Scalable Preconditioners for Navier–Stokes Equations in Hemodynamic Simulations

Simone Deparis, Gwenol Grandperrin, Alfio Quarteroni

MATHICSE - Chair of Modelling and Scientific Computing (CMCS)
EPFL, Lausanne, Switzerland

European Trilinos User Group

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

June 5th, 2012

CMCS

Acknowledgements

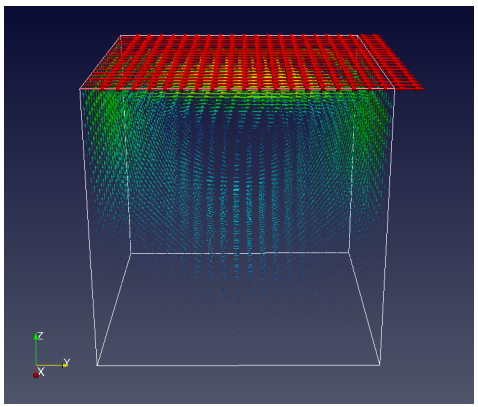

CSCS
Swiss National Supercomputing Centre

HP2C High Performance and High Productivity Computing

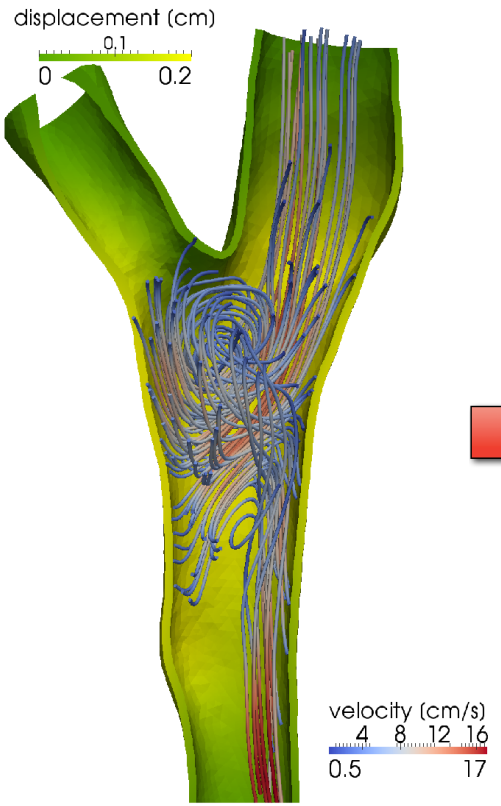
E. Cyr
J. Gaidamour
M. Heroux
M. Hoemmen
J. Hu
A. Salinger
C. Siefert

Michele Benzi
Paolo Crosetto
Cristiano Malossi
Tiziano Passerini
Radu Popescu
Samuel Quinodoz
Tim Robinson
Umberto Villa
Alessandro Veneziani
Zhen Wang
Andrew Wathen

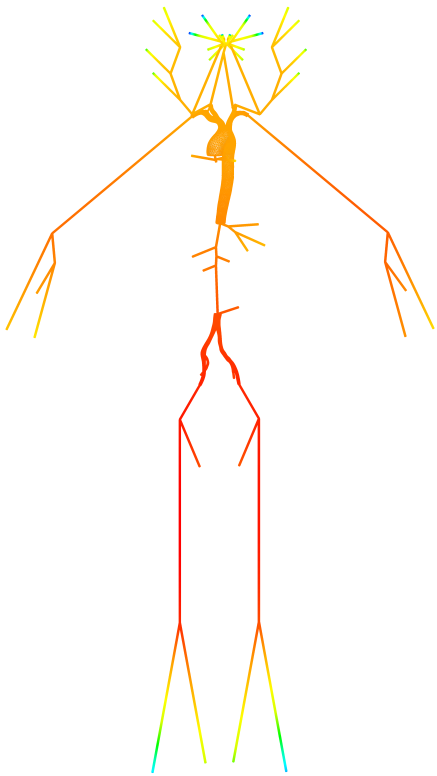
Motivation



N–S



FSI



Multiscale model

Malossi, Blanco, Deparis, Quarteroni. Algorithms for the partitioned solution of weakly-coupled fluid models. 2010. Submitted.

Crosetto, Deparis, Fourestey, Quarteroni. Parallel algorithms for fluid-structure interaction problems in haemodynamics. *SIAM J. Sci. Comput., 2011.*

Outline

- Metrics for parallel preconditioners
- Approximate preconditioners for the Navier–Stokes equations
- Experimental results
  - Weak and strong scalability analysis for assembling the preconditioner and solving the problem
  - Viscosity impact on the performances

Mathematical model

The Navier–Stokes equations for an incompressible viscous flow read:

$$\begin{aligned}
\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega\times(0,T] \\
\nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega\times(0,T] \\
\mathbf{u} &= \boldsymbol{\varphi} && \text{on } \Gamma_D\times(0,T] \\
\nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\mathbf{n} &= 0 && \text{on } \Gamma_N\times(0,T] \\
\mathbf{u} &= \mathbf{u}_0 && \text{at } t=0
\end{aligned}$$

where Γ_D and Γ_N are the Dirichlet and Neumann parts of the boundary respectively, $\mathbf{u}$ is the fluid velocity, p the pressure, ν the kinematic viscosity of the fluid, and $\mathbf{f}$ the external forces.

Mathematical model
Discretization

E.g. with semi-implicit Euler scheme:

$$\begin{aligned} \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1} - \nu \Delta \mathbf{u}^{n+1} + \nabla p^{n+1} &= \mathbf{f}^{n+1} && \text{in } \Omega \\ \nabla \cdot \mathbf{u}^{n+1} &= 0 && \text{in } \Omega \\ \mathbf{u}^{n+1} &= \boldsymbol{\varphi} && \text{on } \Gamma_D \\ \nu \frac{\partial \mathbf{u}^{n+1}}{\partial \mathbf{n}} - p^{n+1}\mathbf{n} &= 0 && \text{on } \Gamma_N \end{aligned}$$

FE discretization using $\mathbb{P}_2 - \mathbb{P}_1$ finite elements on unstructured grids:

$$\begin{pmatrix} F(\mathbf{U}^n) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{U}^{n+1} \\ \mathbf{P}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^{n+1}(\mathbf{U}^n) \\ \mathbf{0} \end{pmatrix}$$

Metrics for parallel preconditioners

First perspective: To solve large scale problems as efficiently as possible by parallel algorithms.

Definition (Strong scalability)

Let T_1 and T_P be the computational time to run an application with fixed amount of computational work using one and P processes respectively. An application is said to be strongly scalable if

$$T_P = \frac{T_1}{P}.$$

In particular, the preconditioned iterations of the numerical solver should be strongly scalable.

$\Rightarrow$ The preconditioner plays a key role in the scalability.

Metrics for parallel preconditioners

Second perspective: solving bigger and bigger problems while keeping the computational time constant, provided that suitable resources are available.

Definition (Weak scalability)

Let denote the workload of a given problem W_1 and W_2 be the workload to solve a given problem using P_1 and P_2 processes respectively such that

$$\frac{W_1}{P_1} = \frac{W_2}{P_2}.$$

An application is said to be weakly scalable if for any couple (W_1, P_1) and (W_2, P_2) the computational time of the application is the same.

Metrics for parallel preconditioners

Definition (Preconditioner scalability)

A preconditioner P of $\mathcal{A}$ is said to be scalable if the rate of convergence of the iterative method used to solve the preconditioned system

$$\mathcal{A}P^{-1}\mathbf{y} = \mathbf{b}, \qquad \mathbf{x} = P^{-1}\mathbf{y}.$$

does not deteriorate when the number of processes grows.

Definition (Preconditioner optimality)

A preconditioner is said to be optimal for a given problem $\mathcal{A} \in \mathbb{R}^{N\times N}$ if for

1. the number of preconditioned iterations is bounded with respect to the dimension N of $\mathcal{A}$;
2. the total computational costs to build and use the preconditioner increase linearly wrt the dimension N of $\mathcal{A}$.

Metrics for parallel preconditioners

Definition (Preconditioner robustness)

A preconditioner is said to be *robust* if the convergence rate of the iterative method does not depend on the physical parameters (e.g. viscosity) that characterize the PDE.

This property ensures that the preconditioner handles a wide range of Reynolds numbers; for medical simulations the Navier–Stokes equations have to be solved for a wide range of Reynolds from $\Re e = 1$ to $\Re e = 4000$.

David N. Ku. Blood flow in arteries. *Annu. Rev. Fluid Mech., 1997.*

Designing a Navier–Stokes preconditioner for HPC

Dream list:

1. The algorithms involved to build and apply the preconditioner must be weakly and strongly scalable.
2. The preconditioner should be optimal.
3. The preconditioner should be scalable.
4. The preconditioner should be robust with respect to the viscosity ν.

Designing a Navier–Stokes preconditioner for HPC

The matrix of the linearized N–S system after discretization can be factorized as, e.g.

$$\mathcal{A} = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix} \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}$$

where $S = BF^{-1}B^T$ is the Schur complement.

Idea: Exploit the block structure of the problem matrix:
We consider the following factor as right preconditioner

$$P = \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}$$

One can prove that PGMRES converges in at most 2 iterations!

Murphy, Golub, Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput., 2000.*

Quarteroni, Saleri, Veneziani. Factorization methods for the numerical approximation of Navier–Stokes equations. *Comput. Methods Appl. Mech. Engrg., 2000.*

Elman, Howle, Shadid, Shuttleworth, Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *J. Comput. Phys., 227(3):1790–1808, 2008.*

Classical preconditioners for N–S

- SIMPLE

$$P_{SIMPLE}^{-1} = \begin{pmatrix} D^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & \frac{1}{\alpha} I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\tilde{S}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & I \end{pmatrix} \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix},$$

where $\alpha \in (0, 1]$ is a damping parameter and $\tilde{S} = BD^{-1}B^T$.

> Patankar, Spalding. A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows. *International J. on Heat and Mass Transfer, 15:1787–1806, 1972.*

- Yosida

$$P_{Yosida}^{-1} = \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & I \end{pmatrix} \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix},$$

with $S = \Delta t B M_{\mathbf{u}}^{-1} B^T$.

> Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. Analysis of the Yosida method for the incompressible Navier–Stokes equations. *J. Math. Pures Appl., 1999.*

- PCD

$$P_{PCD}^{-1} = \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -M_p^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & F_p \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A_p^{-1} \end{pmatrix}.$$

> Silvester, Elman, Kay, Wathen. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comput. Appl. Math., 2001.*

> Kay, Loghin, Wathen. A preconditioner for the steady-state Navier–Stokes equations. *SIAM J. Sci. Comput., 2002.*

Approximate preconditioners for N–S

- Approximate SIMPLE (aSIMPLE)

$$P_{aSIMPLE}^{-1} = \begin{pmatrix} D^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} D & 0 \\ 0 & \frac{1}{\alpha} I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\hat{\tilde{S}}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & I \end{pmatrix} \begin{pmatrix} \hat{F}^{-1} & 0 \\ 0 & I \end{pmatrix},$$

where $\alpha \in (0, 1]$ is a damping parameter and $\tilde{S} = BD^{-1}B^T$.

- Approximate Yosida (aYosida)

$$P_{aYosida}^{-1} = \begin{pmatrix} \hat{F}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\hat{S}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B & I \end{pmatrix} \begin{pmatrix} \hat{F}^{-1} & 0 \\ 0 & I \end{pmatrix},$$

with $S = \Delta t B M_{\mathbf{u},\ell}^{-1} B^T$.

- Approximate PCD (aPCD)

$$P_{aPCD}^{-1} = \begin{pmatrix} \hat{F}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\hat{M}_p^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & F_p \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \hat{A}_p^{-1} \end{pmatrix}.$$

where $\hat{}$ denotes the use of a preconditioner to approximate the inverse

Inverses approximation

Details on the preconditioners

- The algebraic additive Schwarz (AS) preconditioner used has 2 layers of overlap, the subdomain problems are solved with exact factorization (with the `Amesos` KLU method in the `Trilinos` library) [Sala, Heroux, Sandia Report, 2005].
- The multilevel preconditioners and the 1-level additive Schwarz are provided by `Ifpack` and `ML` from `Trilinos`.
- The coarsening for the multilevels preconditioners is obtained via aggregations using `METIS`/`ParMETIS`.

Quarteroni, Valli. Domain decomposition methods for partial differential equations. *Numerical Mathematics and Scientific Computation. The Clarendon Press Oxford University Press, New York, 1999.*

Inverses approximation

Details on the preconditioners

- $\hat{F}^{-1}$ and $(BM_{\mathbf{u},\ell}^{-1}B^T)^{-1}$ are replaced with a 2-level Schwarz preconditioner; the first level is applied without overlap with a coarse grid correction. The subdomain problems are solved using exact factorization.
- $\hat{A}_p^{-1}$ and $(BD^{-1}B^T)^{-1}$ are replaced using a V-cycle AMG with 2 sweeps of symmetric Gauss-Seidel as smoother (presmoothing only), exact factorization for the coarsest level. The AMG is implemented in the `ML` package in `Trilinos` [Sala, TOMS, 2006], [Gee, Kuttler, Wall, IJNME, 2010].
- $\hat{M}_p^{-1}$ is replaced by the inverse of the diagonal lumped mass matrix.

Numerical results
Software implementation

- Finite elements library written in C++ (80'000 lines)
- LGPL license
- Used in the Mathcard European project (http://mathcard.eu), and the HP2C project.

`LifeV` relies on several external libraries:

- `ParMetis`/`Metis` for parallel mesh partitioning;
- `Trilinos` (10.8) for matrix and vector parallel distribution, for parallel solvers, and for parallel preconditioners;
- `Boost`, `SuiteSparse` (`UMFPACK`), `HDF5`.

Numerical results
Preconditioners in LifeV

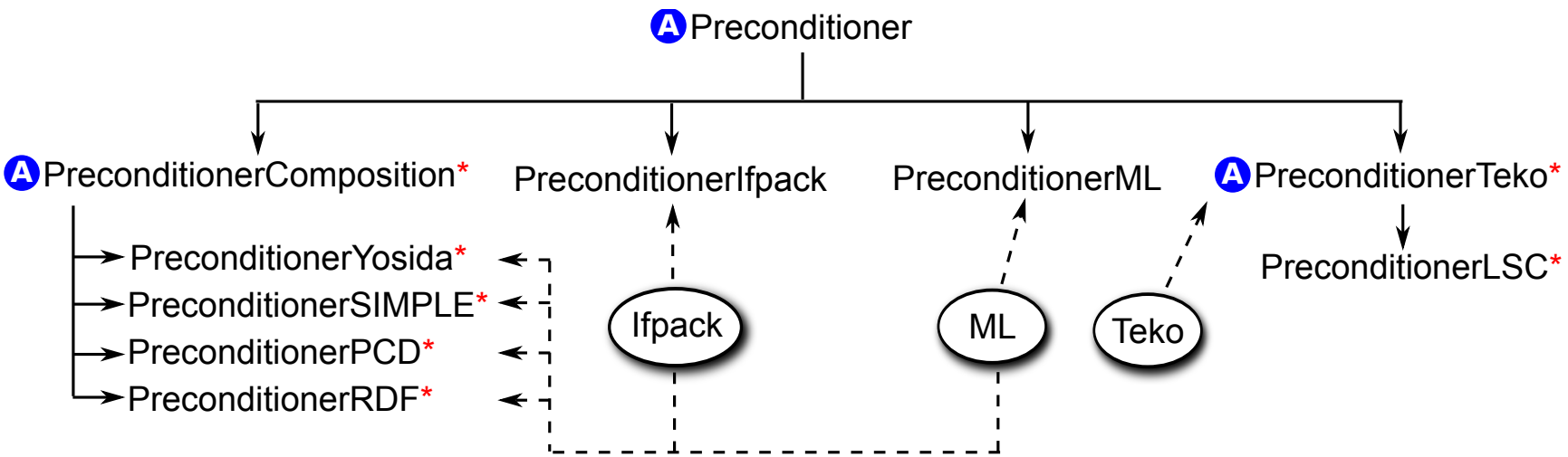


Figure: Overview of the preconditioners in LifeV

A = Abstract classes

= Trilinos packages

* Developed in the context of the HP2C project

Numerical results
Preconditioners in LifeV

The `PreconditionerComposition` class exploits the block structure of the FE matrix $\mathcal{A}$ to create preconditioners.

$$\mathcal{A} = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}$$

The class is able to

- manage composition of operators obtained by factorizing the matrix $\mathcal{A}$.
- replace the inverses of operators by preconditioners (e.g. multigrid preconditioners, additive Schwarz preconditioner).

Numerical results
Simulation protocol

- Linear problem solved at each timestep with preconditioned GMRES provided by the `Belos` package in `Trilinos`;
- Stopping criteria based on the residual scaled by the right hand side:

$$\|\mathbf{b} - \mathcal{A}\mathbf{x}_k\|_2 \leq 10^{-6}\|\mathbf{b}\|_2.$$

Numerical results

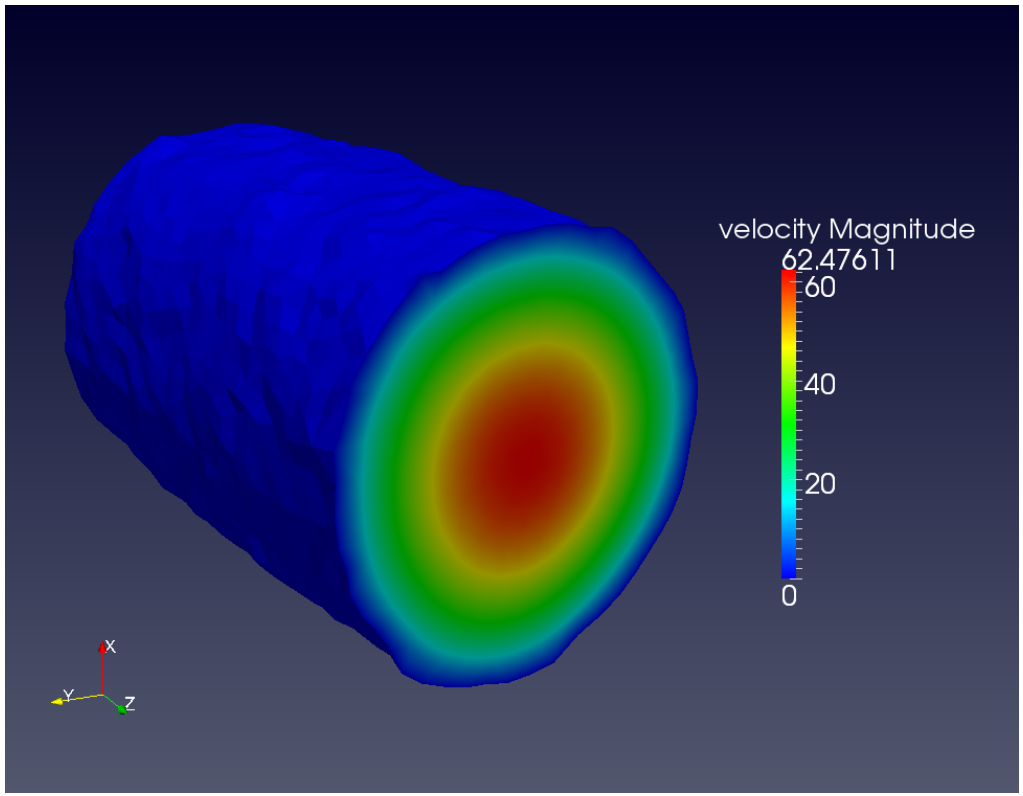
CSCS
Swiss National Supercomputing Centre

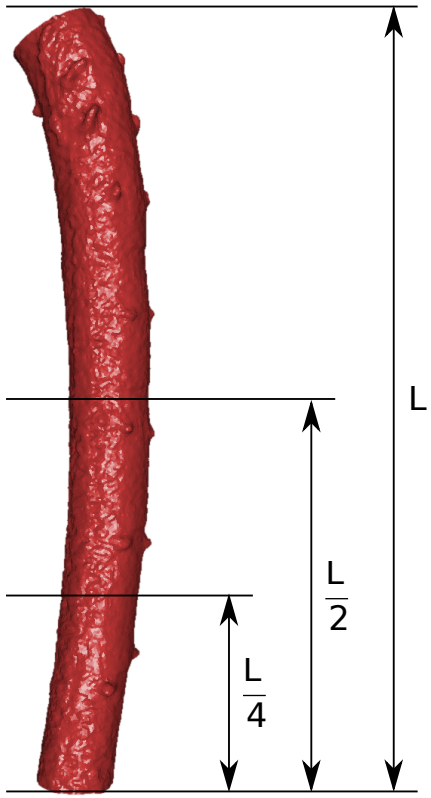
The simulations were run on the Monte Rosa Cray XE6 at the CSCS, Lugano, Switzerland.

Number of nodes	1496
Number of processors per node	2x16-core AMD Interlagos
Processors frequency	2.1 GHz
Processors shared memory	32 GB DDR3
Peak performance	402 Teraflop/s.
Network	Gemini 3D torus

Blood vessel benchmark

Weak scalability of the preconditioners

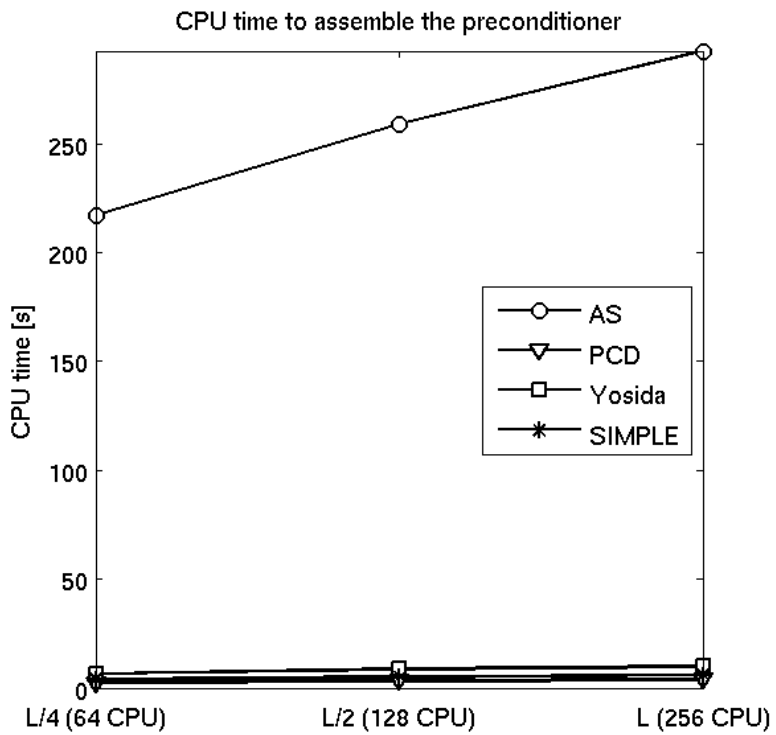




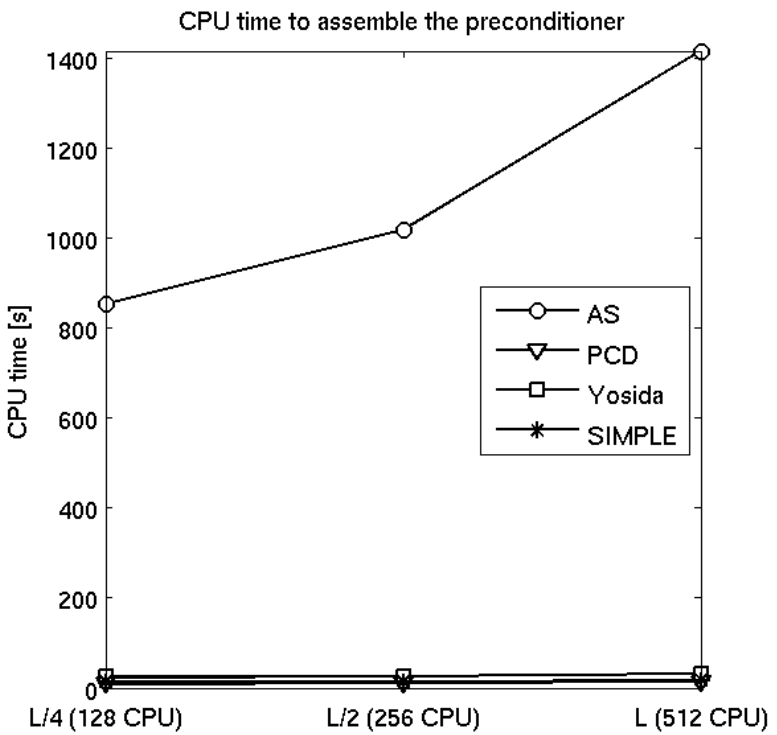
Number of DoFs	$L/4$	$L/2$	L
Coarse	512,747	1,079,563	2,363,158
Fine	2,000,361	4,256,516	9,208,310

Blood vessel benchmark

Weak scalability of the preconditioners



(a) Coarse mesh (max. 256 CPU)

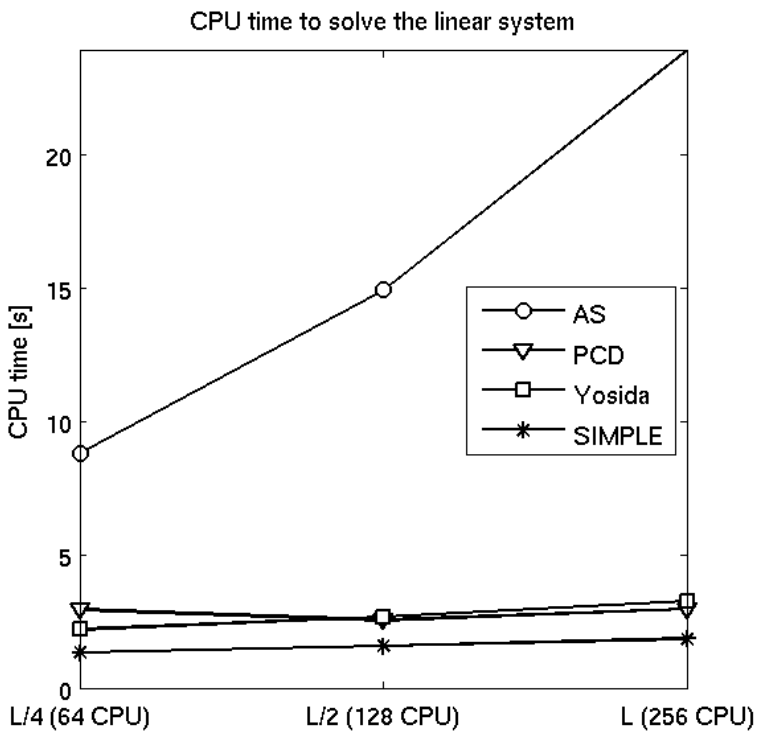


(b) Fine mesh (max. 512 CPU)

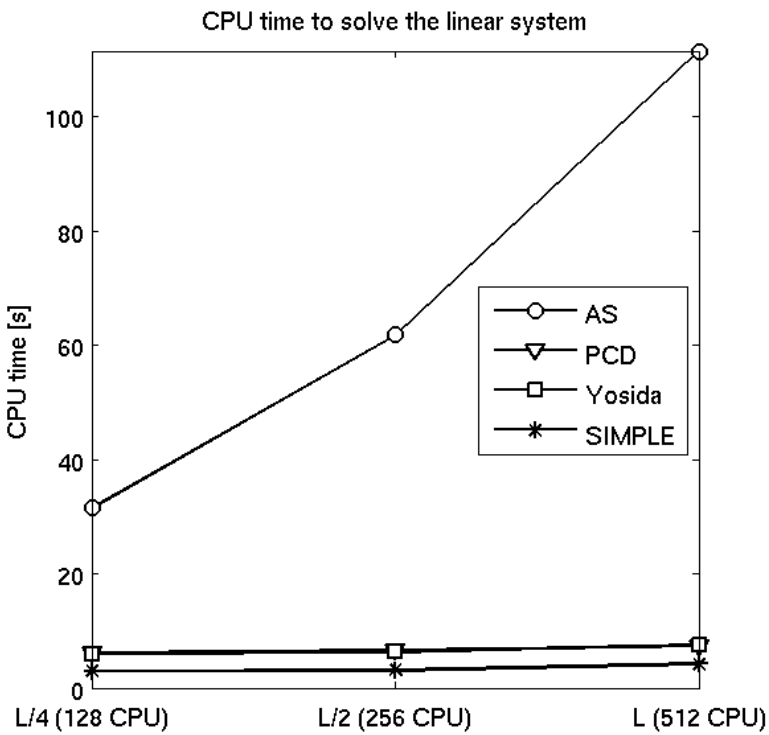
Figure: CPU time to assemble the preconditioner

Blood vessel benchmark

Weak scalability of the preconditioners



(a) Coarse mesh (max. 256 CPU)

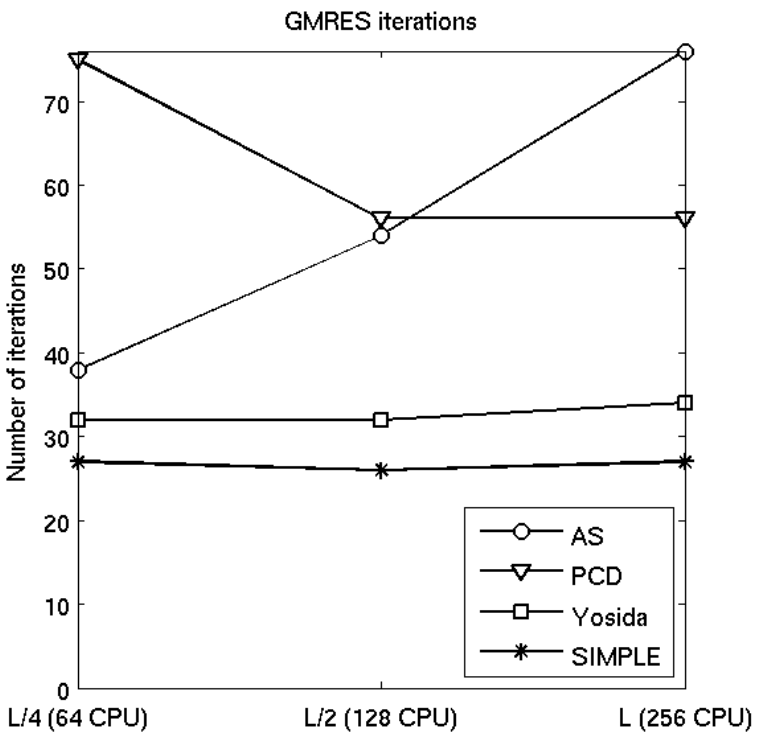


(b) Fine mesh (max. 512 CPU)

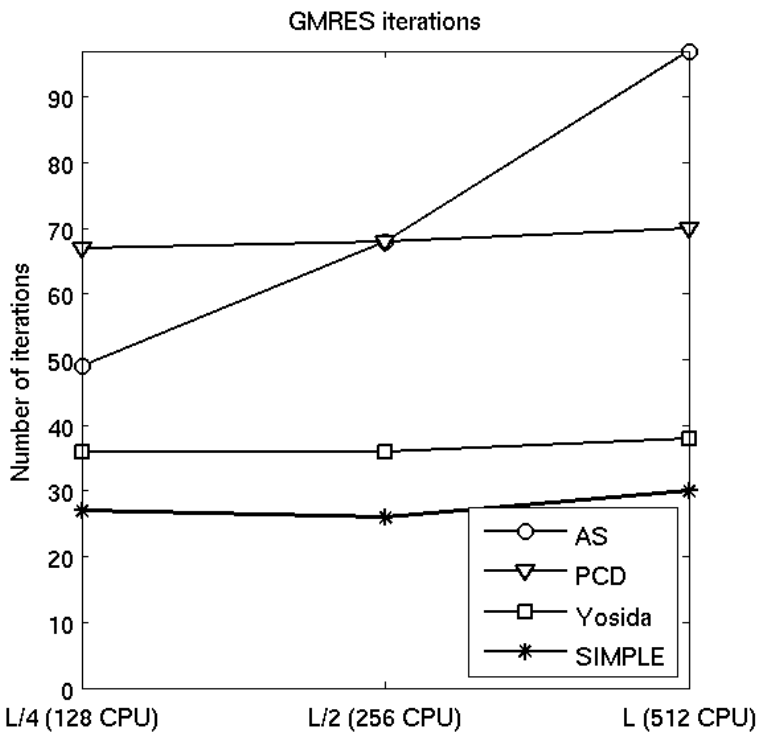
Figure: CPU time to solve the linear system

Blood vessel benchmark

Weak scalability of the preconditioners



(a) Coarse mesh (max. 256 CPU)

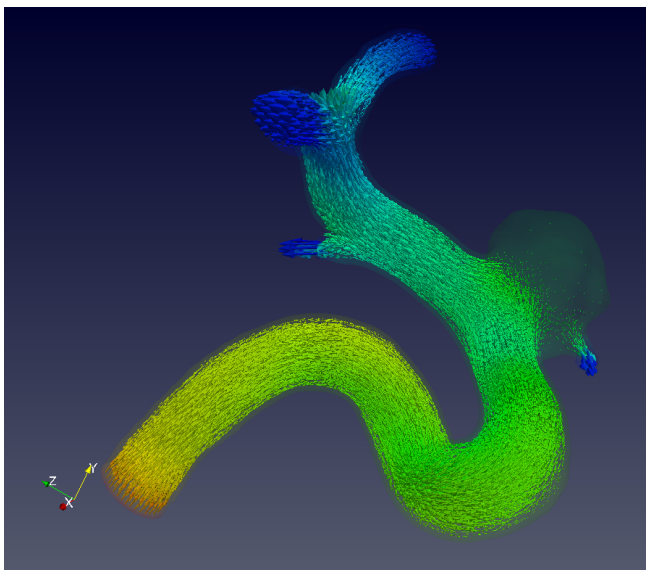


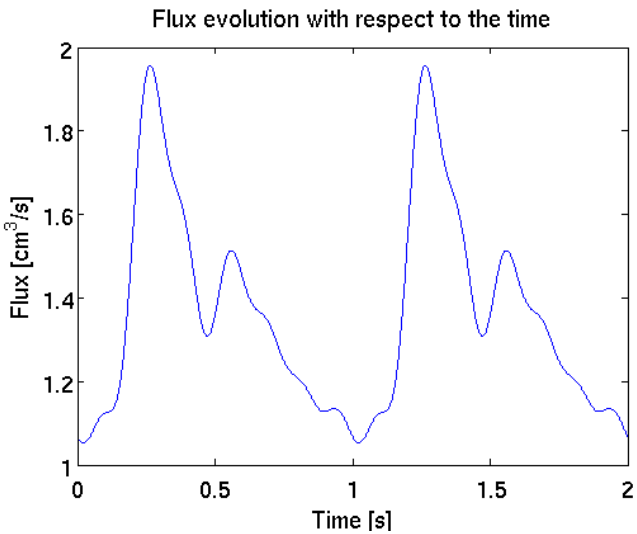
(b) Fine mesh (max. 512 CPU)

Figure: GMRES iterations

Blood-flow in rigid geometry

All our preconditioners are tested and tuned on a benchmark relevant for medical applications ($Re = 400$).

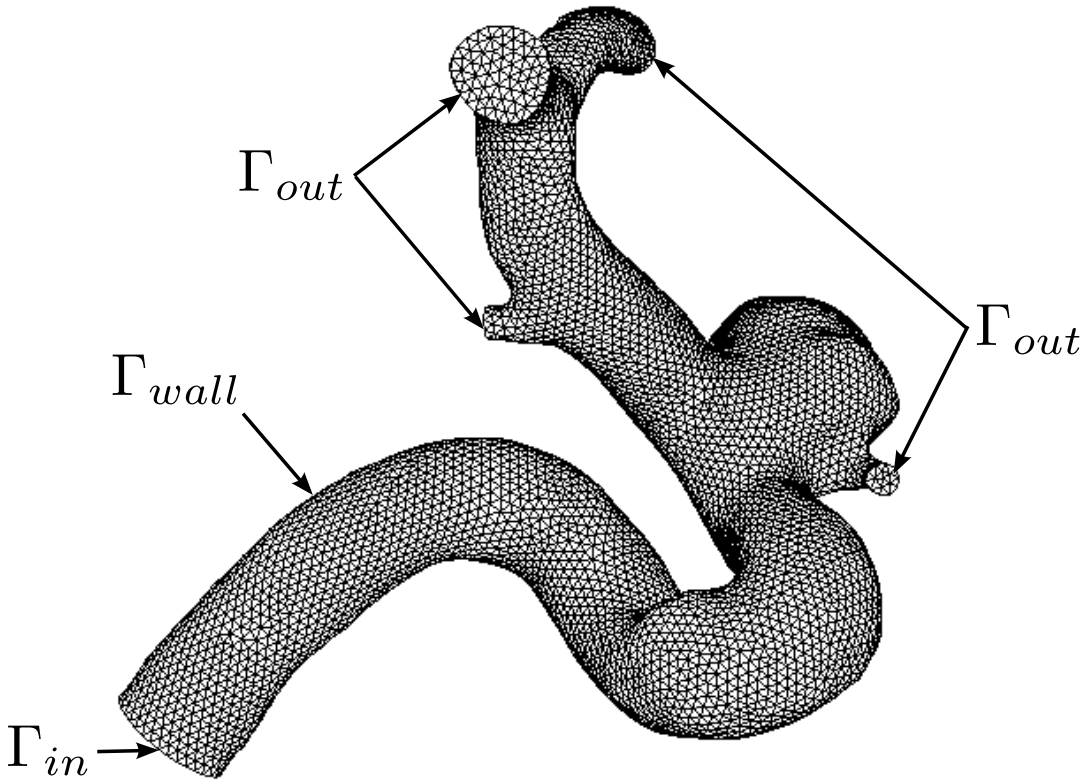




Mesh	Velocity DoFs	Pressure DoFs	h_{min}	h_{max}	h_{av}
Coarse	597,093	27,242	0.015	0.059	0.035
Medium	4,557,963	199,031	0.005	0.051	0.018
Fine	35,604,675	1,519,321	0.0026	0.0277	0.0097

Baek, Jayaraman, Richardson, Karniadakis. Flow instability and wall shear stress variation in intracranial aneurysms. *J R Soc Interface*, 2010.

Blood-flow in rigid geometry

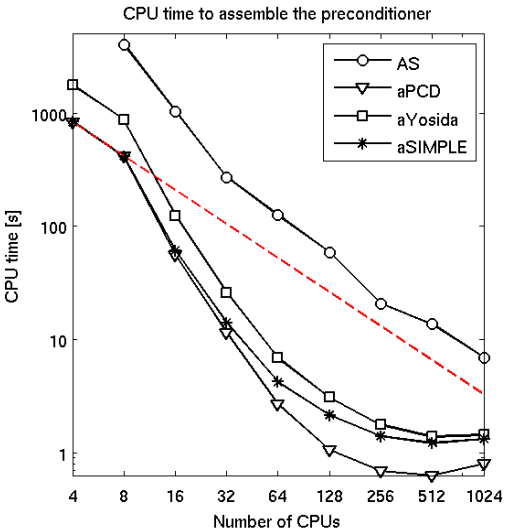


$$
\begin{aligned}
\mathbf{u} &= 0 && \text{on } \Gamma_{wall}, \\
\mathbf{u} &= \varphi_{flux}\mathbf{n} && \text{on } \Gamma_{in}, \\
\nu\frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} &= 0 && \text{on } \Gamma_{out},
\end{aligned}
$$

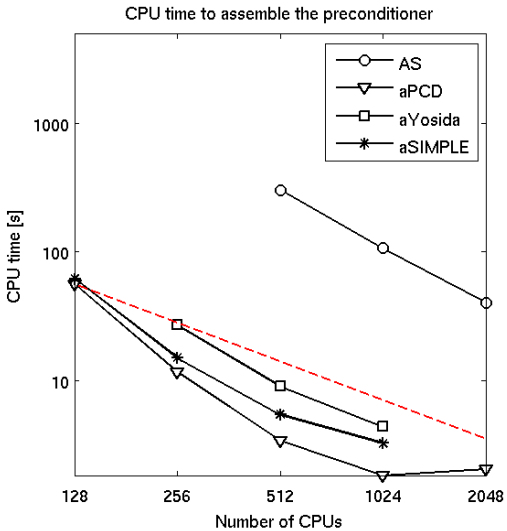
Blood-flow in rigid geometry

Preconditioner build

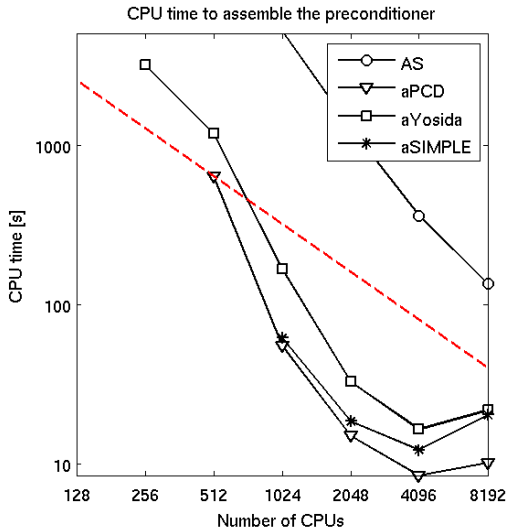
Coarse
max. 1024 CPU



Medium
max. 2048 CPU



Fine
max. 8192 CPU

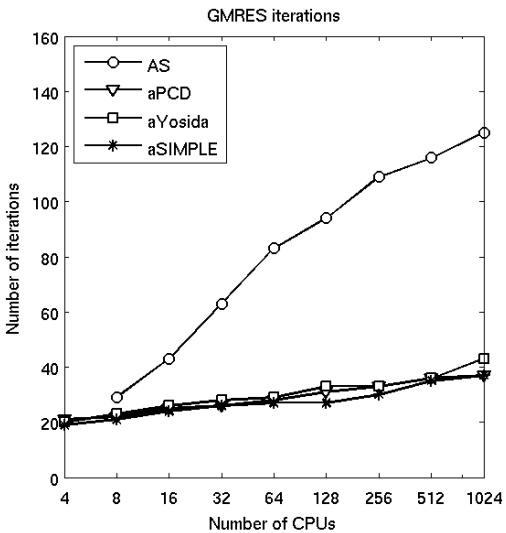


- The curves are superlinear due to the computation of the local LU factorizations.
- When the assembly time goes below a given threshold, the communication time overcomes the computation time for aPCD, aSIMPLE, and aYosida.
- The AS preconditioner is clearly longer to build.

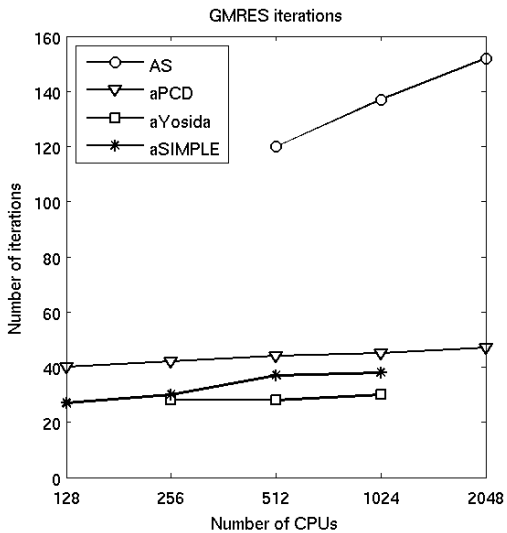
Blood-flow in rigid geometry

GMRES iterations

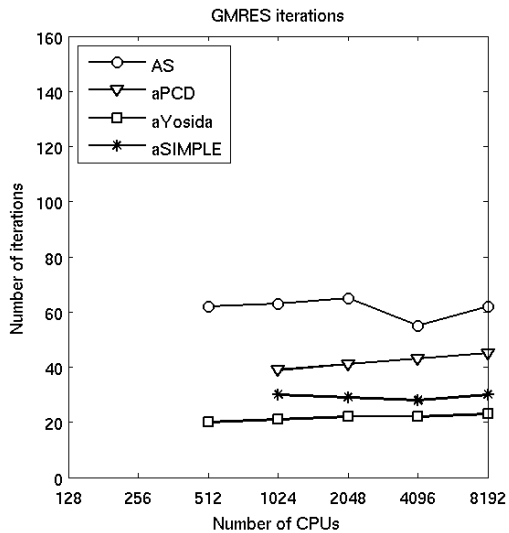
Coarse
max. 1024 CPU



Medium
max. 2048 CPU



Fine
max. 8192 CPU

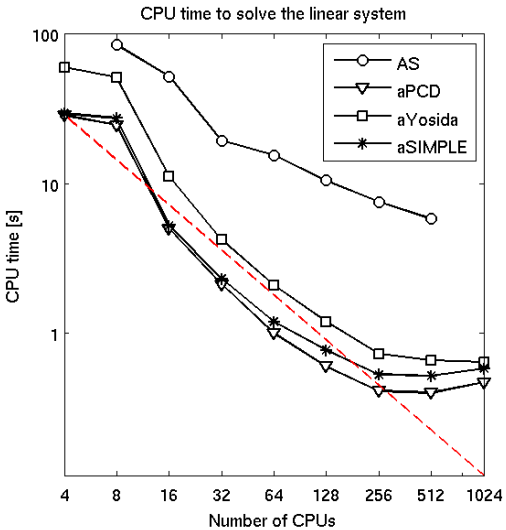


- If the fine mesh is fine enough, the aPCD, aYosida, and aSIMPLE are scalable (flat curves).
- AS requires a further mesh refinement to become scalable.
- GMRES converges slower when the AS preconditioner is used.

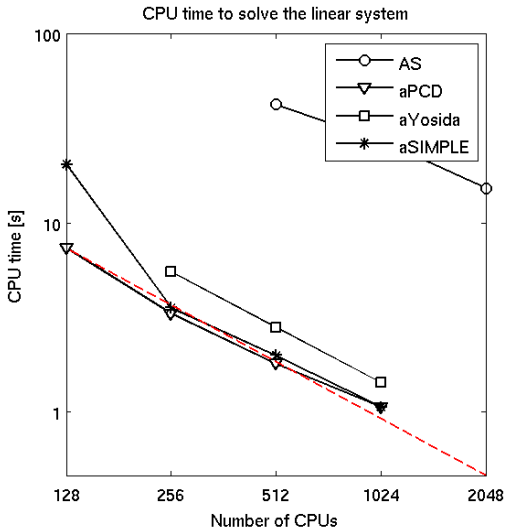
Blood-flow in rigid geometry

Time to solve the linear system

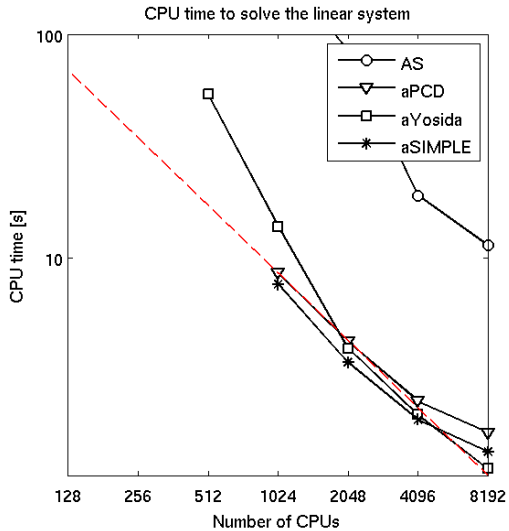
Coarse
max. 1024 CPU



Medium
max. 2048 CPU



Fine
max. 8192 CPU

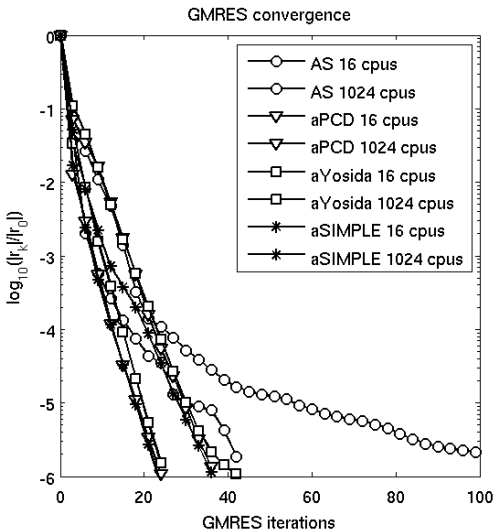


- For the coarse mesh, the AS prec. is not strongly scalable.
- Under ~ 1 s. the communication time overcomes the computation time for aPCD, aSIMPLE, and aYosida (coarse mesh)
- For the medium and fine meshes, the preconditioners are strongly scalable.

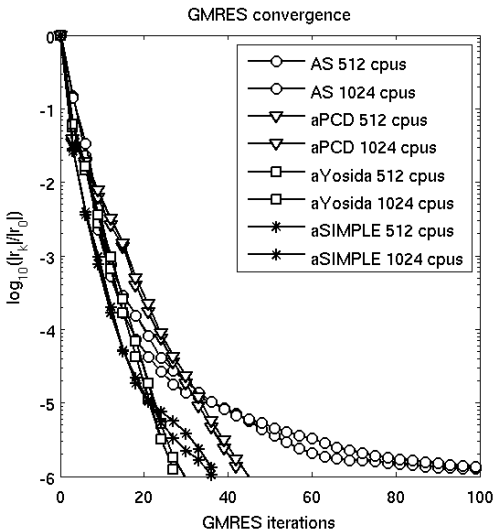
Blood-flow in rigid geometry

GMRES convergence

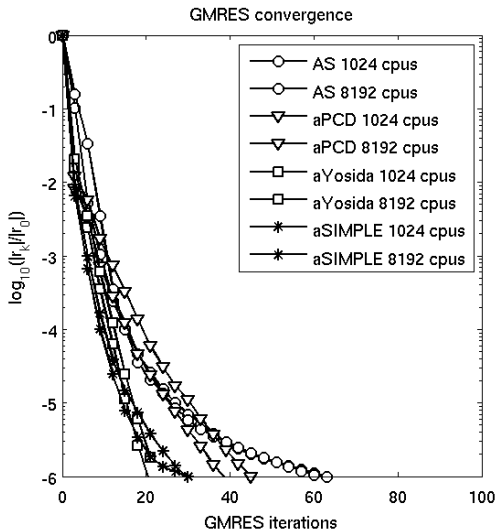
Coarse
16 vs 1024 CPU



Medium
512 vs 1024 CPU



Fine
1024 vs 8192 CPU

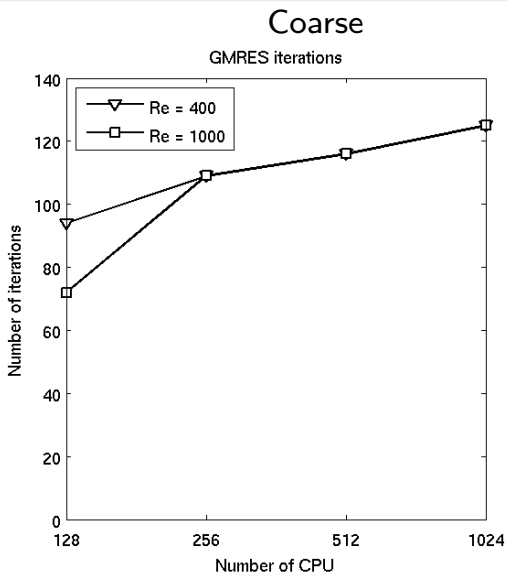


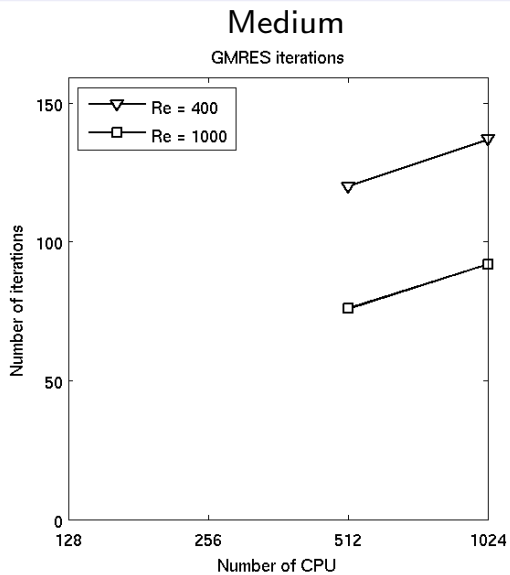
- For aPCD, aYosida, and aSIMPLE, GMRES converges very quickly for all meshes.
- In the case of the AS prec., we observe that using a finer mesh is crucial to obtain a fast convergence rate (typically the size of the subdomains contains more unkowns).

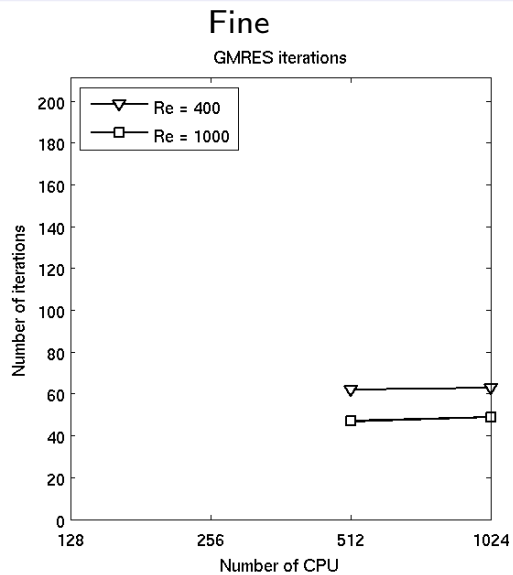
Blood-flow in rigid geometry

Sensitivity to the fluid viscosity

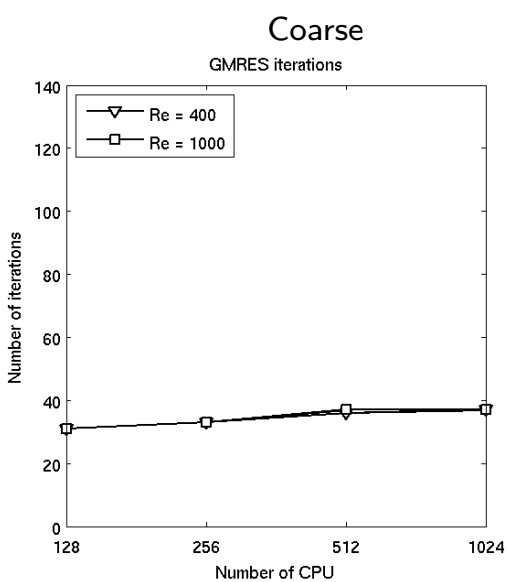
- AS

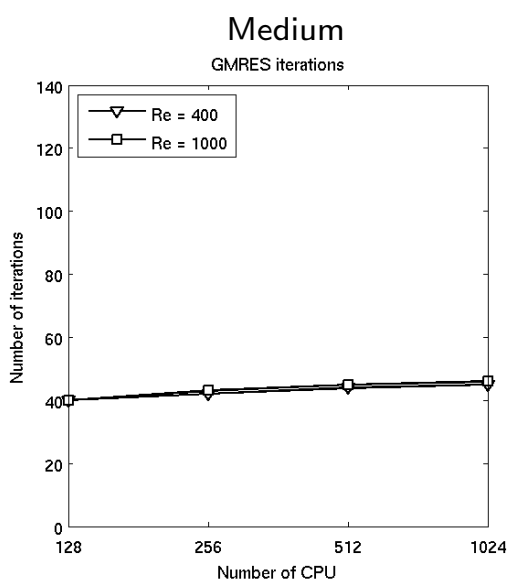


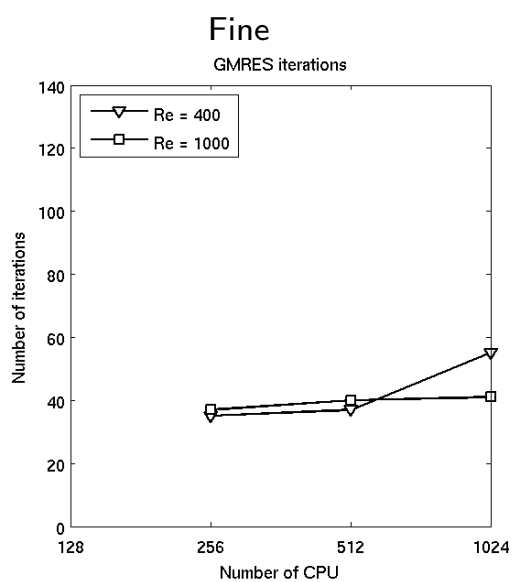




- aPCD



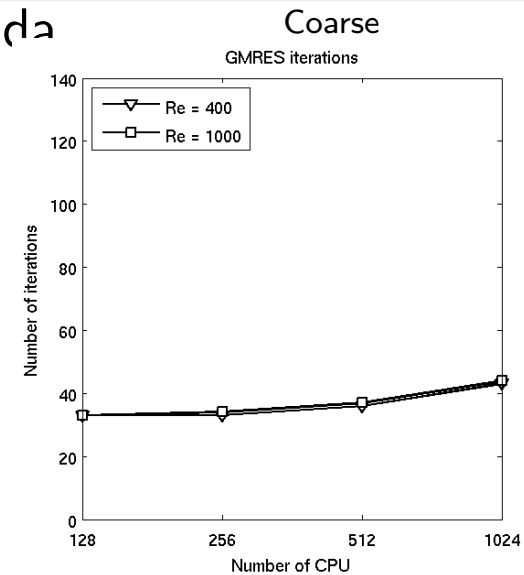


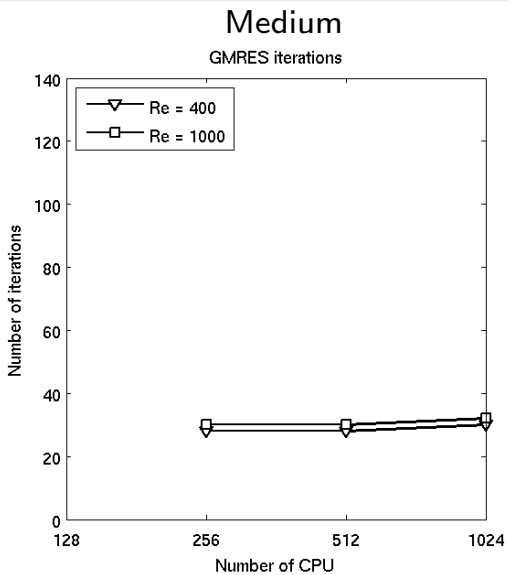


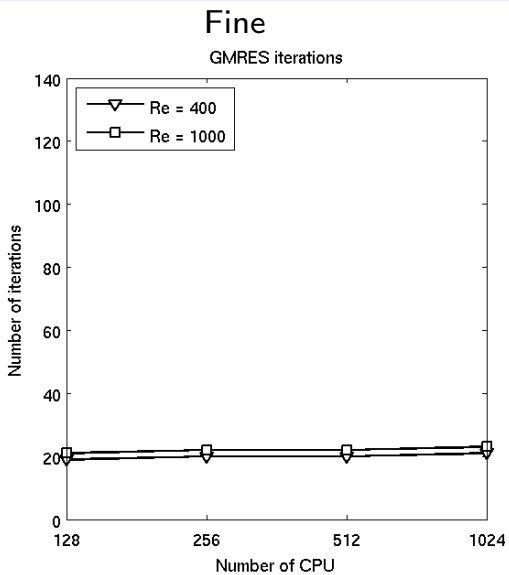
Blood-flow in rigid geometry

Sensitivity to the fluid viscosity

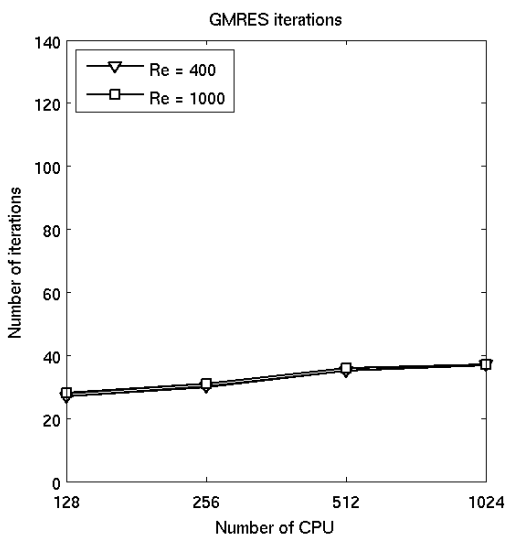
- aYosida

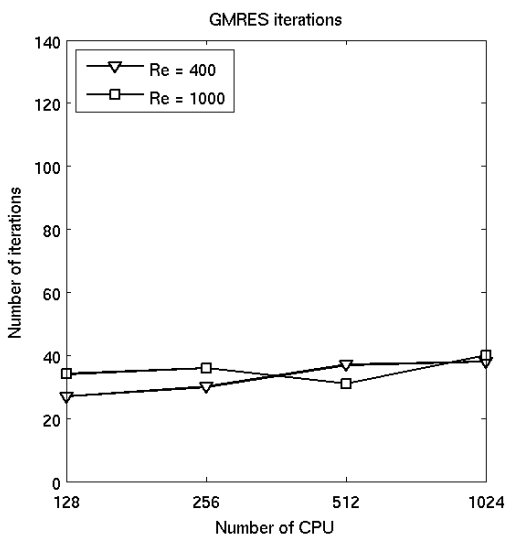


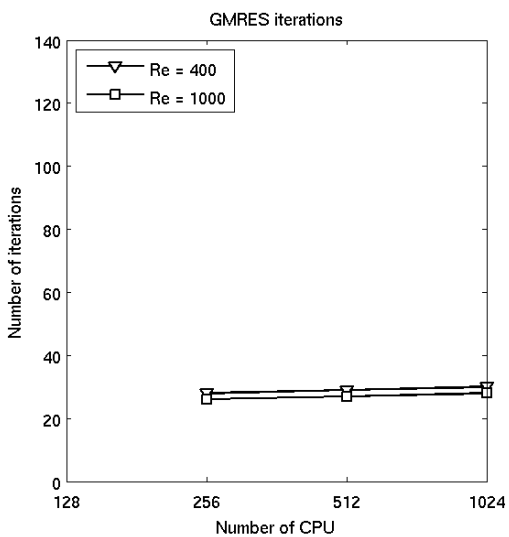




- aSIMPLE







Conclusion and ongoing work

- We developed preconditioners for solving Hemodynamic simulations.
- We tested the weak and strong scalability of our algorithms.
- The proposed preconditioners are scalable (i.e. number of iterations remains constant wrt the number of processors).

Ongoing work

- Integration of the aPCD, aYosida, and aSIMPLE into FSI preconditioners.
- Test the benefits of the preconditioners in a multiscale simulations.