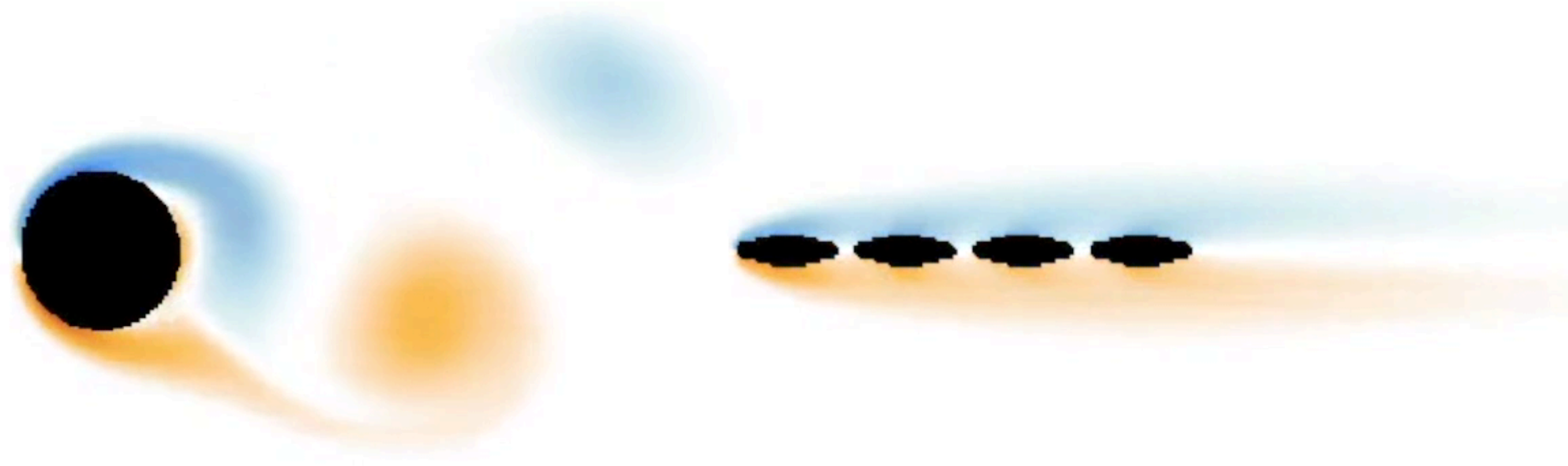
    - ✳ Steering curvature proportional to $(V_R - V_L)$
  - ⋆ Mean propulsion power $\overline{P}^* = 14.01$





1. *Bernier et al, Biorob 2018*

Drafting strategies for swimmers[1]



Stationary cylinder wake tracking - Section IV.B

1. *Bernier et al, Biorob 2018*

Take-home messages (3)

- Iterated Penalization and Immersed Interface schemes for Vortex Particle-Mesh methods (aka VIC,...)
  - ⋆ Penalization = Simplicity < > near-wall accuracy
  - ⋆ Immersed Interface = sharper, more affordable than iterated penalization (if efficient tools are used, e.g. FMM)
- but... ease in handling of deforming geometries
  - ⋆ Penalization > Immersed Interface
- Methodology for Fluid-Structure-Actuation Interaction
  - ⋆ penalization-based, surface quantities-free
  - ⋆ leveraged in bioinspired locomotion and energy harvesting



Acknowledgements: FRIA grant from the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) and Tier-1 supercomputer of Federation Wallonie-Bruxelles

Y. Marichal, Ph. Chatelain, and G. Winckelmans, *An immersed interface solver for the 2-D unbounded Poisson equation and its application to potential flow*, C&F (96), 2014	Y. Marichal, Ph. Chatelain, and G. Winckelmans, *Immersed interface interpolation schemes for particle-mesh methods*, JCP (326), 2016
T. Gillis, G. Winckelmans and Ph. Chatelain, *An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows*, JCP (347), 2017	T. Gillis, G. Winckelmans and Ph. Chatelain, *Fast immersed interface Poisson solver for 3D unbounded problems around arbitrary geometries*, JCP (354), 2018
C. Bernier, M. Gazzola, R. Ronsse and Ph. Chatelain, *Simulations of propelling and energy harvesting articulated bodies via vortex particle-mesh methods*, ArXiv + JCP (submitted)	