

TOWARDS PARTITIONED FLUID-STRUCTURE INTERACTION ON MASSIVELY PARALLEL SYSTEMS

BENJAMIN UEKERMANN*, JUAN CARLOS CAJAS[†], BERNHARD
GATZHAMMER**, GUILLAUME HOUZEAUX[†], MIRIAM MEHL[‡] AND
MARIANO VÁZQUEZ[†]

*Institute for Advanced Study (IAS), Technische Universität München
Lichtenbergstraße 2a, 85748 Garching b. München, Germany
e-mail: uekerman@in.tum.de, web page: <http://www5.in.tum.de>

[†]Barcelona Supercomputing Center, NEXUS I building, Office 204
Gran Capitán 2-4, 08034 Barcelona, Spain
e-mail: {juan.cajas,guillaume.houzeaux,mariano.vazquez}@bsc.es,
web page: <http://www.bsc.es/computer-applications>

**Institut für Informatik , Technische Universität München
Boltzmannstraße 3, 85748 Garching b. München, Germany
e-mail: gatzhamm@in.tum.de, web page: <http://www5.in.tum.de>

[‡]Institute for Parallel and Distributed Systems, Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
e-mail: miriam.mehl@ipvs.uni-stuttgart.de,
web page: <http://www.ipvs.uni-stuttgart.de/abteilungen/sgs/>

Key words: Fluid-Structure Interaction, Partitioned Simulation, Black-Box Coupling, High Performance Computing, Many-Core Systems

Abstract. Partitioned coupling approaches between a black-box fluid and a black-box structure solver allow a maximum flexibility to choose the right solver for a particular application. This flexibility guarantees a decent time-to-solution when developing simulations for complex multi-physics applications.

In this work, we document the coupling of the highly parallel **Alya** System by means of the coupling library **preCICE**. This step allows the coupling of **Alya** to any other solver including commercial tools. **preCICE**, furthermore, benefits from this development, since the parallel nature of the **Alya** System constitutes a perfect test case for the upcoming parallelization of **preCICE**, marking a crucial step towards partitioned fluid-structure interaction on massively parallel systems.

1 INTRODUCTION

Multi-physics applications are of increasing importance, since modern supercomputing facilities make a new range of simulations feasible. On the one hand, such applications possess a natural need to run efficiently on massively parallel system, since, otherwise, the more detailed mathematical models would be useless. The simulation of blood flow in the human cardiovascular system to study the risk of calcification and aneurysms, for example, can only profit from a fluid-structure interaction (FSI) simulation, compared to a single fluid simulation, if boundary layers of the flow are resolved carefully. This can lead to drastic costs. To run simulation tools on massively parallel systems, on the other hand, poses new challenges to the multi-physics community, since many approaches cannot easily be ported to such systems. We believe that only a strong flexibility in the development process, allows to cope with these challenges, while keeping a realistic time-to-solution. Therefore, we favor a partitioned coupling approach, which allows to reuse highly developed single-physics codes that already have proven to run efficiently on modern supercomputing architectures.

The **Alya** system, developed at the Barcelona Supercomputing Center (BSC)) has proven such efficiency ([1]). The library **preCICE** ([2]), on the other hand, developed at the group for Scientific Computing in Computer Science (SCCS), part of the Technische Universität München (TUM), enables coupling of different single-physics codes, providing functionalities for coupling algorithms, mesh mapping, and communication means. In this work, we document the coupling of the **Alya** module for structure simulation, **SOLIDZ**, with the module for incompressible flow, **NASTIN**, through **preCICE**. The complete set-up of the coupling and the test cases as well as the simulation and physical validation was done by 2 part-time programmers during 5 weeks showing the potential of **preCICE** in terms of flexibility and simplicity.

The remainder of this paper is organized as follows. Section 2 and Section 3 give a brief introduction into the coupling library **preCICE**, and the **Alya** system, respectively. Section 4 describes the set-up of the coupling, and Section 5 shows numerical results validating the correctness of our coupling approach. The conclusion in Section 6 draws future lines in the development of **preCICE** and the collaboration between the BSC and SCCS.

2 A BRIEF INTRODUCTION TO PRECICE

preCICE (precise code interaction coupling environment) ([2]) is developed to allow for partitioned multi-physics simulations using black-box solvers. The main goal is to achieve an as flexible as possible coupling environment allowing for solver exchange in a nearly plug-and-play manner. Therefore, **preCICE** provides a high-level library programming interface which enables to transparently employ equation coupling schemes, data mapping methods for non-matching meshes, and communication means for distributed solver

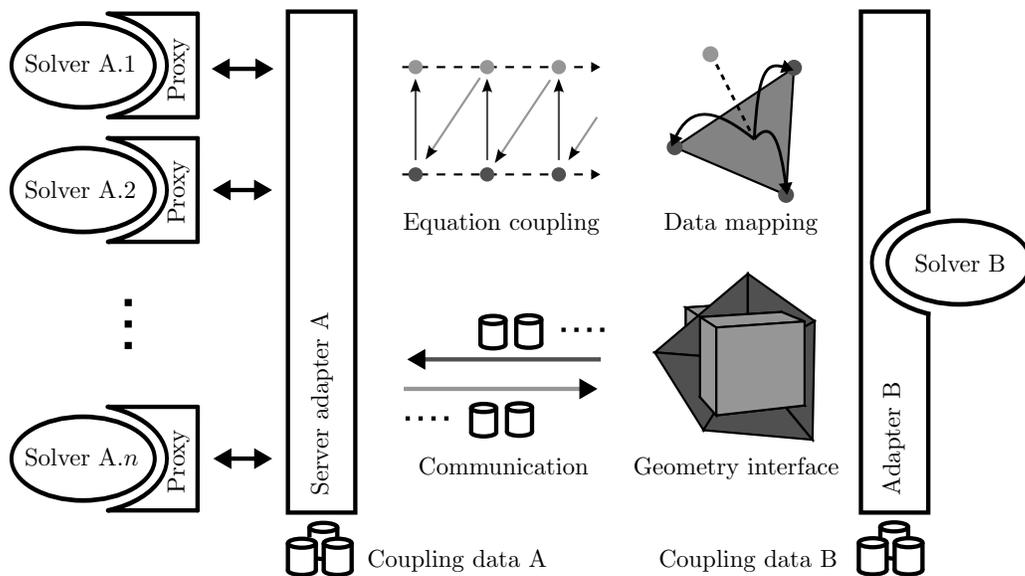


Figure 1: Main functionality and coupling principles of `preCICE`. In the middle, the four main groups of functions are shown, namely equation coupling schemes, data mapping methods, communication between distributed solvers, and surface geometry query functionality (cf. [3]). `preCICE` can be used for parallel solvers (such as solver A on the left) or sequential solvers (such as solver B on the right).

executables. Solvers, once adapted to `preCICE`, can be exchanged without changing the solver codes or the solver adapter, respectively. Commercial codes benefit from the black-box compatible functionality in `preCICE`, which works with minimal information from the solvers. Figure 2 gives an overview of the functionality groups and the component deployment.

To cope with instabilities caused by the added-mass effect, which are well-known in FSI applications with a comparable light or heavily deformable structure ([4, 5]), sophisticated equation **coupling schemes** are necessary. The supported classical and advanced schemes in `preCICE` are categorized into explicit or implicit and parallel or serial schemes. As explicit schemes, the conventional serial staggered (CSS) procedure and the conventional parallel staggered procedure (CPS) [6] have been implemented. The supported serial implicit schemes encompass a simple Gauß-Seidel solver, dynamic Aitken under-relaxation ([7]), and the IQN-ILS solver ([8]). Furthermore, the parallel implicit coupling scheme V-IQN, developed in [9, 10], is available.

In case of non-matching meshes, **data mapping methods** need to be applied. Three types of schemes have been implemented in `preCICE`: a nearest neighbor mapping [11], nearest projection mapping [12, 13], and a radial basis function mapping [11, 14].

Additionally, in order to use distributed solver executables, `preCICE` provides **communication** means based on files, sockets, and MPI, where the MPI implementation allows for a common or separate start-up of the solver executables.

To couple the solvers, a peer-to-peer communication layout is used reducing the com-

```

1  precice::SolverInterface precice("SolverName", solverRank, solverThreadSize);
2  precice.configure("configuration.xml");
3  int dataID = precice.getDataID("DataName");
4  int[] dataIndices = precice.setMeshVertices(meshID, dataSize, coordinates);
5  setup solver data structures
6  double preciceMaxDt = precice.initialize()
7  while (precice.isOngoing()){
8      dt = min(preciceMaxDt, solverDt);
9      compute solver time step using dt
10     precice.writeBlockVectorData(dataID, dataSize, dataIndices, data);
11     preciceMaxDt = precice.advance(dt);
12     precice.readBlockVectorData(dataID, dataSize, dataIndices, data);
13 }
14 precice.finalize();
15 tear down solver data structures

```

Figure 2: Example for adapting a solver to `preCICE`. All coupling numerics and the data exchange happens within the method `advance` in line 11. The original single-physics code is marked in grey.

plexity of the execution logic needed. In the alternative approach with a central server, two (mirrored) instead of one logic of communication are needed: one for the solvers and one for the server. When using parallel solvers, `preCICE` currently uses a client-server approach, where the solver processes are clients and `preCICE` data belonging to the solver processes are located in a separate process acting as a server. Note that, if both solvers run in parallel, each of them has an own server. At the moment, the peer-to-peer communication between the two solvers is serialized and becomes a bottleneck for very large amounts of data. In upcoming work, we will focus on the parallelization of this bottleneck. This work constitutes a major step towards this development as the highly parallel nature of `Alya` allows for a perfectly suited test case.

To adapt a solver to `preCICE`, the `preCICE` application programming interface (API) has to be integrated into the solver code (or its programming interface). Existing solvers have a predefined structure and changes to it can be undesirable or even impossible in case of commercial solvers. Providing the `preCICE` functionality in form of a library (instead of a framework) allows to keep the structure of a solver since API methods can be inserted at appropriate places in the solver code. Algorithm 2 shows the logical structuring of a solver and the main `preCICE` C++ API methods integrated into it (corresponding C and FORTRAN APIs exist as well). `preCICE` is configured from an XML file in line 2, defining the coupling data, interface meshes, coupled solvers and coupling functionality used in the simulation. Vectorial data are written and read to/from `preCICE` in lines 10 and 12 en block from the solver's C-array `data`. All coupling numerics and the data exchange happens within the method `advance` in line 11. `preCICE` can also prescribe the time step size of a simulation by giving an upper limit for the next time step to the solver (lines 6

and 10). It is an upper limit, since a solver can always perform a sub-cycle, i.e., a smaller time step.

At the moment `preCICE` adapters have been written for various solver including the commercial tools `Fluent` and `Comsol`, the open source solvers `OpenFOAM` and `Calculix`, and many in-house solvers (cf. [2, 10]).

3 A BRIEF INTRODUCTION TO ALYA

The `Alya` System, developed at the Barcelona Supercomputing Center (BSC), is a computational mechanics code with two main features. First, it is specially designed to run with the highest efficiency standards in large-scale supercomputing facilities. Second, it is capable to solve different physics problems, each one with its own modeling characteristics. These two main features are intimately related, which means that any complex, coupled problem solved by `Alya` will be solved efficiently. This work, now, allows `Alya` to also couple to other solver enabling a new range of applications, and to furthermore benefit from the coupling and data mapping methods that `preCICE` is offering.

`Alya` is organized using a modular architecture organized in kernel, modules and services. The kernel contains the facilities required to solve any set of discretized partial differential equations (e.g., the solver, the I/O, the elements database, the geometrical information, etc.), while the modules provide the physical description of a given problem. There are modules for handling incompressible and compressible flows, non-linear solids mechanics, species transport equations, excitable media, thermal flows, n-body collisions, electro-magnetism, quantum mechanics, and Lagrangian particle transport. In the present contribution, the modules to solve the incompressible Navier-Stokes equations, called `NASTIN` combined with the mesh moving module `ALEFOR`, and the module for solid mechanics problems, called `SOLIDZ` are used. Currently, `NASTIN` and `SOLIDZ` are part of the benchmark suite of the Partnership for Advanced Computing in Europe (PRACE)¹.

Full details of the parallelization of `Alya` can be found in [1]. Briefly, the parallelization is based on a master-slave strategy for distributed memory supercomputers, using MPI as the message passing library. The master reads the mesh and performs the partition of the mesh into subdomains using `METIS` (an automatic graph partitioner)². Each process will then be in charge of a subdomain. These subdomains are the slaves. The slaves build the local element matrices and the local right-hand sides, and are in charge of solving the resulting systems in parallel.

In the assembling tasks, no communication is needed between the slaves, and the scalability depends only on the load balancing. In the iterative solvers the scalability depends on the size of the interfaces and on the communication scheduling. During

¹<http://www.prace-ri.eu/>

²<http://glaros.dtc.umn.edu/gkhome/views/metis>

the execution of the iterative solvers, two main types of communications are required: Global communications via `MPI_Allreduce`, which are used to compute residual norms and scalar products, and point to point communications via `MPI_Sendrcv`, which are used when sparse matrix vector products are calculated.

In `NASTIN`, the numerical model is a stabilized finite element method. The stabilization is based on the Variational Multi-Scale method (VMS). The formulation is obtained by splitting the unknowns into grid scale and subgrid scale components. This method was introduced in 1995 [15] and established a remarkable mathematical basis for understanding and developing stabilization methods. In the present formulation of Alya, the subgrid scale is tracked in time and space, making the numerical model more accurate and stable [16]. The discretization of the Navier-Stokes equations yields a coupled algebraic system to be solved at each linearization step within a time loop. Algebraic solvers for this coupled system are not robust enough, therefore the system is split to solve the momentum and continuity equations independently. This is achieved by applying an iterative strategy, named the Orthomin(1) method for the Schur complement of the pressure [17]. At each linearization step it is necessary to solve the momentum equation twice and the continuity equation once. The momentum equation is solved using the GMRES or BICGSTAB method (diagonal and Gauß-Seidel preconditioners are usually efficient), and the continuity equation is solved using the Deflated Conjugate Gradient method [18] together with a linelet preconditioner well-suited for boundary layers.

In `SOLIDZ`, the equation of balance of momentum, casted in a total Lagrangian formulation, is solved using a standard Galerkin method for a large deformation framework, and a generalized Newmark time integration scheme [19], combined with the iterative Newton-Raphson algorithm. Well known constitutive equations for small and large deformation elasticity models are available, and very complex solid mechanics problems can be addressed [20].

To solve the fluid structure interaction problem, the mesh of the fluid is moved accordingly with the deformation of the solid body. The Alya module which is in charge of this movement is named `ALEFOR`, and calculates the mesh displacement and velocity which are used in an Algebraic Lagrangian Eulerian formulation (ALE) included in `NASTIN`. The movement of the mesh is governed by a Laplace equation and pseudo physical properties are used to preserve the quality of the mesh near the deforming boundary.

4 COUPLING ALYA WITH PRECICE

For the coupling of the Alya modules `NASTIN/ALEFOR` and `SOLIDZ` it turns out, that checkpointing of the unknowns plays a crucial role. By checkpointing, we understand the treatment of the solver variables after a coupling iteration that did not lead to convergence. A bad choice may hinder the convergence of the coupling algorithms, or may even change the physical results, as, for example, forces on the structure may be accumulated

otherwise. Table 4 gives an overview over the **Alya** variables used for a fluid-structure interaction simulation, the associated **Alya** module, and the checkpointing strategy.

Table 1: Overview over the **Alya** variables used for a fluid-structure interaction simulation, the associated **Alya** module, and the checkpointing strategy after a coupling iteration

module	variable	explanation	strategy
NASTIN	veloc	fluid velocity	keep
	press	fluid pressure	keep
	coord	fluid mesh coordinates	converged \Rightarrow update
ALEFOR	dispm	mesh displacement	keep
	velom	mesh velocity	keep
	coord_ale	local ALE mesh coordinates	keep
SOLIDZ	displ	structure displacement	not converged \Rightarrow reload last ts.
	veloc_sld	structure velocity	not converged \Rightarrow reload last ts.
	accel_sld	structure acceleration	not converged \Rightarrow reload last ts.

5 NUMERICAL RESULTS

To validate the success of our coupling, we simulate the FSI benchmark proposed in [21]. Subsection 5.1 gives information about the scenario set-up whereas Subsection 5.2 presents the results that we achieve, compared to the experimental data.

5.1 Scenario Description

We simulate the 2D incompressible laminar flow around a fixed cylinder with an attached elastic Saint-Venant Kirchhof cantilever placed in the middle of the flow channel with a small vertical offset. The geometry is depicted in Figure 3. At the left boundary, a parabolic inflow profile is prescribed while, at the right boundary, we have a free outflow condition. The top and the lower wall of the channel as well as the surface of the cylinder and the cantilever are no-slip boundaries. In [21], three different FSI scenarios are proposed, which differ only in the material and flow parameters. We restrict the results for this paper to the FSI3 benchmark with the highest Reynolds number and the lowest density of the structure marking the most challenging setting for FSI coupling algorithms ($\rho_S/\rho_F = 1$, $Ae = 1.4 \cdot 10^{-3}$, $Re = 200$, for details see [21]). The transient movement of the cantilever converges to a periodic oscillation. Once converged, the movement of the cantilever can be validated using reference values for mean values, amplitudes, and frequencies of the forces exerted on the structure as well as the movement of point A on the cantilever (compare Figure 3). This benchmark scenario constitutes a severe test case for FSI simulations as it comprises large displacements and a strong coupling between fluid and structure.

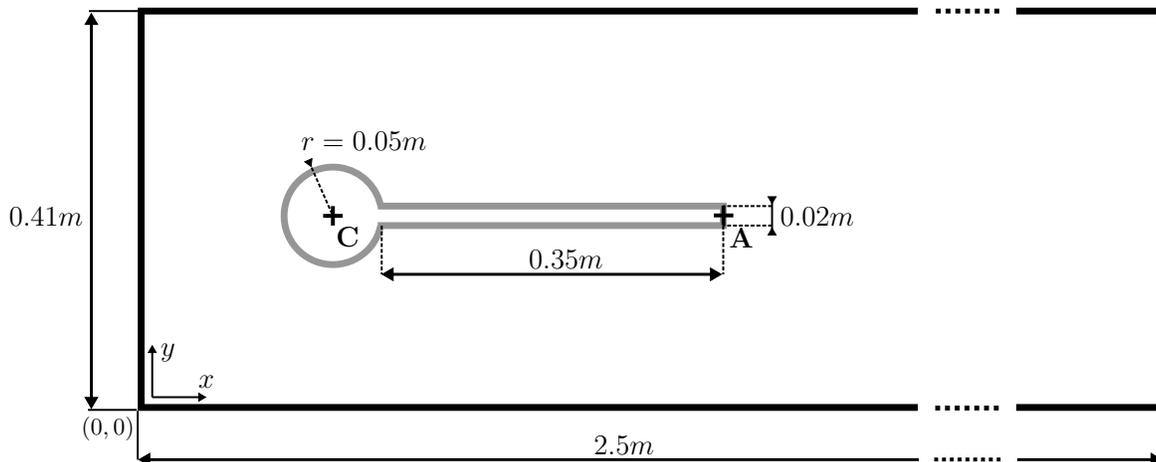


Figure 3: Sketch of the FSI benchmark scenario proposed in [21]. The geometrical layout consists of a channel with a fixed cylinder and an attached elastic beam. The slightly off-centric position of the cylinder’s center (Point $C = (0.2m, 0.2m)$) fosters oscillations. The measurements of displacements at the moving point $A = (0.6m, 0.2m)$, which is at the center of the backside of the cantilever, allows for the validation of numerical results.

5.2 Results

The simulation was carried out on the SNB partition of the MAC Cluster³ with a moderate MPI parallelization (32 processors for each `Alya` executable `NASTIN`, and `SOLIDZ`). Figure 4 shows the velocity magnitude, the fluid solver’s mesh, and the position of the structure at $t = 2.22s$. The fluid and structure mesh consist of 35524 and 8295 nodes, respectively. The time step size is set to 10^{-3} . We use the V-IQN coupling algorithm (compare [9, 10]), based on the difference between the actual displacement and the one from the last time step. The associated interface least-square system uses up to 50 columns, based on up to 5 previous time steps. At the beginning of each time step, the coupling variables (i.e., displacement differences and forces) are extrapolated using a second order scheme. A relative convergence criterion for displacements and forces leads to an average coupling iteration number of 3.82 per timestep.

Figure 5 shows the position of point A, while Table 2 lists the relevant reference values, compared to achieved results. We observe close agreement for the displacement values, validating our coupling approach.

³http://www.mac.tum.de/wiki/index.php/MAC_Cluster

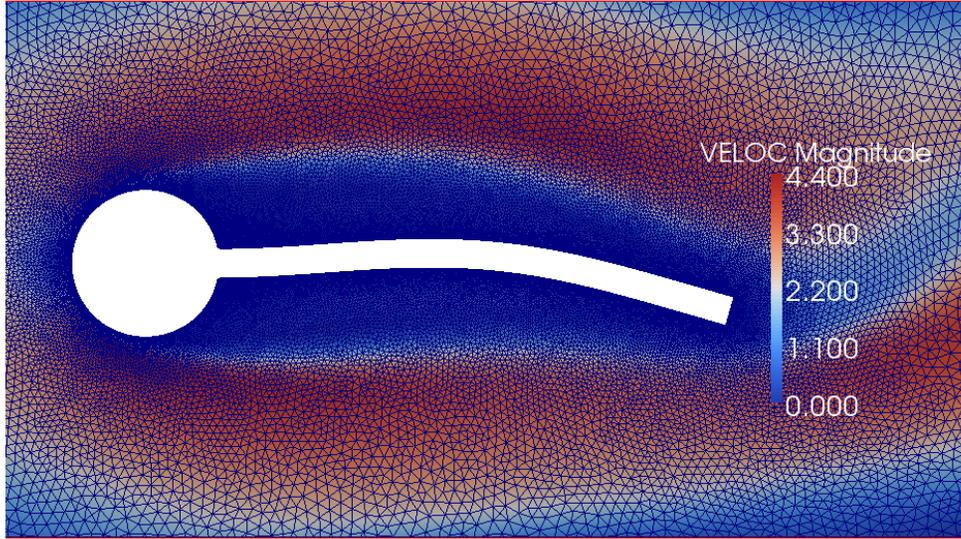


Figure 4: Velocity magnitude and grid of the fluid solver for the FSI3 benchmark scenario at $t = 2.22s$.

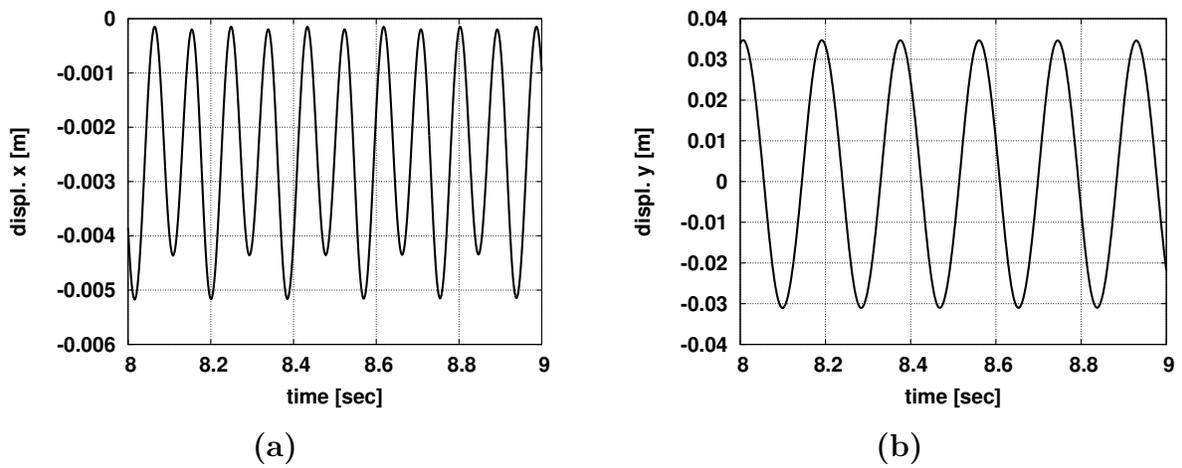


Figure 5: Displacement values for the FSI3 benchmark scenario: (a) Displacement of point A in x -direction; (b) displacement of point A in y -direction.

Table 2: Displacement values at point A for the FSI3 benchmark scenario, given as mean \pm amplitude [frequency]. We calculated these values by averaging over the first 4 oscillation after $t = 8s$.

	displ. x [$10^{-3}m$]	displ. y [$10^{-3}m$]
Simulation	-2.59 ± 2.42 [10.8]	2.75 ± 33.32 [5.4]
Reference	-2.69 ± 2.53 [10.9]	1.48 ± 34.38 [5.3]

6 CONCLUSIONS

We coupled the fluid and structure solver of the highly efficient **Alya** System by means of the coupling library **preCICE** and validated our approach with a well-established benchmark scenario. Now, both **Alya** solvers can be coupled to other solvers including commercial tools. Together with these new possibilities, **Alya** constitutes an environment for flexible multi-physics simulations covering a broad field of applications. In upcoming work, we want to use these possibilities to simulate real-world 3D applications.

The serialized server processes of **preCICE** constitute the biggest bottleneck at the moment, limiting the solvers to an only moderate parallelization. However, with the coupling of the highly scalable **Alya** system, we have a perfect test case for the upcoming parallelization strategies that we want to develop for **preCICE**.

Acknowledgement The financial support of the Institute for Advanced Study (IAS) of the Technische Universität München, of the Consejo Nacional de Ciencia y Tecnología (CONACyT, México), grant number 231588_290790, and of SPPEXA, the German Science Foundation Priority Programme 1648 – Software for Exascale Computing, are thankfully acknowledged.

REFERENCES

- [1] Houzeaux G., Vázquez M., Aubry R. and Cela J. A massively parallel fractional step solver for incompressible flows, *J. of Comput. Phys.* (2009) **228**:6316–6332.
- [2] Gatzhammer, B. Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions. *PhD Thesis*. Technische Universität München, Institut für Informatik (2014).
- [3] Bungartz, H.-J., Benk, J., Gatzhammer, B., Mehl, M. and Neckel, T.: Partitioned Simulation of Fluid-Structure Interaction on Cartesian Grids. In Bungartz, H.-J., Mehl, M., Schäfer, M., editors. *Fluid-Structure Interaction – Modelling, Simulation, Optimisation, Part II. Lecture notes in CSE*. Berlin, Springer (2010) **73**:255–284.
- [4] Van Brummelen, E. H. Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction. *J. Appl. Mech.* (2009) **76**.

- [5] Causin, P., Gerbeau, J. F. and Nobile, F. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comput. Methods Appl. Mech. Eng.* (2005) **194**:4506–4527.
- [6] Farhat, C. and Lesoinne, M. Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Comput. Method. Appl. M.* (2000) **182**:499–515.
- [7] Küttler, U. and Wall, W. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Comput. Mech.* (2008) **43**:61–72.
- [8] Degroote, J., Bathe, K.-J. and Vierendeels, J. Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction *Comput. Struct.* (2009) **87**:793–801.
- [9] Uekermann, B., Bungartz H. -J., Gatzhammer, B. and Mehl, M. A parallel, black-box coupling algorithm for fluid-structure interaction. *Proceedings of 5th International Conference on Computational Methods for Coupled Problems in Science and Engineering* (2013).
- [10] Mehl, M., Uekermann, B., Bijl, H., Blom, D., Gatzhammer, B. and Van Zuijlen, A. Parallel Coupling Numerics for Partitioned Fluid-Structure Interaction Simulations. *submitted to SIAM SISC* (2013).
- [11] De Boer, A., Van Zuijlen, A. H. and Bijl, H. Comparison of Conservative and Consistent approaches for the Coupling of Non-Matching Meshes. *Comp. Meth. in Appl. Mech. and Eng.* (2008) **197**:4284–4297.
- [12] Ahrem, R., Post, P., Steckel, B. and Wolf, K. MpCCI: A Tool for Coupling CFD with Other Disciplines. *Proceedings of the 5th World Conference in Applied Fluid Dynamics, CFD - Efficiency and the Economic Benefit in Manufacturing* (2001).
- [13] Brenk, M. Algorithmic Aspects of Fluid-Structure Interactions on Cartesian Grids. *PhD Thesis*. Technische Universität München, Institut für Informatik (2007).
- [14] Beckert, A. and Wendland, H. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerospace Science and Technology* (2001) **5**:125–134.
- [15] Hughes, T.J.R. Multiscale Phenomena: Green’s functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, (1995), **127**:387–401.
- [16] Houzeaux G. and Principe J. A variational subgrid scale model for transient incompressible flows. *Int. J. Comp. Fluid Dyn.*, (2008), **22**:135–152.

- [17] Houzeaux G., Aubry R. and Cela J. Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement. *Computers & Fluids* (2011), **44**:297–313.
- [18] Lohner R., Mut F., Cebal J., Aubry R., Houzeaux G. Deflated preconditioned conjugate gradient solver for the pressure-Poisson equation: extensions and improvements. *Int. J. Num. Meth. Engng.* (2010), **87**:2-14
- [19] Belytschko T., Liu W. K., Moran B., *Nonlinear Finite Elements for Continua and Structures*, Wiley, (2000).
- [20] Lafortune P., Aris R., Vázquez M., Houzeaux G. Coupled electromechanical model of the heart: Parallel finite element formulation. *Int. J. Num. Meth. Bio. Engng.*, (2012), **28**:72–86.
- [21] Turek S. and Hron, J. Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow. In Bungartz, H. -J. and Schäfer, M., editors. *Fluid-Structure Interaction Lecture notes in CSE*. Berlin, Springer (2006) **53**:371–385.