A hierarchical iterative solver for the Navier-Stokes equations

Driss Yakoubi

Joint works with:

E. Chamberland, J. Deteix, A. Fortin, A. Jendoubi and J. Urquiza

2012 CAIMS *Annual Meeting*

Toronto June 25

Outline

- Discretization of Navier-Stokes Equations
- Iterative Method
  - Preconditioning of the matrix associated with the velocity part
  - Schur complement approximation
- Numerical Simulations
  - Fluid flow around rigid objects
  - Symmetry breaking in a sudden axisymmetric constriction
  - High Performance Computing (HPC)
- Conclusion and Ongoing Works

Discretization of NSE

Navier Stokes Equations

$$\left\{ \begin{array}{r} \rho \dfrac{\partial \mathbf{u}}{\partial t} - \operatorname{div}(2\eta\varepsilon(\mathbf{u})) + \rho(\mathbf{u}\cdot\nabla\mathbf{u}) + \nabla p = \mathbf{f} \\ \operatorname{div}(\mathbf{u}) = 0 \\ \mathbf{u}(x,0) = \mathbf{u}_0(x) \quad \text{such that} \quad \operatorname{div}(\mathbf{u}_0) = 0 \end{array} \right.$$

with boundary conditions

$\varepsilon(\mathbf{u}) = (\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{t}})/2$ is the rate of strain tensor.

Discretization of NSE

Time Discretization

$$t_n = t_0 + n\Delta t$$

We use the backward second-order accurate implicit scheme :

$$\frac{\partial \mathbf{u}}{\partial t} \approx \frac{3\mathbf{u}^n - 4\mathbf{u}^{n-1} + \mathbf{u}^{n-2}}{2\Delta t}$$

Thus, at each time step, we obtain the non-linear equation:

$$\alpha \mathbf{u}^n - \mathrm{div}(2\eta\varepsilon(\mathbf{u}^n)) + \rho(\mathbf{u}^n \cdot \nabla \mathbf{u}^n) + \nabla p^n = \mathbf{f} + \alpha \mathbf{u}^*$$

$$\mathrm{div}(\mathbf{u}^n) = 0$$

where the right hand side depends only on $\mathbf{u}^{n-1}$ and $\mathbf{u}^{n-2}$

Discretization of NSE

Treatment of the non-linear term

At time step,

$$\alpha\mathbf{u}^n - \mathrm{div}(2\eta\varepsilon(\mathbf{u}^n)) + \rho(\mathbf{u}^n\cdot\nabla\mathbf{u}^n) + \nabla p^n = \mathbf{f} + \alpha\mathbf{u}^*$$

This non-linearity can be treated by several approach :

- Linearized method $\mathbf{u}^n\cdot\nabla\mathbf{u}^n \approx \mathbf{u}_e\cdot\nabla\mathbf{u}^n$ (for instance: $\mathbf{u}_e = 2\mathbf{u}^{n-1} - \mathbf{u}^{n-2}$)

- Fixed point method (Picard method):

  - Given $(\mathbf{u}_j^n, p_j^n)$ a solution at iteration j,
  - Obtain $(\mathbf{u}_{j+1}^n, p_{j+1}^n)$ a solution of

$$\alpha\mathbf{u}_{j+1}^n - \mathrm{div}(2\eta\varepsilon(\mathbf{u}_{j+1}^n)) + \rho(\mathbf{u}_j^n\cdot\nabla\mathbf{u}_{j+1}^n) + \nabla p_{j+1}^n = \mathbf{f} + \alpha\mathbf{u}^*$$

Discretization of NSE

Treatment of the non-linear term

- *Newton method:*

at "Newton" iteration j, the approximation of the convection term:

$$\mathbf{u}_{j+1}^n \cdot \nabla \mathbf{u}_{j+1}^n \approx \delta\mathbf{u} \cdot \nabla \mathbf{u}_j^n + \mathbf{u}_j^n . \nabla(\delta\mathbf{u}) + \mathbf{u}_j^n . \nabla \mathbf{u}_j^n$$

where $\mathbf{u}_{j+1}^n = \mathbf{u}_j^n + \delta\mathbf{u}.$

$$\Rightarrow \quad \alpha \mathbf{u}_{j+1}^n - div(2\eta\varepsilon(\mathbf{u}_{j+1}^n)) + \rho(\mathbf{u}_j^n \cdot \nabla \delta\mathbf{u} + \delta\mathbf{u} \cdot \nabla \mathbf{u}_j^n) + \nabla p_{j+1}^n = \mathbf{f} + \alpha \mathbf{u}^* - \rho(\mathbf{u}_j^n . \nabla)\mathbf{u}_j^n$$

Discretization of NSE

Space Discretization

The discrete subspaces $\mathbf{V}_h$ and Q_h are chosen as follows:

$$\mathbf{V}_h = \{\mathbf{v}_h \in C^0(\Omega)^d, \mathbf{v}_h|_T \in (P_2)^d \ \forall T \in \mathcal{T}_h\}$$

$$Q_h = \{q_h \in C^0(\Omega), q_h|_T \in (P_1) \ \forall T \in \mathcal{T}_h\}$$

Then, the velocity and pressure can be decomposed as follows:

$$\mathbf{u}_h = \sum_{i=1}^{n} u_i \boldsymbol{\phi}_i \quad \text{and} \quad p_h = \sum_{j=1}^{m} p_j \psi_j$$

Furthermore, the hierarchical basis allows the follows decompozition

$$V_h = V_l \oplus V_q \qquad (\ u = u_l + u_q\ )$$

Where: $\left\{ \begin{array}{ll} V_l & \text{is the subspace of continuous piecewise linear polynomials, and} \\ V_q & \text{is the complementary subspace of continuous piecewise quadratic} \end{array} \right.$

Discretization of NSE

Algebraic System

Finally, the discrete formulation can be rewritten as:

$$\mathscr{A}\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & B^{t} \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

Where $F = \alpha M + \eta D + \rho(C + N)$ such that

- *Mass matrix* $M_{ij} = \int_{\Omega} \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j \, \mathrm{d}x$
- *Diffusion matrix* $D_{ij} = \int_{\Omega} \varepsilon(\boldsymbol{\phi}_i) : \varepsilon(\boldsymbol{\phi}_j) \, \mathrm{d}x$
- *Convection matrix* $C_{ij} = \int_{\Omega} (\mathbf{w} \cdot \nabla \boldsymbol{\phi}_i) \cdot \boldsymbol{\phi}_j \, \mathrm{d}x$ $\qquad \underbrace{N_{ij} = \int_{\Omega} (\boldsymbol{\phi}_i \cdot \nabla \mathbf{w}) \cdot \boldsymbol{\phi}_j \, \mathrm{d}x}_{\text{From Newton Method}}$
- *Divergence matrix* $B_{ij} = -\int_{\Omega} \psi_j \operatorname{div}(\boldsymbol{\phi}_i) \, \mathrm{d}x$

Resolution Method

Direct Method

To solve the system: $$\begin{bmatrix} F & B^t \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

we can use the direct method

- The matrix is singular, then we introduce a penalization term

$$\begin{bmatrix} F & B^t \\ B & \varepsilon M \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

- But, ...

Resolution Method

Iterative Method

Our approach consists:

- Solve simultaneously the velocity and the pressure
- GCR or GMRES (Kryolv method)
- Preconditioner:

$$\begin{bmatrix} F & B^{t} \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ B F^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & -S \end{bmatrix} \begin{bmatrix} I & F^{-1} B^{t} \\ 0 & I \end{bmatrix}$$

where $S = B F^{-1} B^{t}$ stands for the Schur complement

This factorization cannot be used as preconditioner because
F is large-scale matrix and S is dense matrix.

Iterative Method

Iterative Method

- Then, we introduce an approximation: $\tilde{F}$ *and* $\tilde{S}$
- And, we choose the follow preconditioner:

$$\mathscr{P}_{\mathrm{R}} = \begin{bmatrix} \tilde{F} & B^{\mathrm{t}} \\ 0 & -\tilde{S} \end{bmatrix}$$

- The action of this preconditioner on a residual vector can be rewritten as:

$$\begin{cases} \delta p = -\tilde{S}^{-1} r_p \\ \delta u = \tilde{F}^{-1}(r_u - B^{\mathrm{t}} \delta p) \end{cases}$$

We have to solve 2 systems

Iterative Method

Preconditioning the matrix F

$$\delta u = \tilde{F}^{-1}(r_u - B^{\mathrm{t}}\delta p) \quad \Leftrightarrow \quad F\delta u = \tilde{r}_u$$

We use the **hierarchical** basis for the quadratic FE discretization

Then, we can decompose the velocity into the linear part and a quadratic correction:

$$u = u_{\mathrm{l}} + u_{\mathrm{q}}$$

Consequently, the matrix F can be rewritten as: $F = \begin{bmatrix} F_{\mathrm{ll}} & F_{\mathrm{lq}} \\ F_{\mathrm{ql}} & F_{\mathrm{qq}} \end{bmatrix}$

Finally, we use the following Algorithm proposed by El Maliki and Fortin

1. Solve by few iterations of SOR: $F\delta = r$ where $\delta = (\delta_{\mathrm{l}}, \delta_{\mathrm{q}})^{\mathrm{t}}$ and $r = (r_{\mathrm{l}}, r_{\mathrm{q}})^{\mathrm{t}}$.
2. Compute the residual: $d_{\mathrm{l}} = r_{\mathrm{l}} - F_{\mathrm{ll}}\delta_{\mathrm{l}} - F_{\mathrm{lq}}\delta_{\mathrm{q}}$.
3. Solve by a direct or few iterations of an iterative method: $F_{\mathrm{ll}}\delta_{\mathrm{l}}^{*} = d_{l}$.
4. Update the correction: $\delta = (\delta_{\mathrm{l}} + \delta_{\mathrm{l}}^{*}, \delta_{\mathrm{q}})^{\mathrm{t}}$.

Iterative Method

Schur complement approximation

Thanks to the discrete inf-sup condition proved by Brezzi-Fortin:

$$\xi^2 \leqslant \frac{p^{\mathrm{t}}(B D^{-1} B^{\mathrm{t}}) p}{p^{\mathrm{t}} M_p p} \leqslant \chi^2$$

we can remark that the matrix $BD^{-1}B^{\mathrm{t}}$ is spectrally equivalent to the mass matrix M_p

- *Stokes Case:* We can choose this approximation

$$S \simeq \tilde{S} = \frac{1}{\eta} M_p \simeq \frac{1}{\eta} \mathrm{diag}(M_p)$$

- *Navier Stokes Case:* Turek proposes the additive preconditioner:

$$S^{-1} \simeq M_p^{-1}(\alpha M_p + \eta D_p + \rho C_p) D_p^{-1}$$

Numerical Simulations with MEF++ Code

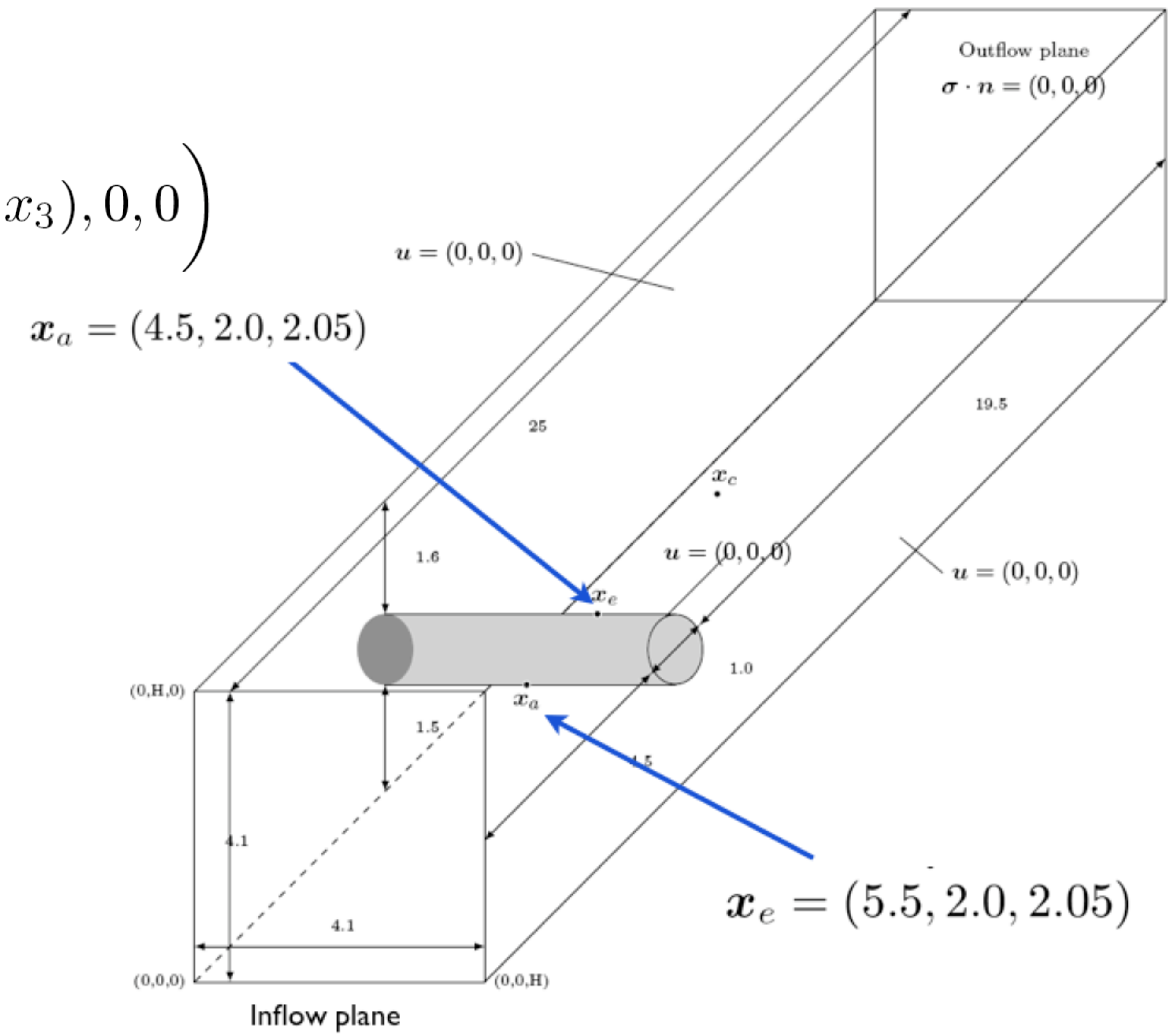
Fluid flow around rigid objects

- *Configuration and boundary conditions for flow around cylinder*

The inflow condition is:

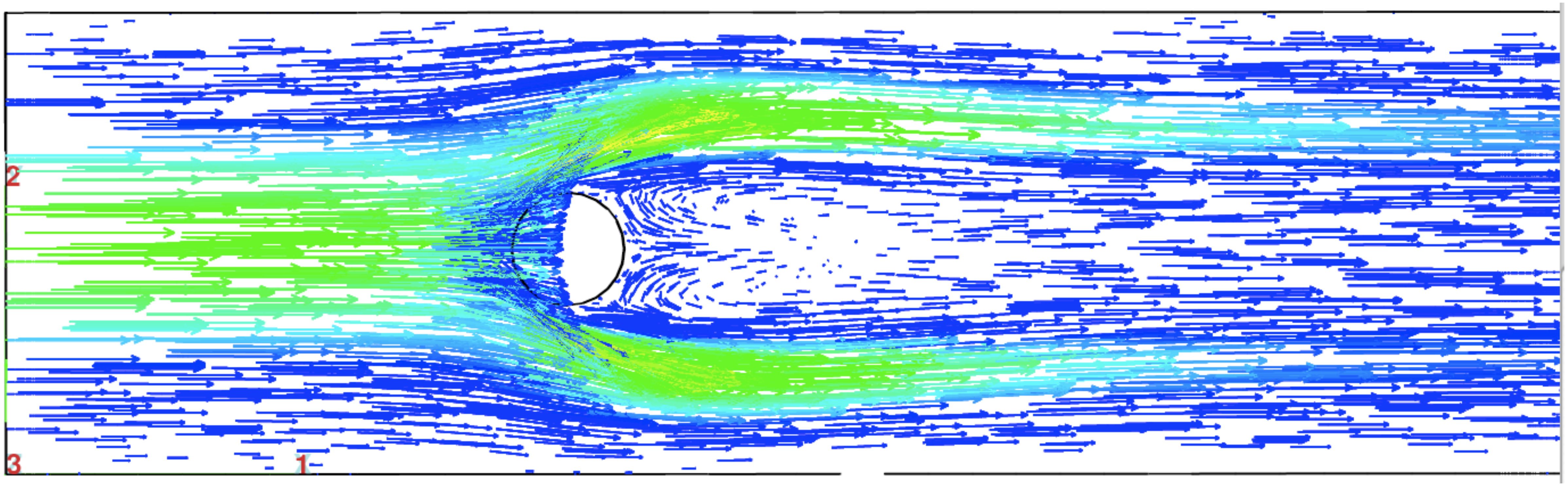
$$\boldsymbol{u}(0, x_2, x_3) = \left(\frac{72}{H^4} x_2 x_3 (H - x_2)(H - x_3), 0, 0\right)$$

$$Re = 20.$$



Numerical Simulations with MEF++ Code

Fluid flow around rigid objects



- *Comparison with Turek, Schafer,...*

	Our Results	*Turek Teams Results*
Δp	0.1694	**0.1693**
C_D	**6.1430**	**6.0928**

Numerical Simulations with MEF++ Code

Fluid flow in a sudden axisymmetric constriction

- *Geometry*

- *Physical experience*

  - **J. Vetel and A. Garon**

  Symmetry breaking

  at Reynolds Number = 255.

- *Stability Method*

  - **S. J. Sherwin et al**

  Symmetry breaking

  at Reynolds Number = 721.

Numerical Simulations with MEF++ Code

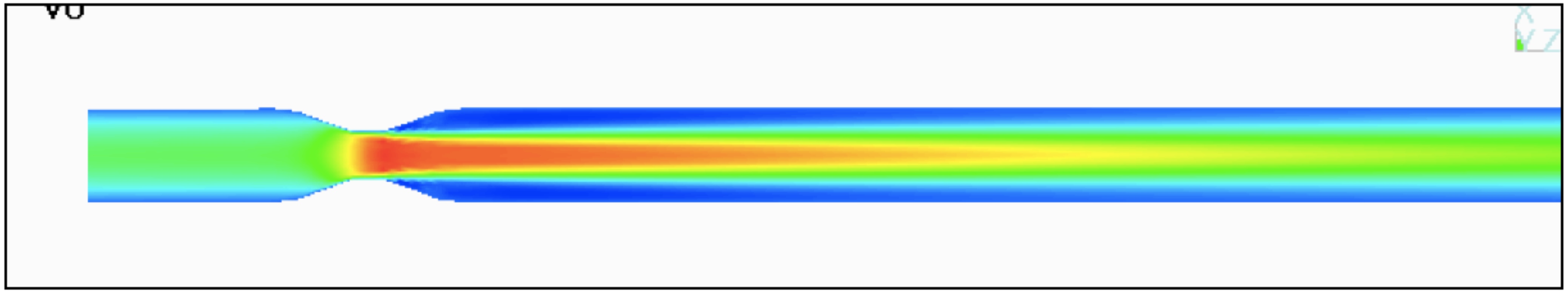
Fluid flow in a sudden axisymmetric constriction

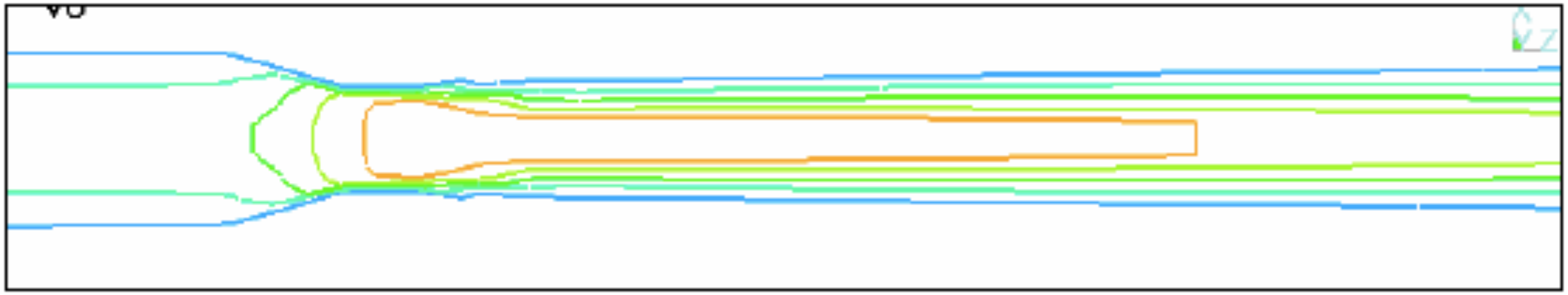
- *Our Strategy! (Reynolds number =300)*
  - We consider a symmetric Mesh
    1. By a Reynolds Continuation Procedure, we obtain a symetric solution
    2. Build a non-symmetric solution by imposing some boundary conditions
    3. Redo the simulation by setting this non-symmetric as initial data $\mathbf{u}_0$

Numerical Simulations with MEF++ Code

Fluid flow in a sudden axisymmetric constriction

1. *Symmetric Solution at Reynolds=300*



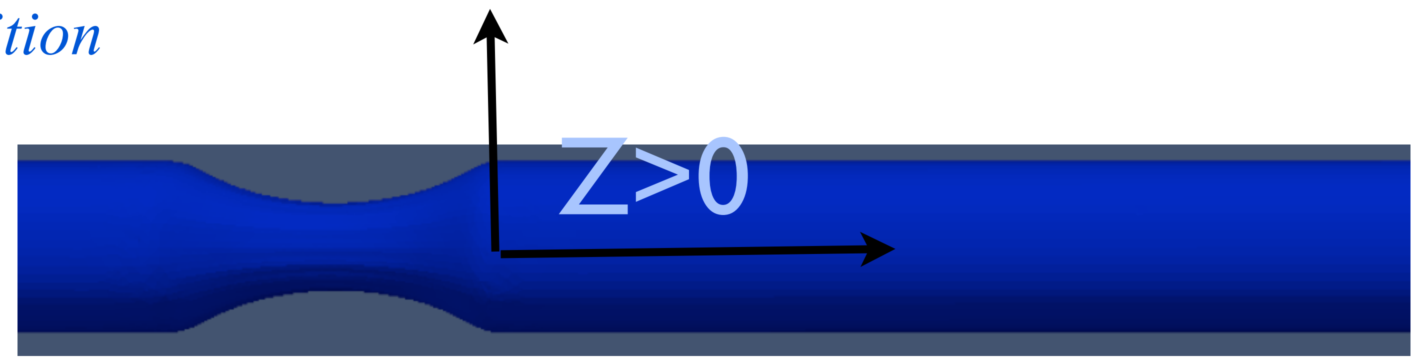


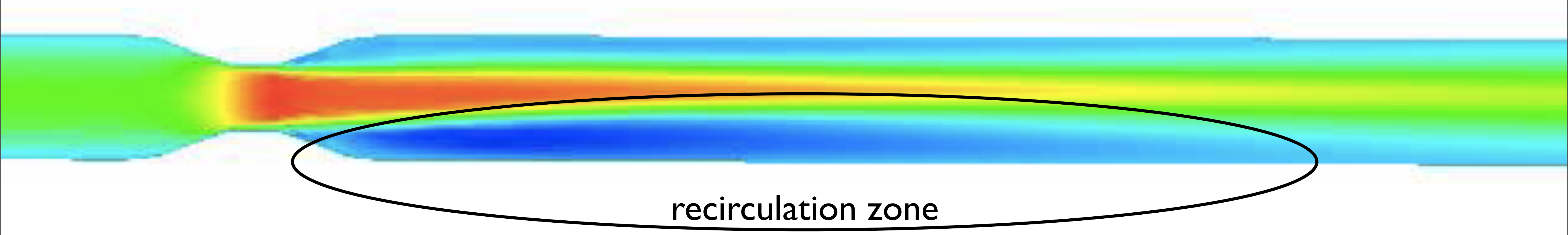
Numerical Simulations with MEF++ Code

Fluid flow in sudden axisymmetric constriction

2. Perturbation of the Initial Condition

- *Boundary condition:*
  $z>0$, $y>0$: $u(x,y,z) = (0,0, free)$ and
if $z<= 0$ or $y<= 0$: $u(x,y,z) =(0,0,0)$.

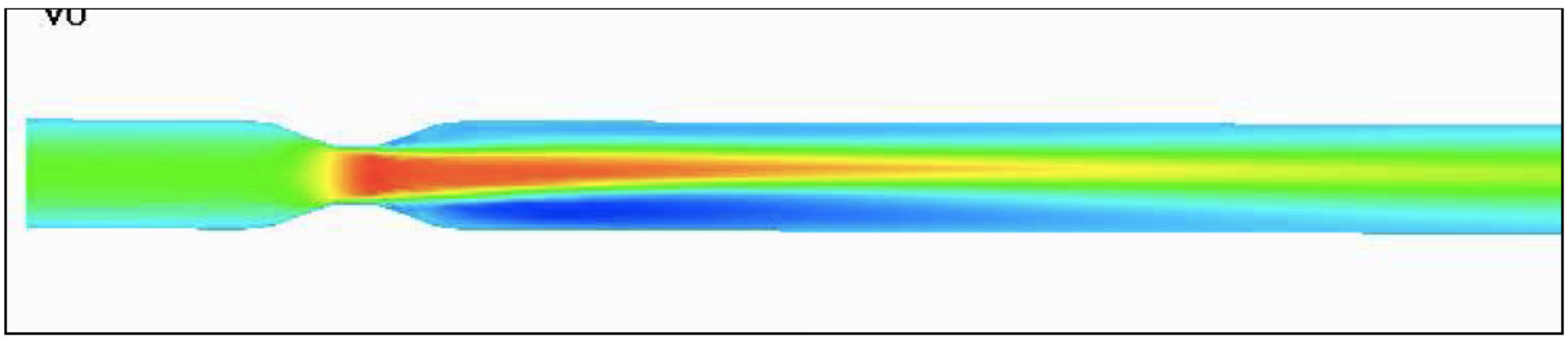


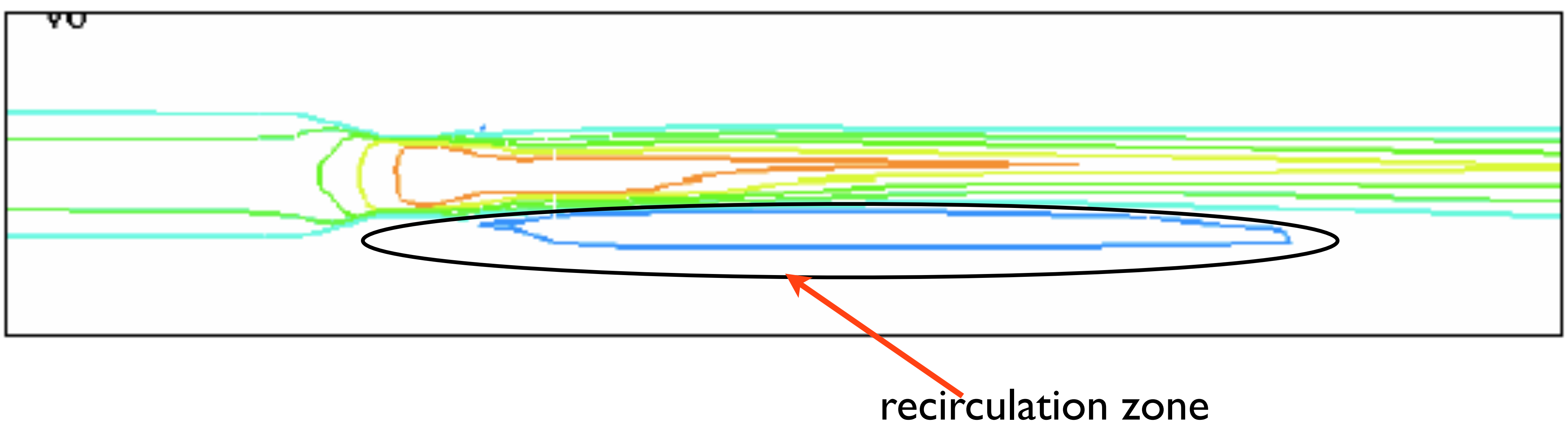


Numerical Simulations with MEF++ Code

Fluid flow in sudden axisymmetric constriction

3. Existence of the asymmetric solution at Reynolds =300





High Performance Computing
MEF++ //

colosse.clumeq.ca: 7900 Cores

High Performance Computing
MEF++ //

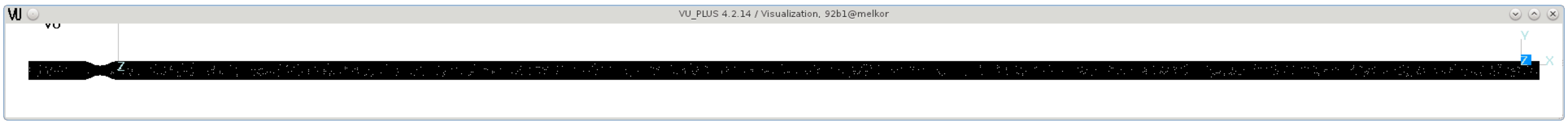
colosse.clumeq.ca: 7900 Cores

- ***4.5M DOF***

High Performance Computing MEF++ //

colosse.clumeq.ca: 7900 Cores

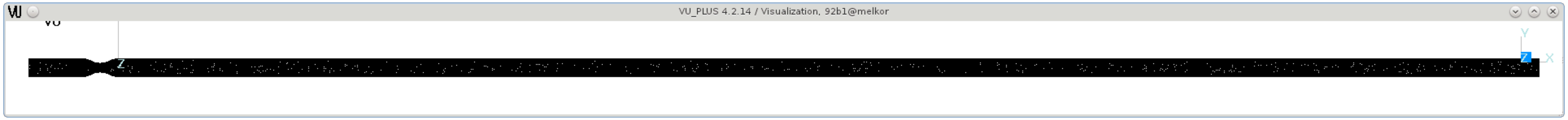
- ***4.5M DOF***

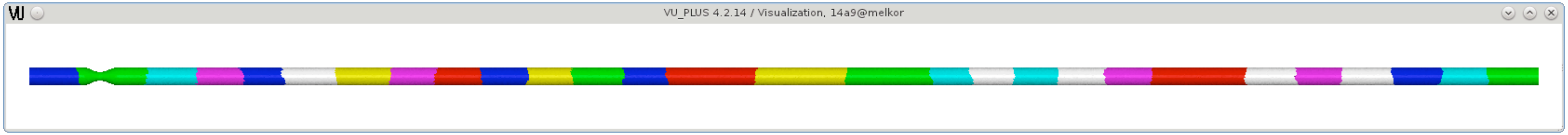


High Performance Computing MEF++ //

colosse.clumeq.ca: 7900 Cores

- *4.5M DOF*

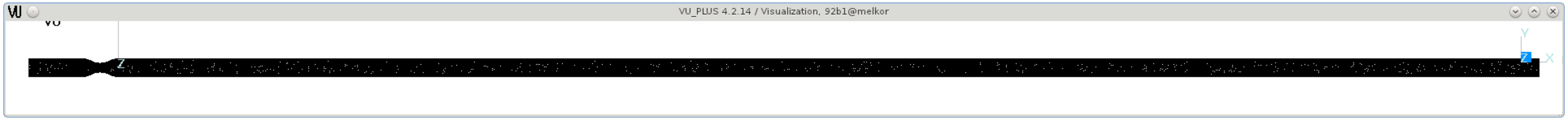


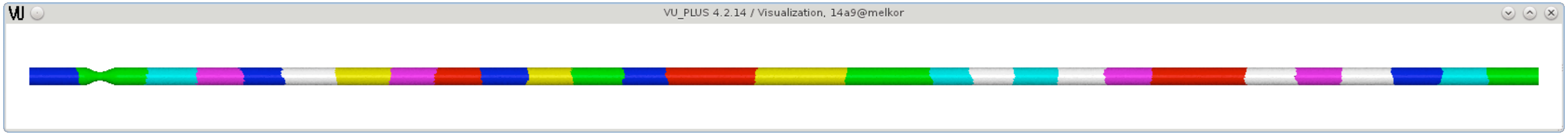


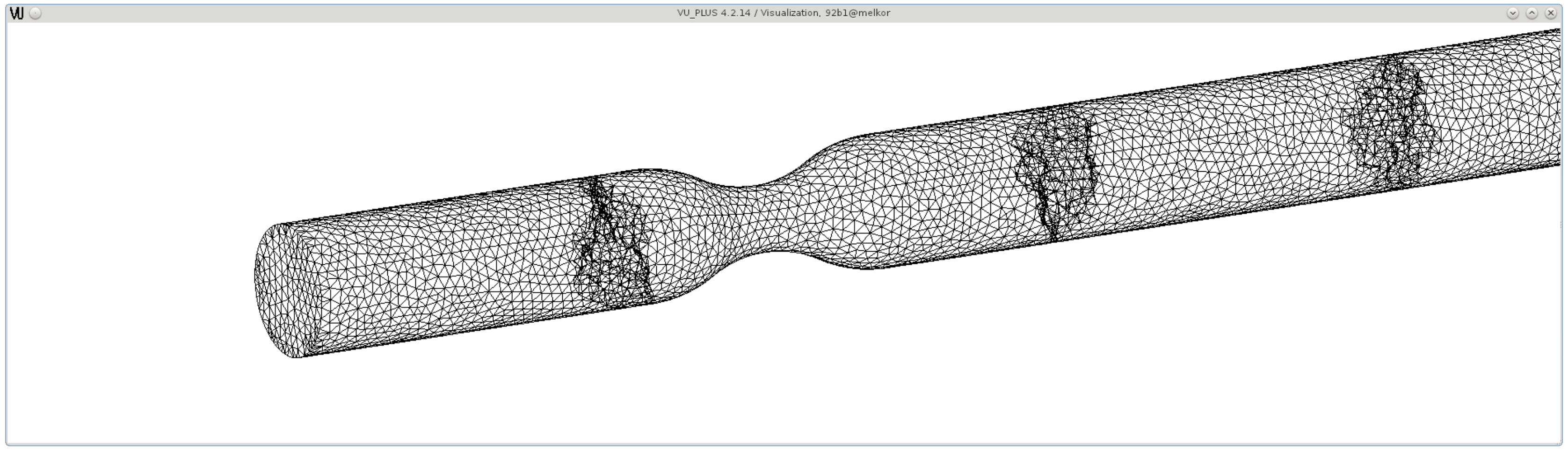
High Performance Computing
MEF++ //

colosse.clumeq.ca: 7900 Cores

- ***4.5M DOF***



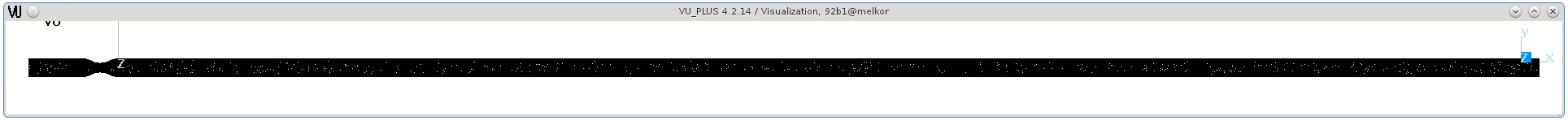


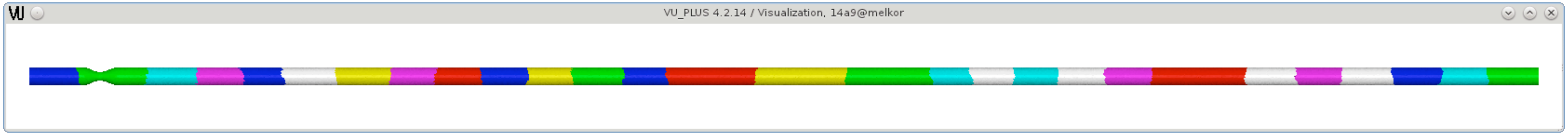


High Performance Computing
MEF++ //

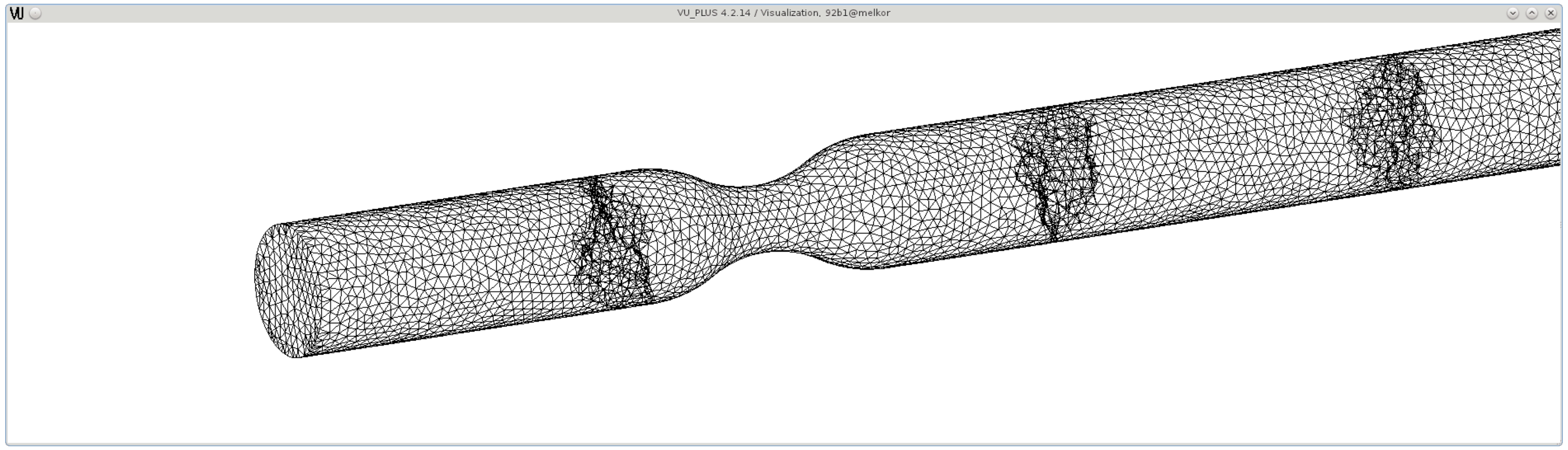
colosse.clumeq.ca: 7900 Cores

- ***4.5M DOF***





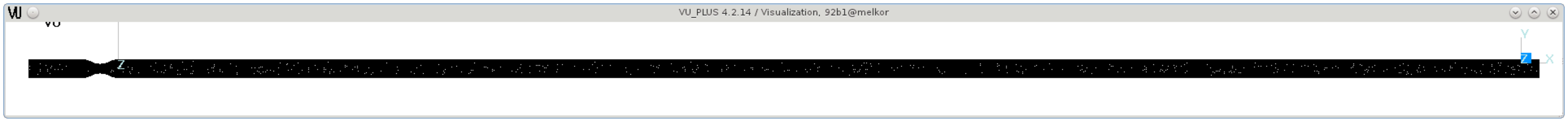
For 3 time steps, 10 Iterations of Newton

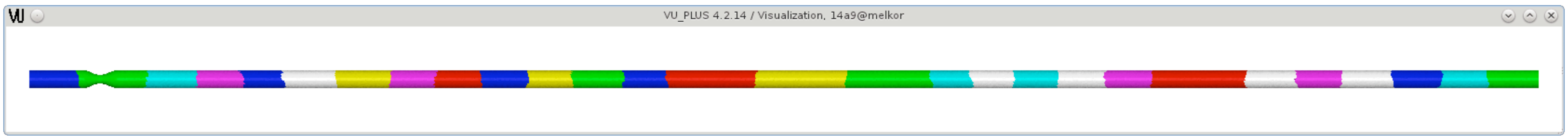


High Performance Computing
MEF++ //

colosse.clumeq.ca: 7900 Cores

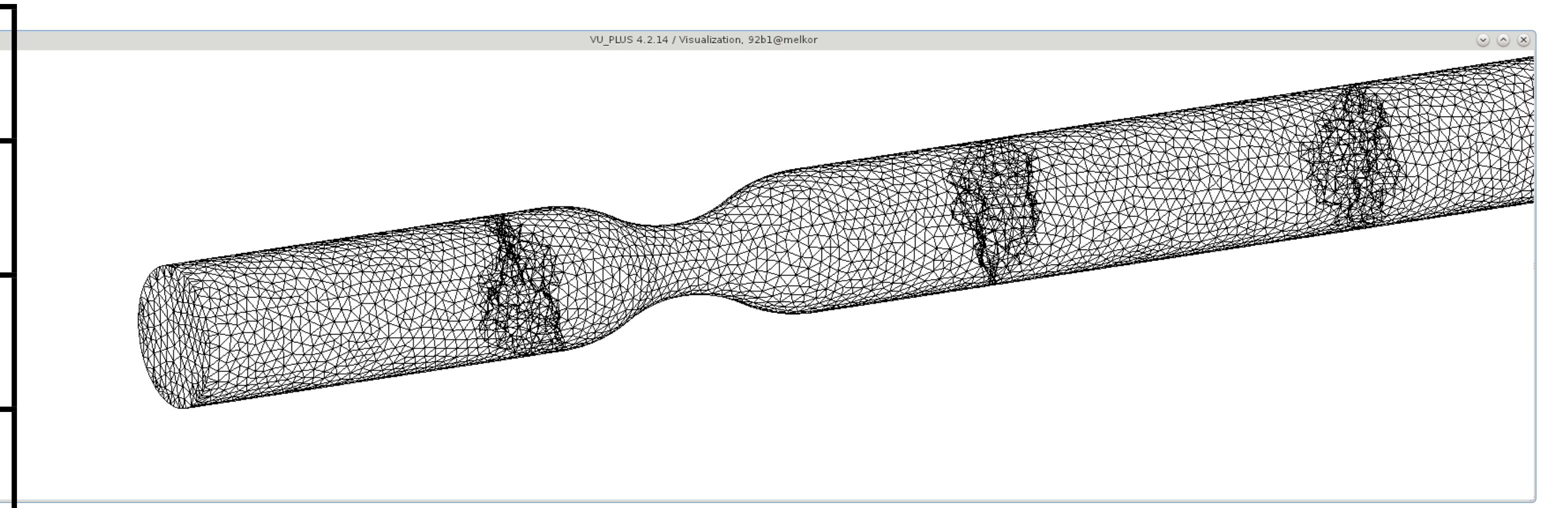
- ***4.5M DOF***





For 3 time steps, 10 Iterations of Newton

DOF	Nb_proc	Time(s)
1 403 216	2	2002
2 790 733	4	2394
4 929 523	8	2682
10 281 710	14	3539
47 083 332	64	5939



Conclusion and Ongoing Works

Conclusion and Ongoing Works

- Conclusion

Conclusion and Ongoing Works

- Conclusion

1. Robust Solver for Navier Stokes Equations
2. MEF++ Code is Parallel
3. We can simulate big problems

Conclusion and Ongoing Works

- Conclusion

  1. Robust Solver for Navier Stokes Equations
  2. MEF++ Code is Parallel
  3. We can simulate big problems

- Ongoing Works

Conclusion and Ongoing Works

- Conclusion

  1. Robust Solver for Navier Stokes Equations
  2. MEF++ Code is Parallel
  3. We can simulate big problems

- Ongoing Works

  1. Verify experimental results of Vetel and Garon (and Sherwin)
  2. Improvement (optimization) of the solver
  3. Fluid-Structure Interaction

Conclusion and Ongoing Works

- Conclusion

  1. Robust Solver for Navier Stokes Equations
  2. MEF++ Code is Parallel
  3. We can simulate big problems

- Ongoing Works

  1. Verify experimental results of Vetel and Garon (and Sherwin)
  2. Improvement (optimization) of the solver
  3. Fluid-Structure Interaction

Thank you!