Technical Report

TR-2016-008

The LifeV library: engineering mathematics beyond the proof of concept

by

Luca Bertagna, Simone Deparis, Davide Forti, Luca Formaggia, Alessandro Veneziani

Mathematics and Computer Science

EMORY UNIVERSITY

The LifeV library: engineering mathematics beyond the proof of concept

Luca Bertagna, Department of Mathematics and Computer Science, Emory University, Atlanta (GA) 30322 USA
Simone Deparis, CMCS–MATHICSE–SB, École Polytechnique Fédérale de Lausanne, Station 8, Lausanne, CH–1015, Switzerland (simone.deparis@epfl.ch)
Luca Formaggia, MOX, Dipartimento di Matematica, Politecnico di Milano, Italy (luca.formaggia@polimi.it)
Davide Forti, CMCS–MATHICSE–SB, École Polytechnique Fédérale de Lausanne, Station 8, Lausanne, CH–1015, Switzerland (davide.forti@epfl.ch)
Alessandro Veneziani, Department of Mathematics and Computer Science, Emory University, Atlanta (GA) 30322 USA (avenez2@emory.edu) .

LifeV is a library for the finite element (FE) solution of partial differential equations in two and three dimensions. It is written in object oriented C++ and designed to run on diverse parallel architectures, including cloud facilities. In spite of its academic research nature to be a code for the development and testing of new methods, one distinguishing feature of the library is its use on real world problems and it is intended to provide a tool for many engineering applications. It has been actually used in computational hemodynamics, including cardiac mechanics and fluid-structure interaction problems, in porous media, ice sheets dynamics for both forward and inverse problems. In this paper we give a short overview of the features of LifeV and its coding paradigms on simple problems. The main focus is on the parallel environment which is mainly driven by domain decomposition methods and based on external libraries such as MPI, the Trilinos project, HDF5 and ParMetis.
Dedicated to the memory of Fausto Saleri.

CCS Concepts:•**Mathematics of computing** → **Solvers;** Mathematical software performance;•**Software and its engineering** → **Software libraries and repositories;**

ACM Reference Format:
Luca Bertagna, Simone Deparis, Luca Formaggia, Davide Forti, Alessandro Veneziani, 2016. The LifeV library: engineering mathematics beyond the proof of concept *ACM Trans. Model. Comput. Simul.* 0, 0, Article 0 (2016), 29 pages.
DOI: 0000001.0000001

1. INTRODUCTION

LifeV[1] is a parallel library written in C++ for the approximation of Partial Differential Equations (PDEs) by the finite element method in one, two and three dimensions. The project started in 1999/2000 as a collaboration between the modeling and scientific computing group (CMCS) at EPFL Lausanne and the MOX laboratory at Politecnico di Milano. Later the REO and ESTIME groups at INRIA joined the project. In 2006 the

[1]Pronounced "life five", the name stands for Library for Finite Elements, 5th edition as V is the Roman notation for the number 5

This work is supported by several public agencies, cf. Section 1.1.
Corresponding Author: Simone Deparis

DOI: 0000001.0000001

library has been progressively parallelized using MPI with the Trilinos library suite as back-end interface. In 2008 the Scientific Computing group at Emory University joined the LifeV consortium. Since then, the number of active developers has fluctuated between 20 and 40 people, mainly PhD students and researchers from the laboratories within the LifeV consortium.

LifeV is open source and currently distributed under the LGPL license on github[2], and migration to BSD License is currently under consideration. The developers page is hosted by a Redmine system at https://cmcsforge.epfl.ch/projects/lifev, while divulgative pages are at http://www.lifev.org. LifeV has two specific aims: (i) it provides tools for developing and testing novel numerical methods for single and multi-physics problems, and (ii) it provides a platform for simulations of engineering and, more generally, real world problems. In addition to "basic" Finite Elements tools, LifeV also provides data structures and algorithms tailored for specific applications in a variety of fields, including fluid and structure dynamics, heat transfer, and transport in porous media, to mention a few. It has already been used in medical and industrial contexts, particularly for cardiovascular simulations, including fluid mechanics, geometrical multiscale modeling of the vascular system, cardiac electro-mechanics and its coupling with the blood flow.

LifeV has benefited from the contribution of many PhD students, almost all of them working on project financed by public funds. We provide a list of supporting agencies hereafter.

When in 2006 we decided to introduce parallelism, the choice has turned towards available open-source tools: MPI (mpich or openmpi implementations), ParMETIS, and the Epetra, AztecOO, IFPACK, ML, Belos, and Zoltan packages distributed within Trilinos [Heroux et al. 2005].

In this review article we explain the parallel design of the library and provide two examples of how to solve PDEs using LifeV. Section 2 is devoted to a description of how parallelism is handled in the library while in Section 3 we discuss the distinguishing features and coding paradigms of the library. In Section 4 we illustrate how to use LifeV to approximate PDEs by the finite element method, using a simple Poisson problem as an example. In Section 5 we show how to approximate unsteady Navier–Stokes equations and provide convergence, scalability, and timings. We conclude by pointing to some applications of LifeV.

1.1. Financial support

LifeV has been supported by the 6th European Framework Programme (Haemodel Project, 2000-2005, PI: A. Quarteroni), the 7th European Framework Programme (VPH2 Project, 2008-2011, ERC Advanced Grant MATHCARD 2009-2014, PI: A. Quarteroni), the Italian MIUR (PRIN2007, PRIN2009 and PRIN12 projects, PI: A. Quarteroni and L. Formaggia), the Swiss National Science Foundation (several projects, from 1999 to 2017, 59230, 61862, 65110, 100637, 117587, 125444, 122136, 122166, 141034, PI: A. Quarteroni), the Swiss supercomputing initiatives HP2C and PASC (PI: A. Quarteroni), the Fondazione Politecnico di Milano with Siemens Italy (Aneurisk Project, 2005-2008, PI: A. Veneziani), the Brain Aneurysm Foundation (Excellence in Brain Aneurysm Research, 2010, PI: A. Veneziani), Emory University Research Committee Projects (2007, 2010, 2015, PI: A. Veneziani), ABBOTT Resorb Project (2014-2017, PI: H. Samady, D. Giddens, A. Veneziani), the National Science Foundation (NSF DMS 1419060, 2014-2017, PI: A. Veneziani, NSF DMS 1412963, 2014-2017, PI: A. Veneziani, NSF DMS 1620406, 2016-2018, PI: A. Veneziani), iCardioCloud Project (2013-2016, PI: F. Auricchio, A. Reali, A. Veneziani), the Lead beneficiary programm

[2]https://github.com/lifev

(Swiss SNF and German DFG, 140184, 2012-2015, PI: A. Klawonn, A. Quarteroni, J. Schröder), and the French National Research Agency. Some private companies also collaborated to both the support and the development, in particular MOXOFF SpA (2012-2104) exploited the geometric multiscale paradigm for the simulation of an industrial packaging system and and Eni SpA (2011-2014) contributed to the development of the Darcy solver and extended FE capabilities, as well as the use of the library for the simulation the evolution of sedimentary basins [Cervone et al. 2012].

1.2. Main Contributors

The initial core of developers was the group of A. Quarteroni at MOX, Politecnico di Milano, Italy and at the Department of Mathematics, EPFL, Lausanne, Switzerland from an initiative of L. Formaggia, J.F. Gerbeau, F. Saleri and A. Veneziani. The group of J.F. Gerbau at the INRIA, Rocquencourt, France gave significant contributions from 2000 through 2009 (in particular with M. Fernandez and, later, M. Kern). The important contribution of C. Prud'homme and G. Fourestey during their stays at EPFL and of D. Di Pietro at University of Bergamo are acknowledged too. S. Deparis has been the coordinator of the LifeV consortium since 2007.

Here, we limit to summarize the list of main contributors who actively developed the library in the last five years. We group the names by affiliation. As some of the authors moved over the years to different institutions, they may be listed with multiple affiliations hereafter.

[D. Baroli, A. Cervone, M. Del Pra, N. Fadel, E. Faggiano, L. Formaggia, A. Fumagalli, G. Iori, R.M. Lancellotti, A. Melani, S. Palamara, S. Pezzuto, S. Zonca][3]. [L. Barbarotta, C. Colciago, P. Crosetto, S. Deparis, D. Forti, G. Fourestey, G. Grandperrin, T. Lassila, C. Malossi, R. Popescu, S. Quinodoz, S. Rossi, R. Ruiz Baier, P. Tricerri][4], M. Kern[5], [S. Guzzetti, L. Mirabella, T. Passerini, A. Veneziani] [3,6], U. Villa [6], L. Bertagna [67], M. Perego [3,6,7,8], A. Quaini[9 6], H. Yang [6,7], A. Lefieux[10 6].

Abbreviations. : FE (Finite Elements), FEM (Finite Element Method), DoF (Degree(s) of Freedom), OO (Object Oriented), PCG (Preconditioned Conjugate Gradient), PGMRes (Preconditioned Generalized Minimal Residual), ML (Multi Level), DD (Domain Decomposition), MPI (Message Passing Interface), ET (Expression Template), HDF (Hierarchical Data Format), HPC (High Perfromance Computing), CSR (Compact Sparse Row)

2. THE PARALLEL FRAMEWORK

The library can be used for the approximation of PDEs in one dimension, two dimensions, and three dimensions. Although it can be used in serial mode (i.e., with one processor), parallelism is crucial when solving three dimensional problems. To better underline the ability of LifeV to tackle large problems, in this review we focus on PDEs discretized on unstructured linear tetrahedral meshes, although we point out that LifeV also supports hexahedral meshes as well as quadratic meshes.

Parallelism in LifeV is achieved by domain decomposition (DD) strategies, although it is not mandatory to use DD preconditioners for the solution of sparse linear systems.

---

[3]MOX, Politecnico di Milano IT
[4]CMCS, EPFL Lausanne, CH
[5]INRIA, Rocquencourt, FR
[6]Dept. Math & CS, Emory Univ, Atlanta GA USA
[7] Department of Scientific Computing, Florida State University, Tallahassee, FL USA
[8]Sandia National Lab, Albuquerque, NM USA
[9]Department of Mathematics, University of Houston, TX USA
[10]Department of Civil Engineering and Structures, University of Pavia, IT,

In a typical simulation, the main steps involved in the parallel solution of the finite element problem using LifeV are the following:

(1) All the MPI processes load the same (not partitioned) mesh.
(2) The mesh is partitioned in parallel using ParMETIS or Zoltan. At the end each process keeps only its own local partition.
(3) The degrees of freedom (DoFs) are distributed according to the mesh partitions. By looping on the local partition, a list of local DoF in global numbering is built.
(4) The FE matrices and vectors are distributed according to the DoF. In particular, the matrices are stored in row format, for which whole rows are assigned to the process owning the associated DoF.
(5) Each process assembles its local contribution to the matrices and vectors. Successively, global communication consolidates contributions on shared nodes (at the interface of two subdomains).
(6) The linear system is solved using an iterative solver, typically either a Preconditioned Conjugate Gradient (PCG) when possible or a Preconditioned GMRes (PGMRes). The preconditioner runs in parallel. Ideally, the number of preconditioned iterations should be independent of the number of processes used.
(7) The solution is downloaded to mass storage in parallel using HDF5 for post-processing purposes (see Sect. 2.4).

The aforementioned steps are explained in detail in the next subsections.

2.1. Mesh partitioning: ParMETIS and Zoltan

As mentioned above, LifeV achieves parallelism by partitioning the mesh among the available processes. Typically, this is done "online": the entire mesh is loaded by all the processes but it is deleted after the partitioning phase, so that each process keeps only the part required for the solution of the local problem and to define inter-process communications. As the mesh size increases, the "online" procedure may become problematic. Therefore for large meshes it is possible, and sometimes necessary, to partition the mesh offline [Popescu 2013] . This may require a significant amount of memory on a local workstation. Furthermore, it is also possible to include an halo of ghost elements such that the partitions overlap by one or more layers of elements with each other (see e.g. [Guzzetti et al. 2015]). This may be relevant for schemes that require a large stencil.

In addition to the online vs offline choice, the user can also choose the package to use to perform the partitioning. In particular, LifeV can interface with two third party libraries: ParMetis and Zoltan.

"ParMETIS is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. ParMETIS extends the functionality provided by METIS and includes routines that are especially suited for parallel AMR computations and large scale numerical simulations"[11] [Karypis and Kumar 1998].

Zoltan is a package distributed within Trilinos, which offers a variety of capabilities, including graph partitioning and coloring, dynamic load balancing, and distributed directory, to mention a few. For more details, see [Devine et al. 2002].

The following code snippet shows an example of structured mesh generation and subsequent online partitioning, performed using ParMETIS. For simplicity, the example refers to a structured mesh that is however handled as unstructured, as this is the most typical case in the applications.

[11] Quoted from http://glaros.dtc.umn.edu/gkhome/views/metis

```
// Mesh
typedef RegionMesh< LinearTetra > mesh_Type;
boost::shared_ptr< mesh_Type > fullMeshPtr ( new mesh_Type ( Comm ) ); // Comm is the
                                                                       // MPI communicator

// For simplicity, generate structured mesh. However, in general,
// LifeV deals with unstructured meshes
// Building structured mesh, in this case a cube [-1,1]^3, which is handled as unstructured
regularMesh3D ( *fullMeshPtr, 0,
        dataFile( "mesh/nx", 15 ), dataFile( "mesh/ny", 15 ),
        dataFile( "mesh/nz", 15 ), dataFile ( "mesh/verbose", false ),
        2.0, 2.0, 2.0, -1.0,-1.0, -1.0 );

// Partitioning mesh, possibly with overlap
Teuchos::ParameterList params;
params.set<std::string>("graph-lib","parmetis");
params.set<int>("overlap",0);
MeshPartitionTool< mesh_Type > meshCutter(fullMeshPtr,Comm,param);
if (! meshCutter.success() )
{
  std::cout << "Partitioning failed." << std::endl;
  return EXIT_FAILURE;
}

boost::shared_ptr< mesh_Type > localMeshPtr = meshCutter.meshPart();
// Clearing global mesh to save memory
fullMeshPtr.reset();
```

2.2. Distributed arrays: Epetra

The sparse matrix class used in LifeV is a wrapper to the Epetra matrix container Epetra_FECrsMatrix and, similarly, the vector type is a wrapper to Epetra_FEVector, both provided by the Epetra [Heroux 2009] package of Trilinos. The distribution of the unknowns is determined automatically by the partitioned mesh: with a loop over each element of the local mesh we create the list of DoF managed by the current processor. This procedure in fact creates a *repeated* map, i.e., an instance of an Epetra_Map with some entries referring to the DoF associated with geometric entities lying on the interface between two (or more) subdomains. Then, a *unique* map is created, in such a way that, among all the owners of a *repeated* DoF, only one will also own it in the *unique* map. The unique map is used for the vectors and matrices to be used in the linear algebra routines as well as for the solution vector. The repeated map is used to access information stored on other processors, which is usually necessary only in the assembly and post-processing phases.

Epetra implements import and export tools that take care of the communications to redistribute the information among vectors with different maps. For instance, when solving the unsteady Navier–Stokes equations using a semi-implicit time-advance scheme, the solution vector for the velocity (which has a *unique* map) at the previous time iteration is used as the convective vector field in the transport term, and a *repeated* copy has to be created in order to perform the assembly correctly. The handling of inter-processor communication is delegated to the Epetra_FECrsMatrix and Epetra_FEVector classes, however the LifeV wrapper to Epetra_FECrsMatrix provides a method for explicit and transparent use of this kind of communication.

The assembly of the FE matrices is typically performed by looping on the local elements [Ern and Guermond 2006; Formaggia et al. 2012]. To reduce latency time, the loops on each subdomain are performed in parallel, without need of any communication during the loop. Once all processes have assembled their local contribution, a single communication phase takes place to consolidate contributions on interface dofs.

Efficiency and stability may be improved by two further avilable operations, (i) precomputing the matrix graph; (ii) using overlapping meshes. The former demands for

an additional step at the beginning of the simulation to compute the matrix pattern. This is an Epetra_Graph, associated to the matrix. Since it depends on the problem at hand and the chosen finite elements, the computation of the pattern needs a loop on all the elements. This is coded by *expression templates* (see Sect. 3.2) using the same call sequence as for the matrix assembly. The latter further reduces communications by allowing all processes to compute the local finite element matrix on all elements sharing a DoF on the interface. As a result, each process can independently compute all the entries of matrices and vectors pertaining to the DoF it owns. Although the processes which do not own the interface degrees of freedom will eventually discard those extra entries, the little overhead is justified by the reduction (in fact, complete elimination) of the post-assembly communication costs.

2.3. Parallel preconditioners

The solution of linear systems in LifeV relies on the Trilinos [Heroux et al. 2005] packages AztecOO and Belos [Bavier et al. 2012], which provide an extensive choice of iterative or direct solvers. LifeV provides a common interface to both of them.

The proper use of ParMETIS and Zoltan for the partitioning, and of Epetra matrices and vectors for the linear algebra, ensures that matrix-vector multiplication and vector operations are properly parallelized, i.e., they scale well with the number of processes used, and communications are optimized. In this situation the parallel scalability of iterative solvers like PCG or PGMRES depends essentially on the properties of the preconditioner.

The choice of preconditioner is thus critical. In our experience it may follow two directions: (i) parallel preconditioners for the generic linear systems , like single or multilevel overlapping Schwarz preconditioners, or multigrid preconditioners, that are generally well suited for highly coercive elliptic problems, or incomplete factorization (ILU), which are generally well suited for advection-dominated elliptic problems; (ii) problem specific preconditioners, typically required for multifield or multiphysics problem;

these preconditioners exploits specific features of the problem at hand to recast the solution to standard problems that can be eventually solved with the generic strategies in (i), like, e.g., the SIMPLE, the Least Square Commutator, the Caouet-Chabard and the Yosida preconditioners for the incompressible Navier-Stokes equations [Patankar and Spalding 1972; Elman et al. 2014; Cahouet and Chabard 1988; Veneziani 2003] or the Monodomain preconditioner for the Bidomain problem in electrocardiology [Gerardo-Giorda et al. 2009].

In LifeV preconditioners for elliptic problems are indeed an interface to the Trilinos package IFPACK [Sala and Heroux 2005], which is a suite of algebraic preconditioners based on incomplete factorization, and to ML [Gee et al. 2006] or MueLu [Prokopenko et al. 2014], which are two Trilinos packages for multi-level preconditioning based on algebraic multigrid. Typically we use IFPACK to define algebraic overlapping Schwarz preconditioners with exact or inexact LU factorization of the restricted matrix. The preconditioner P in this case can be formally written as

$$P^{-1} = \sum_{i=1}^{n} R_i^T A_i^{-1} R_i, \text{ where } A_i = R_i A^{-1} R_i^T, \tag{1}$$

where A is the finite element matrix related to the PDE approximation, n is the number of partitions (or subdomains) Ω_i, R_i is the restriction operator to Ω_i and R_i^T the extension operator from Ω_i to the whole domain Ω. A_i is inverted many times during the iterations of PCG or PGMRES, which is why it is factorised by LU or ILU. In LifeV, the choice of the factorization is left to the user.

The geometric Schwarz preconditioner in (1) needs some clarification. The operations are intended to be performed at the matrix level, so R_i is the matrix restricting a global vector to the DoF belonging to Ω_i and the overlap is defined by the graph of A. When considering the discretization of a Laplacian problem by piecewise linear (P1) finite elements the geometric and algebraic approaches are equivalent. For other types of problems/elements, they may differ at the interface between subdomains. For more details, see [Sala and Heroux 2005]

Similarly, it is possible to use multilevel preconditioners via the ML [Gee et al. 2006] or MueLu [Prokopenko et al. 2014; Hu et al. 2014] packages. They work at the algebraic level too and the coarsening and extension are done either automatically, or by user defined strategies, based on the parallel distribution of the matrix and its graph. The current distribution of LifeV does not offer the last option, but the interested developer could add this extra functionality with relatively little effort. We point out that the overlapping Schwarz preconditioners provided by IFPACK and described above can also be used as smoothers for multigrid methods in ML and MueLu.

2.4. Parallel I/O with HDF5

When dealing with large meshes and large number of processes, input/output access to files on disk deserves particular care. A prerequisite is that the filesystem of the supercomputing architecture being used provides the necessary access speed in parallel. However this is not enough. MPI itself offers parallel I/O capabilities, for which HDF5 is one of the existing front-ends [The HDF Group 1997]. LifeV uses HDF5 for I/O processing essentially for three reasons.

— The number of files produced is independent of the number of running processes: each process accesses the same file in parallel and writes out his own chunks of data; moreover it is possible to organize the binary file as a directory tree where the different variables may be stored. As a result, LifeV generally produces one single large output binary file, along with a xmf text file describing its contents.
— HDF5 is compatible with open source post-processing visualization tools like Paraview [Ahrens et al. 2005] and VisIt [Childs et al. 2012]). When dealing with very large meshes, it is also possible to run it in parallel and/or remotely. These features and its user friendly interface make it very attractive.
— Having one single binary file makes it very easy to use for restarting a simulation. In particular, no particular care is needed when using a different number of processes with respect to the first simulation, since all the data is stored independently of the underlying number of processes used.

The interface to HDF5 in LifeV exploits the facilities of the EpetraExt package in Trilinos, and since the LifeV vectors are compatible with Epetra format the calls are simple [Heroux 2009]. Since the meshes are unstructured and the numbering of the DoF is independent of the partitions (on the original mesh), the indices of the DoF in a subdomain are not necessarily continuous. However, the writing phase by HDF5 is very efficient as it is based on hyperslices which are used by EpetraExt to redistribute all the data between the processes before writing. Although this may represent a parallel bottleneck, in our experience it is negligible.

3. FEATURES AND PARADIGMS

3.1. I/O data formats

In a typical simulation, the user will provide a text file containing input data (including physical and discretization parameters, and options for the linear/nonlinear solvers), and the code will generate results, which need to be stored for post-processing. Al-

though it is up to the user to write the program main file where the data file is parsed, LifeV makes use of two particular classes in order to forward the problem data to all the objects involved in the simulation: the GetPot class [Schaefer 2007] (for which LifeV also provides an *ad hoc* re-implementation inside the core module), and the ParameterList class from the Teuchos package in Trilinos. The former has been the preferred way since the early development of LifeV, and is therefore supported by virtually all classes that require a setup. The latter is used mostly for the linear and nonlinear solvers, since it is the standard way to pass configuration parameters to the Trilinos solvers, but is also supported by several other LifeV structures. Both classes map strings representing the names of properties to their actual values (be it a number, a string, or other), and they both allow the user to organize the data in a tree structure. Details one the syntax of the two formats are available online. Here, we just present a one-line example to illustrate how the structures can be queried. In the scenario where the data structure has to be queried for the value of the property "viscosity", which is located in the *subtree* "fluid", and which should default to the value `1.0` when no specific value is provided, the syntax would be

```
double mu = data_g("fluid/viscosity",1.0);
```
or
```
double mu = data_pl.sublist("fluid").get<double>("viscosity",1.0);
```

where `data_g` and `data_pl` are two objects of type GetPot and ParameterList, respectively.

When it comes to mesh handling, LifeV has the built in capability of generating structured meshes on domains of the form $\Omega = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$. If a more general mesh is required, the user needs to create it beforehand, store it in a file, and pass the information about the mesh file via data file to LifeV. Currently, LifeV supports the FreeFem [Hecht 2012] and Gmsh [Geuzaine and Remacle 2009] formats (both usually with extension `.msh`) in 2D, while in 3D it supports the formats of Gmsh, Netgen [Schöberl 1997] (usually with extension `.vol`)

and Medit [Frey 2001] (usually with extension `.mesh`). Additionally, as mentioned in Section 2.1, LifeV offers the capability of offline partitioning, where the partitions of a mesh are stored and subsequently loaded using HDF5.

Finally, LifeV offers three different formats for storing the simulation results for post-processing: Ensight [Computational Engineering International, Inc. 2011], HDF5 [The HDF Group 1997], and VTK [Schöberl et al. 2006]. All these formats are supported by the most common scientific visualization software packages, including Paraview [Ahrens et al. 2005] and VisIt [Childs et al. 2012]. The details on these formats can be found on their respective webpages, and the differences in terms of performance are beyond the scope of this work. However, we point out that, from the user point of view, the three formats behave in a different manner when the simulation is run with multiple processor and/or with multiple time steps. In particular, as already mentioned in the previous section, HDF5 generates only two files, regardless of the number of processors and time steps. On the other hand, VTK and Ensight generate one different file for each processor per time step. This may have an impact on storage and execution time, depending on the specifics (particularly concurrent I/O capabilities) of the filesystem where the results are saved. Si, mi sembra un dettaglio che si può omettere, però il tool aggiungiamolo alla libreria!

3.2. Expression Templates for Finite Elements

One of the aims of LifeV is to be used in multiple contexts, ranging from industrial and social applications to teaching purposes. For this reason, it is important to find the best trade-off between computational efficiency and readability of the code. The high

level of abstraction proper of C++ is in principle perfectly matched by the abstraction of mathematics. Concepts like differential operators - where the same mathematical framework can be used for different problems and applications - can be expressed using the generic programming feature of C++. However, versatility and efficiency may conflict. Quoting [Furnish 1997], a "natural union has historically been more of a stormy relationship". This aspect is crucial in High Performance Computing, where efficiency is a priority. Operator overloading has a major impact on efficiency, readability, maintainability and versatility, however it may adversely affect the run time. *Expression Templates* (ET) have been originally devised to minimize this drawback by T. Veldhuizen [Veldhuizen 1995], and later further developed in the context of linear algebra [Härdtlein et al. 2009; Iglberger et al. 2012] and solution of partial differential equations [Pflaum 2001; Di Pietro and Veneziani 2009a].

We recall here the basic idea. Let us assume to have a vector operation to code, i.e. a scalar operation to be repeatedly performed on different components of vectors. This operation is called the *scalar core*. The ET technique allows to express the vector operations in a generic and abstract way, without complex and error prone loops, while maintaining the low memory footprint and speed of component-wise operations. In fact, this relies on two pillars,

(1) collect the logical structure of the mathematical expression to be performed in a special object (the *Expression*);
(2) expand the operation at compile time only during the final assignment of the values, producing efficient code.

These two tasks can be achieved by four steps:

- defining the operand types;
- creating wrapping class of *Expression* type to homogenize heterogeneous operands;
- overloading operators, or creating functions, that accepts *Expression*s and return a more complex expression where the desired operation has been added;
- creating spacial assignment and constructors that "evaluate" at compile time the expression, producing efficient code.

Notice that the assembly of a finite element matrix can be naturally arranged as a sequence of vector operations, where the scalar core is the evaluation of a combination of polynomial basis functions and their derivatives in a specific quadrature node. Basis functions and their derivatives are the operands where diverse algebraic operations are to be performed. The specific combination may involve constants, functions or vectors, depending on the specific differential operator to solve.

For instance, for the classical linear advection-diffusion-reaction equation in the unknown u, $-\mu\Delta u + \beta \cdot \nabla u + \sigma u$, we need to combine three different differential operators weighted by coefficients μ, β and σ. These may be numbers, prescribed functions or pointwise functions (inherited for instance by another FE computation). In nonlinear problems — after proper linearization — expressions may involve finite element functions too. Breaking down the assembly part to each differential operator individually with its own coefficient is possible but leads to duplicating the loops over the quadrature nodes, as opposed to the assembly of their sum at the level of the scalar core. In [Di Pietro and Veneziani 2009b] a first introduction to ET in finite element modeling is given. Templetized definition of all the possible differential operators in the construction of linear and nonlinear advection-diffusion-reaction problems is given. Numerical tests show that for non-constant coefficients and nonlinear problems — for both continuous and discontinuous finite elements — ET provide readable code with no efficiency loss for operator parsing. As a matter of fact, the final gathering of all the assembly operations in one loop as opposed to standard approaches with a separate assembly of

elemental operators (diffusion, reaction, advection) introduces computational advantages.

The ET paradigm has been therefore used in the development of LifeV since 2009. A detailed description of the paradigm definition and its implementation in LifeV can be found in [Quinodoz 2012] and in the code snapshots presented later on.

3.3. Integrated Pipelines: from images to simulations

LifeV is oriented to address practical problems, beyond the proof of concept of specific numerical methods for PDEs. In particular, one of the most important applications — yet non exclusive — is the simulation of cardiovascular problems. Examples of applications to real clinical problems are simulations of Left Ventricular Assist Devices (LVAD) [Bonnemain et al. 2013; Bonnemain et al. 2012], the study of the physiological [Crosetto et al. 2011b] and abnormal fluid-dynamics in ascending aorta in presence of a bicuspid aortic valve [Bonomi et al. 2015; Vergara et al. 2012; Faggiano et al. 2013; Viscardi et al. 2010], of Thoracic EndoVascular Repair (TEVAR) [van Bogerijen et al. 2014; Auricchio et al. 2014], of the Total Cavopulmonary Connection [Mirabella et al. 2013; Restrepo et al. 2015; Tang et al. 2015], of blood flow in stented coronary arteries [Gogas et al. 2013] and in cerebral aneurysms [Passerini et al. 2012].

As such, it is crucial to integrate this tool into a systematic pipeline devised to be used in a professional routine, clinical or more generally industrial. In the specific case of cardiovascular applications, we are envisioning LifeV as part of a systematic use of computational results in what one of the authors introduced as the *Computer Aided Clinical Trial* (CACT) concept [Veneziani 2015]. In practice, clinical trials — currently based on systematic and repeated measurements on enrolled suitable patients — will be soon be improved by informations obtained by numerical simulations. One of the projects supporting the development of the library[12] has been developed with this perspective. To this aim, LifeV needs to be embedded in a routine pipeline covering the entire process from image/data retrieval to the post-processing in a totally patient-specific setting. Eventually, this environment will heavily rely on Cloud Computing infrastructures.

Currently, the pipeline is articulated as follows.

(1) Image/Data retrieval: this basically consists of storing images and data of patients on repositories/devices according to standard data formats. For image data, the most common format is the so-called DICOM (DIgital COMmunication). Ideally, this step can work directly from imaging devices to cloud infrastructures. However, deidentification and data protection procedures are necessary to guarantee compliance with privacy requirements.
(2) Image processing: this consists on the extraction of morphological and functional data to be used in numerical simulations. In particular, for vascular geometries, our tool of choice has been Vascular Modeling ToolKit (VMTK), a Python/C++ library for image segmentation and 3D reconstruction based on the level set method[Antiga et al. 2008a]. When possible, data are retrieved to prescribe boundary conditions too. At the end of this step, a surface triangulation of the region of interest is available. The most popular format is STL.
(3) For a finite element analysis we typically use unstructured tetrahedral meshes. However, LifeV has been developed also for hexahedral elements. Mesh generation is generally outsourced to available softwares like NetGen, TetGen [Page 2016c] or GMSH [Geuzaine and Remacle 2009]. In practical applications geometries retrieved from patients may feature non regular properties that prevent the meshing step. In

[12]ABSORB Project granted by Abbott Inc. at Emory University

this case, CAD tools are used to perform local regularization and surface remeshing operations. In established pipelines we use VMTK [Antiga et al. 2008b], NetFabb [Page 2016b] and MeshLab [Page 2016a] to regularize the surface for obtaining appropriate meshes. Meshing may be an iterative procedure per se, where after a first standard step, local refinement may be needed. This is the case of refinements required at the vascular walls when wall shear stress analysis need to be performed (sometimes called *mesh boundary layers*) or computational domains are added by inlet/outlet artificial regions (the so-called *flow extensions*). These steps can be done with NetGen (by an appropriate specification of the mesh size) and VMTK, respectively. At the end of this step, volumetric meshes with appropriate boundary labels are available.

(4) Finite Element analysis is performed by LifeV.
(5) Postprocessing is regularly performed by Paraview. As mentioned previously, this is an open source software compatible with VTK format for sophisticated visualizations and analysis of the results. LifeV has been generally developed to write results in different formats compatible with Paraview, EnSight and HDF5.

A complete automation of the pipeline is currently out of reach for general purposes. In fact, specific problems still require ad hoc procedure for image processing and meshing. Geometry regularization for meshing may be quite problematic in patient-specific settings and a semi-automatic pipeline is currently the only possible approach, even though more automation will be included, see e.g. [Yang et al. 2015].

4. BASICS: LIFE = LIBRARY OF FINITE ELEMENTS

The library supports different type of finite elements. The use of ET makes the set-up of simple problems easy, as we illustrate hereafter.

4.1. The Poisson problem

As a first example, we present the setup of a finite element solver for a Poisson problem. We assume that the domain Ω be polygonal and that its boundary $\partial\Omega$ be split into two subsets Γ^D and Γ^N of positive measure such that $\Gamma^D \cup \Gamma^N = \delta\Omega$ and $\Gamma^D \cap \Gamma^N = \emptyset$. Let $V_h^D \subset H^1_{\Gamma^D}$ be a discrete finite element space relative to a mesh of Ω, for example continuous piecewise linear functions vanishing on Γ_D. The Galerkin formulation of the problem reads: find $u_h \in V_h^D$ such that

$$\int_\Omega \kappa \boldsymbol{\nabla} u_h : \boldsymbol{\nabla}\varphi_h = -\int_\Omega \kappa \boldsymbol{\nabla} u^D : \boldsymbol{\nabla}\varphi_h + \int_\Omega f\varphi_h + \int_{\Gamma^N} g^N \varphi_h \qquad \forall \varphi_h \in V_h^D, \tag{2}$$

where κ is the diffusion coefficient, possibly dependent on the space coordinate, g^N is the Neumann boundary condition, $\frac{\partial u}{\partial n} = g^N$ on Γ^N, and f are the volumetric forces. The lifting u_D can be any finite element function such that $u^D|_{\Gamma^D}$ is a suitable approximation of g^D. As usually done, u^D is such that $u^D|_{\Gamma^D}$ is the Lagrange interpolation of g^D, extended to zero inside Ω.

In LifeV, DoF associated with Dirichlet boundary conditions are not physically eliminated from the FE unknown vectors and matrices. Even though this elimination would certainly affect positively the performances of the linear algebra solver, it introduces a practical burden in the implementation and memory particularly in 3D unstructured problems, that makes it less appealing. The enforcement of these conditions can be done alternatively in different ways as illustrated e.g. in [Formaggia et al. 2012], after the matrix assembly. We illustrate the strategy adopted in the following sections.

4.2. Matrix form and expression templates

Firstly, we introduce the matrix assembled for homogeneous Natural conditions associated with the differential operator at hand (aka "do-nothing" boundary conditions, as they do not require any extra work to the pure discretization of the differential operator), i.e.

$$A = (a_{ij})_{i,j=1,...,n} \text{ and } a_{ij} = \int_\Omega \kappa \boldsymbol{\nabla} \varphi_j : \boldsymbol{\nabla} \varphi_i, i, j = 1, ..., n,$$

where φ_j, $j = 1, ..., n$ are the basis functions of the finite element space V_h. Thanks to expression templates [Di Pietro and Veneziani 2009b; Quinodoz 2012] in LifeV we can express the assembly of matrix A as

```
// Matrices and graphs. We use shared pointer
typedef Epetra_FECrsGraph                   graph_Type;
typedef boost::shared_ptr<graph_Type>       graphPtr_Type;
typedef MatrixEpetra< Real >                matrix_Type;
typedef boost::shared_ptr< matrix_Type >    matrixPtr_Type;

graphPtr_Type systemGraph ( new graph_Type ( Copy, * ( uFESpace->map().map( Unique ) ), 50,
     useOverlap ) );

// Reading the diffusion coefficient kappa from datafile
double kappa = datafile("coefficients/kappa", 1.0);

using namespace ExpressionAssembly;

buildGraph (
             elements ( localMeshPtr ),
             quadrature,
             uFESpace,
             uFESpace,
             dot ( grad ( phi_i ) , grad ( phi_j ) )
            )
              >> systemGraph;

systemGraph->GlobalAssemble();

matrixPtr_Type systemMatrix( new matrix_Type ( uFESpace->map(), *systemGraph, useOverlap ) );

integrate (
            elements ( localMeshPtr ),
            uFESpace->qr(),
            ETuFESpace,
            ETuFESpace,
            value(kappa)*dot ( grad ( phi_i ) , grad ( phi_j ) )
           )
             >> systemMatrix;
```

In a similar way, we define the right hand side of the linear problem as the vector

$$\mathbf{b} = (b_i)_{i=1,...,n} \text{ where } b_i = \int_\Omega f \varphi_i + \int_{\Gamma^N} g^N \varphi_i, i = 1, ..., n.$$

In this case, possible non-homogeneous Neumann or natural conditions are included. Finally, the DoF related to the Dirichlet boundary conditions are enforced by setting the associated rows of A equal to zero except for the diagonal entries. In this way, the equation associated with the i-th Dirichlet DoF is replaced by

$$cu_i = cg_i$$

where c is a scaling factor, depending in general on the mesh size, to be used to control the condition number of the matrix. A general strategy is to pick up values of the same order of magnitude of the entries of the row of the do-nothing matrix being modified.

Without further modification, the system matrix is not symmetric anymore. Many of the problem faced in application are not symmetric, therefore we describe here only how to deal with non-symmetric matrices.

It is worth noting that the symmetry break does not prevent using specific methods for symmetric systems like CG when appropriate as pointed out in [Ern and Guermond 2006]. A symmetrization of the matrix can be also achieved by enforcing the condition $cu_i = cg_i$ column-wise, i.e. by setting to 0 also the off-diagonal entries of the columns of Dirichlet DoF. Some sparse-matrix formats oriented to row-wise access of the matrix, like the popular CSR, need in this case to be equipped with specific storage information that could make the column-wise access convenient [Formaggia et al. 2012].

4.3. Linear algebra

The linear system $A\mathbf{x} = \mathbf{b}$ can be solved by a preconditioned iterative method like PCG, PGMres, BiCGStab, etc, available in the packages AztecOO or Belos of Trilinos.

```
LinearSolver linearSolver ( Comm ); // initialise the linear solver with a MPI communicator
linearSolver.setOperator ( systemMatrix ); // set the matrix of the system

Teuchos::RCP< Teuchos::ParameterList > aztecList = Teuchos::rcp ( new Teuchos::ParameterList )
    ;
aztecList = Teuchos::getParametersFromXmlFile ( "SolverParamList.xml" );

linearSolver.setParameters ( *aztecList );

// +-----------------------------------------------+
// |           Setup the preconditioner            |
// +-----------------------------------------------+

typedef PreconditionerIfpack                                    prec_Type;
typedef boost::shared_ptr<prec_Type>                            precPtr_Type;

basePtr_Type precPtr;
precPtr.reset( new prec_Type() );
precPtr->setDataFromGetPot ( dataFile, "prec" );

linearSolver.setPreconditioner ( precPtr );

// +-----------------------------------------------+
// |              Solving problem                  |
// +-----------------------------------------------+

linearSolver.setRightHandSide( rhs ); // rhs is the right hand side Vector, including the
    Dirichlet data
linearSolver.solve( solution ); // solution is the solution on all the degrees of freedom
```

The choice of the method and its settings are to be set via an input file, here `SolverParamList.xml`

```
<ParameterList>
    <!-- LinearSolver parameters -->
    <Parameter name="Reuse Preconditioner" type="bool" value="false"/>
    <Parameter name="Max Iterations For Reuse" type="int" value="80"/>
    <Parameter name="Quit On Failure" type="bool" value="false"/>
    <Parameter name="Silent" type="bool" value="false"/>
    <Parameter name="Solver Type" type="string" value="AztecOO"/>

    <!-- Operator specific parameters (AztecOO) -->
    <ParameterList name="Solver: Operator List">

        <!-- Trilinos parameters -->
        <ParameterList name="Trilinos: AztecOO List">
            <Parameter name="solver" type="string" value="gmres"/>
            <Parameter name="conv" type="string" value="rhs"/>
```

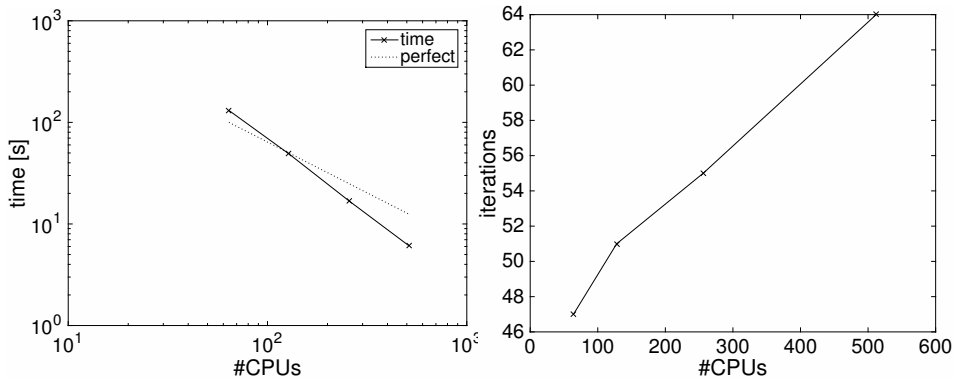

Fig. 1. Solving a Poisson problem in a cube with P2 finite elements with 1'367'631 degrees of freedom. The scalability in terms of CPU time (left) is perfect, however the number of iterations (right) linearly increases. The choice of the preconditioner is not optimal, the use of a coarse level or of multigrid in the preconditioner is essential and allows to use more processes with no loss of resources, cf. also Figures 2 and 3.

```
            <Parameter name="scaling" type="string" value="none"/>
            <Parameter name="output" type="string" value="all"/>
            <Parameter name="tol" type="double" value="1.e-12"/>
            <Parameter name="max_iter" type="int" value="200"/>
            <Parameter name="kspace" type="int" value="100"/>
            <!-- az_aztec_defs.h -->
            <!-- #define AZ_classic 0 /* Does double classic */ -->
            <Parameter name="orthog" type="int" value="0"/>
            <!-- az_aztec_defs.h -->
            <!-- #define AZ_resid 0 -->
            <Parameter name="aux_vec" type="int" value="0"/>
        </ParameterList>
    </ParameterList>
</ParameterList>
```

The main difficulty is to set up a scalable preconditioner. As pointed out , in LifeV there are several options based on algebraic additive Schwarz (AAS) or Multigrid preconditioners. In the first case, the local problem related to A_i in (1) has to be solved. It is possible to use an LU factorization, using the interface with Amesos [Sala et al. 2006b; 2006a] or incomplete factorizations (ILU). LU factorizations are more robust than incomplete ones in the sense that they do not need any parameter tuning, which is delicate in particular within AAS, while incomplete ones are much faster and require less storage. In a parallel context though, for a given problem, the size of the local problem is inversely proportional to the number of subdomains. The LU factorization, whose cost depends only on the number of unknowns, is perfectly scalable, but more memory demanding. An example of the scalability in this settings is given in Figure 1. LifeV leaves the choice to the user depending on the type of problem and computer architecture at hand.

The previous example is just an immediate demonstration of LifeV coding. For other examples we refer the reader to [Formaggia et al. 2012].

5. THE CFD PORTFOLIO

A core application developed since the beginning, consistently with the tradition of the group where the library has been originally conceived, is incompressible fluid dynamics, which is particularly relevant for hemodynamics. It is well-known that the problem has a *saddle point* nature that stems from the incompressiblity constraint. From the mathematical stand point, this introduces specific challenges, for instance the choice of finite element spaces for velocity and pressure that should satisfy the so called *inf-sup*

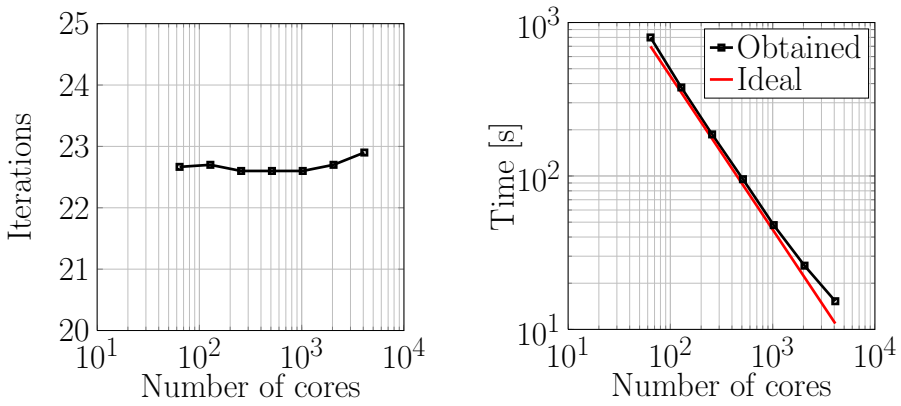

Fig. 2. Flow around a cylinder, Reynolds number equal to 22'000. Scalability of hybrid preconditioners for stabilized Navier–Stokes equations [Forti and Dedè 2015]. Simulations by Davide Forti on PizDora at CSCS. VMS-LES stabilized P2-P2 finite elements, 9'209'040 degrees of freedom and time step 0.0025 s. On the left the number of iterations and on the right the time to solve an entire time step.

condition, unless special stabilization techniques are used [Elman et al. 2014]. LifeV offers both possibilities. In fact, one can choose among inf-sup stable P2-P1 finite element pairs or equal order P1-P1 or P2-P2 stabilized formulations, either by interior penalty [Burman and Fernández 2007] or SUPG-VMS [Bazilevs et al. 2007; Forti and Dedè 2015].

As it is well known, for high Reynolds flow it is important to be able to describe turbulence by modeling it, being impossible to resolve it in practice. Being hemodynamics the main LifeV application, where turbulence is normally less relevant, LifeV has not implemented a full set of turbulence models. However, LifeV includes the possibility of using the Large Eddie Simulation (LES) approach. This relies on the introduction of a suitable filter of the convective field in the Navier-Stokes equations that has the role of bringing the effects of turbulent viscous dissipation to the mesh scale. The Van Cittert deconvolution operator considered in [Bowers and Rebholz 2012] has been recently introduced in LifeV in [Bertagna et al. 2015]. Validation up to a Reynolds number 6500 has been validated with the FDA Critical Path Initiative Test Case.

Another LES procedure based on the variational splitting of resolved and unresolved parts of the solution has been considered in [Forti and Dedè 2015]. Other LES filtering techniques have been implemente in [Lancellotti 2015].

5.1. Preconditioners for Stokes problem

In Section 2.3 we have introduced generic parallel preconditioners based on AAS or multigrid. These algorithms have been originally devised for elliptic problems. In our experience, their use for saddle-point problems like Darcy, Stokes and Navier–Stokes equations, or in fluid structure interaction problems, is not effective. However, their combination with specific preconditioners, like e.g. SIMPLE, Least Square Commutator, Yosida for unsteady Navier–Stokes equations or Dirichlet-Neumann for FSI, leads to efficient and scalable solvers.

In [Deparis et al. 2013] this approach has been applied with success to unsteady Navier-Stokes problems with inf-sup stable finite elements, and then extended to a VMS-LES stabilized formulation with equal order elements for velocity and pressure [Forti and Dedè 2015], see also Figure 2.

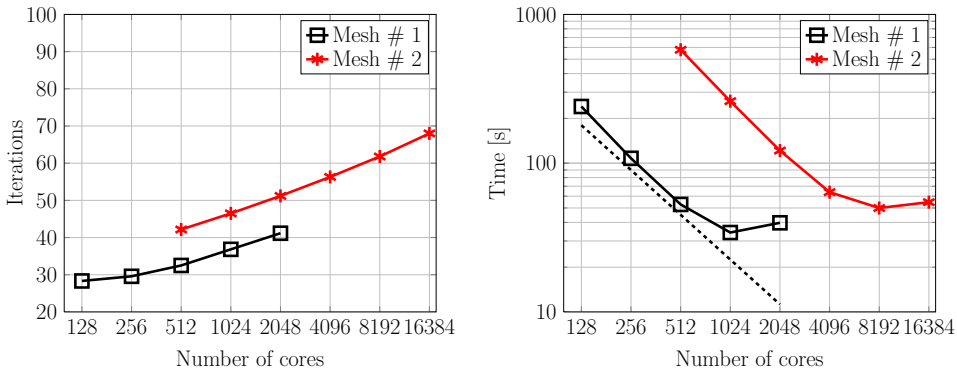

Fig. 3. Blood flow in a patient-specific arterial bypass. Scalability of FaCSI preconditioner for FSI simulations [Deparis et al. 2015]. Simulations by Davide Forti on PizDora at CSCS. Finite elements used: fluid P2-P1, structure P2, ALE mapping P2. See Table I for the number of degrees of freedom.

Table I. Femoropopliteal bypass test case: number of Degrees of Freedom (DoF).

Mesh	Fluid DoF	Structure DoF	Coupling DoF	Geometry DoF	Total
Coarse	9'029'128	2'376'720	338'295	8'674'950	20'419'093
Fine	71'480'725	9'481'350	1'352'020	68'711'934	151'026'029

Applying the same techniques to build a parallel preconditioner for FSI based on a Dirichlet–Neumann splitting [Crosetto et al. 2011a] and a SIMPLE [Patankar and Spalding 1972; Elman et al. 2008] preconditioner does not lead to a scalable algorithm. To this end, it is necessary to add additional algebraic operations which leads to a FaCSI preconditioner [Deparis et al. 2015], see also Figure 3.

5.2. Algebraic Factorizations

Another issue is to reduce the computational cost by separating pressure and velocity computations. The origin of splitting schemes for Navier-Stokes problems may be dated back to the separate work of A. Chorin and R. Temam that give rise to the well-known scheme that bears their name. The basic scheme operates at a differential level by exploiting the Helmholtz decomposition Theorem (also known as Ladhyzhenskaja Theorem) to separate the differential problem into the sequence of a vector advection-diffusion-reaction problem and a Poisson equation, with a final correction step for the velocity. As opposed to the "split-then-discretized" paradigm, B. Perot in [Perot 1993] advocates a "discretize then split" strategy, by pointing out the formal analogy between the Chorin-Temam scheme and an inexact LU factorization of the matrix obtained after discretization of the Navier-Stokes equations. This latter approach, often called "algebraic factorization" is easier to implement, particularly when one has to treat general boundary conditions [Quarteroni et al. 2000].

Let us introduce briefly a general framework. Let A be the matrix obtained by the finite element discretization of the (linearized) incompressible Navier-Stokes equations. The discretized problem at each step reads

$$A \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix} \qquad \text{with} \quad A = \begin{bmatrix} C & D^T \\ D & 0 \end{bmatrix}$$

where A collects the contribution of the linearized differential operator acting on the velocity field in the momentum equation, D and D^T are the discretization of the diver-

gence and the gradient operators, respectively. Notice that

$$A = LU = \begin{bmatrix} C & 0 \\ D & -DC^{-1}D^T \end{bmatrix} \begin{bmatrix} I & C^{-1}D^T \\ 0 & I \end{bmatrix}.$$

This "exact" LU factorization of the problem formally realizes a velocity-pressure splitting. However there is no computational advantage because of the presence of the matrix C^{-1}, which is not explicitly available, so any matrix-vector product with this matrix requires to solve a linear system. The basic idea of algebraic splitting is to approximate this factorization. A first possibility is to replace C^{-1} with

the inverse of the velocity mass matrix scaled by Δt. This is the result of the first term truncation of the Neumann expansion that may be used to represent C^{-1}. The advantage of this approximation is that the mass matrix can be further (and harmless) approximated by a diagonal matrix by the popular "mass-lumping" step. In this way, $DC^{-1}D^T$ is approximated by a s.p.d matrix — sometimes called "discrete Laplacian" for its spectral analogy with the Laplace operator — that can be tackled with many different convenient numerical strategies. In addition, it is possible to see that the splitting error gathers in the first block row, i.e. in the momentum equation. We finally note that replacing the original C^{-1} with the velocity mass matrix in the U factor of the splitting implies that the exact boundary conditions cannot be enforced exactly. The Yosida strategy, on the other hand, follows a similar pathway, except for not approximating C^{-1} in U. Similar properties can be proved as for the Perot scheme, however in this case the splitting error affects only the mass conservation (with a moderate mass loss depending on the time step) and the final step does actually enforce the exact boundary conditions for the velocity. Successively, different splittings have been proposed in [Gauthier et al. 2004; Saleri and Veneziani 2005; Gervasio et al. 2006; Gervasio and Saleri 2006; Gervasio 2008; Veneziani 2009] to reduce the impact of the splitting error by successive corrections of the pressure field.

LifeV incorporates these last developments. In particular, Yosida scheme has been preferred since the error on the mass conservation has less impact on the interface with the structure in fluid-structure interaction problems. It is worth noting that a special block operator structure reflecting the algebraic factorization concept has been implemented in [Villa 2011].

It is worth noting that a robust validation of these methods has been successfully performed not only against classical analytical test cases but also within the framework of one *Critical Path Initiative* promoted by the US Food and Drug Administration (FDA) [Passerini et al. 2013], (https://fdacfd.nci.nih.gov).

Time adaptivity. An interesting follow up of the pressure corrected Yosida algebraic factorizations is presented in [Veneziani and Villa 2012]. This work stems from the fact that the sequence of pressure corrections not only provides an enhancement of the overall splitting error, but also provides an error estimator in time for the pressure field - with no additional computational cost. Based on this idea, a sophisticated time adaptive solver has been introduced [Villa 2011; Veneziani and Villa 2012], with the aim of cutting the computational costs by a smart and automatic selection of the time step. The latter must be the trade-off among the desired accuracy, the computational efficiency and the numerical stability constraints introduced by the splitting itself. The final result is a solver that automatically detects the optimal time step, possibly performing an appropriate number of pressure correction to attain stability.

This approach is particularly advantageous for computational hemodynamics problems featuring a periodic alternation of fast and slow transients (the so called systolic and diastolic phases in circulation). As a matter of fact, for the same level of accuracy,

the total number of time steps required within a heart beat is reduced to one third of the ones required by the non adaptive scheme.

In fact, in [Veneziani and Villa 2012] a smart combination of algebraic factorizations as solvers and preconditioners of the Navier-Stokes equations based on the *a posteriori* error estimation provided by the pressure corrections is proposed as a potential optimal trade-off between numerical stability and efficiency.

6. BEYOND THE PROOF OF CONCEPT

As explained in Sect. 1, LifeV is intended to be a tool to work aggressively on real problems, aiming at a general scope of bringing most advanced methods for computational predictive tools in the engineering practice (in broad sense). For instance, a massive use of HPC in Computer Aided Clinical Trials (CACT) — as a way for extracting more knowledge from gold standard medical procedures for testing hypothesis, drugs or devices — is expected to bring a paradigm shift in clinical practice and decision-making in the years to come. In this respect, we may consider LifeV as a vehicle of *methodological transfer*, as leading edge methods are made available to the engineering community with a short time-to-market. In this section we present a series of distinctive applications where we feel that using LifeV actually allowed to to bring rapidly new methods to real problems beyond the proof of concept stage.

6.1. FSI

In the Fluid-Structure Interaction (FSI) context, LifeV has offered a very important bench for testing novel algorithms. For example, it has been possible to test Robin-based interface conditions for applications in hemodynamics [Nobile and Vergara 2008; Nobile et al. 2013; 2014], or compare segregated algoritms, the monolithic formulation, and the Steklov-Poincaré formulation [Deparis et al. 2006].

An efficient solution method for FSI problem considers the physical unknowns and the fluid geometry problem for the Arbitrary Lagrangian-Eulerian (ALE) mapping as a single variable. This monolithic description implies to use ad-hoc parallel preconditioners. Most often they rely on a Dirichlet-Neumann inexact factorization between fluid and structure and then specific preconditioners for the subproblems [Crosetto et al. 2011a]. Recently a new preconditioner FaCSI [Deparis et al. 2015] has been developed and tested with LifeV with an effective scalability up to 4 thousands processors. A next step in FSI has been the use of non-conforming meshes between fluid and structure using rescaled-localized radial basis functions [Deparis et al. 2014].

A study on different material constitutive models for cerebral arterial tissue — in particular Hypereslatic isotropic laws, Hyperelastic anisotropic laws — have been studied in [Tricerri et al. 2015]. A benchmark for the simulation of the flow inside carotids and the computation of shear stresses [Balzani et al. 2015] has been tested with LifeV coupled with the FEAP library [Taylor 2014], which includes sophisticated anisotropic material models.

6.2. Defective boundary conditions

One of the distinctive features of solving complex problems in life science and medicine in particular is that there is a significant difference between mathematical theory and practical aspects. In cardiovascular problems this refers particularly to the availability of data to be used as boundary conditions. As a matter of fact, geometry and boundary conditions are commonly considered the most important factors affecting the reliability of a simulation in computational hemodynamics. This topic has motivated an extensive piece of work to identify a mathematically sound way to complete the gap between the data needed by the theory of the equations to be solved as boundary conditions and the ones actually available from measures. The expression “defective boundary

conditions" has been introduced since the seminal work of Heywood, Rannacher and Turek [Heywood et al. 1996]. Several strategies have been introduced and analyzed in this respect to identify the *less invasive* way to fill the gap. With this, we mean the approach that reduces the impact of the arbitrariness of the closure conditions to the final solution. Among the others, we mention [Formaggia et al. 2002; Veneziani and Vergara 2005; 2007; Formaggia et al. 2008; 2010].

In particular, in hemodynamics we often have flow rate conditions like

$$\int_{\Gamma} \rho \boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n} d\gamma = Q(t).$$

A practical way to deal with this is to convert the average data into pointwise standard Dirichlet conditions by selecting a proper velocity profile. This procedure is quite standard. In LifeV we can prescribe flat, Poiseuille (parabolic) and Womersley profiles, being the Womersley function a well known analytical solution in unsteady problems with a periodic pressure drop in a cylindrical circular pipe. However, this approach significantly affects the entire numerical solution, as arbitrary Dirichlet conditions significantly affect the solution in the neighborhood of the associated boundary. For this reason, a common approach is to augment the geometrical domain by the so-called *flow extensions*. These take the arbitrary conditions far from the region of interest. However, this approach has the drawback of introducing extra — sometimes nontrivial — pre-processing, to introduce extra DoF that may burden the computations and to be largely based on an empirical rationale. Alternatives less impactful to the final results are commendable. Among the others, in LifeV the augmented Lagrange multiplier approach is available [Veneziani and Vergara 2007], which provides an excellent trade-off between sensitivity and computational costs. It is also worth mentioning its applications to Ultrasound velocity estimates [Ponzini et al. 2006].

6.3. Geometric Multiscale

The cardiovascular system features coupled local and global dynamics. Modeling its integrity by three dimensional geometries including FSI is either unfeasible or very expensive computation-wise and, most of the time, useless. A more efficient model entails for the coupling of multiple dimensions, like lumped zero-dimensional models, hyperbolic one-dimensional ones, and three dimensional FSI, leading to the so called *geometric multiscale* modeling as advocated in [Veneziani 1998; Formaggia et al. 2001]. We do not cover in detail this topic here, as it has been subject of several papers, see e.g. the recent review [Quarteroni et al. 2016].

LifeV implements a full set of tools to integrate 0D, 1D and 3D-FSI models of the cardiovascular system [Malossi et al. 2011; Malossi et al. 2013; Blanco et al. 2013; Passerini et al. 2009], with also a multirate time stepping scheme to improve the computational efficiency [Malossi et al. 2012]. It has been used to simulate integrated models of the cardiovascular system [Bonnemain et al. 2013].

An accurate treatment of the boundary conditions with strong mathematical roots is expected to play a major role in the introduction of HPC in CACT and the geometrical multiscale approach is sometimes the appropriate way for transferring data from measurment sites (accessible to the doctor) to the boundaries of a region/district of interest. We envision therefore a great relevance of these possibilities offered by the library in the years to come.

6.4. Heart dynamics

Electrocardiology is one of the problems - beyond CFD but still related to cardiovascular mathematics - where LifeV has cumulated extensive experience. An effective

preconditioner for the bidomain equations has been proposed and demonstrated in [Gerardo-Giorda et al. 2009]. The basic idea is to use the simplified extended monodomain model to precondition the solution of the more realtistic bidomain equations. Successively, the idea has been adapted to reduce the computational costs by mixing Monodomain and Bidomain equations in an adaptive procedure. A suitabe *a posteriori* estimator is used to decide when the Monodoimain equations are enough or the bidomain solution is needed [Mirabella et al. 2011; Gerardo-Giorda et al. 2011]. Ionic models solved in LifeV ranges from the classical Rogers McCulloch, Fenton Karma, Luo Rudy I and II [Clayton et al. 2011] to more involved ones [Dupraz et al. 2015]. Specific high order methods (extending the classical Rush Larsen one) have been proposed and implemented in the library [Perego and Veneziani 2009].

In addition, one research line has been oriented to the coupling of electrocardiology with cardiac mechanics. Hyperelasticity problems based on non-trivial mixed and primal formulations with applications in cardiac biomechanics have been studied in [Rossi et al. 2011; 2012]. LifeV-based simulations of fully coupled electromechanics (using modules for the abstract coupling of solvers) can be found in [Nobile et al. 2012; Ruiz-Baier et al. 2013; Rossi et al. 2014; Andreianov et al. 2015] for whole organ models, and in [Ruiz-Baier et al. 2014; Gizzi et al. 2015; Ruiz-Baier 2015] for single-cell problems. The coupling with ventricular fluid dynamics and arterial tree FSI description are possible through a multiscale framework [Quarteroni et al. 2015]. The coupling the Purkinje network, a network of high electrical conductivity myocardium fibers, has been implemented in [Vergara et al. 2016].

Another research line in this field successfully carried out with LifeV is the variational estimation of cardiac conductivities (the tensor coefficients that are needed by the Bidomain equations) from potential measures [Yang and Veneziani 2013].

6.5. Inverse problems and data assimilation

One of the most recent challenging topics in computational hemodynamics is the quantification of uncertainty and the improvement of the reliability in patient-specific settings. As a matter of fact, while the inclusion of patient specific geometries is now a well established procedure (as we recalled above), many other aspects of the patient-specific modeling still deserve attention. Parameters like viscosity, vessel wall rigidity, or cardiac conductivity are not routinely measured (or measurable) in the specific patient and however have generally a major impact on the numerical results. These concepts have been summarized in [Veneziani and Vergara 2013]. Variational procedures have been implemented in LifeV, where the assimilation with available data or the parameter estimation are obtained by minimizing a mismatch functional. In [D'Elia et al. 2012] this approach was introduced to incorporate into the numerical simulation of the incompressible Navier-Stokes equations sparse data available in the region of interest; in [Perego et al. 2011] the procedure was introduced for estimating the vascular rigidity by solving an inverse FSI problem, while a similar procedure in [Yang and Veneziani 2013] aims at the estimate of the cardiac conductivity.

6.6. Model reduction

One of the major challenges of modern scientific computing is the controlled reduction of the computational costs. In fact, practical use of HPC demands for extreme efficiency — even real time solutions. Improvement of computing architectures and cloud solutions that make relatively easy the access to HPC facilties is only a partial answer to this need [Guzzetti et al. 2016]. From the modeling and methodological side, we need also customized models that can realize the trade off between efficiency and accuracy. These may be found by a smart combination of available High Fidelity solutions, according to the offline/online paradigm; or by the inclusion of specific features

of a problem that may bring a significant advantage in comparison with general versatile but expensive methods. In LifeV these strategy have been both considered. For instance, in [Colciago et al. 2014], a model for blood flow dynamics in a fixed domain, obtained by transpiration condition and a membrane model for the structure, has been compared to a full three dimesional FSI simulation. The former model is described only in the lumen with the Navier–Stokes equations, the structure is taken into account by surface Laplace–Beltrami operator on the surface representing the fluid-strucutre interface. The reduced model allows for roughly one third of the computational time and, in situations where the displacement of the artery is pretty small, the dynamics, including e.g. wall shear stresses, are very close if not indistinguishable from a full three dimensional simulation

In [Bertagna and Veneziani 2014] a solution reduction procedure based on the Proper Orthogonal Decomposition (POD) was used to accelerate the variational estimate of the Young modulus of vascular tissues by solving an Inverse Fluid Structuire Interaction problem. POD wisely combines available offline High Fidelity solutions to obtain a rapid (online) parameter estimation. More challenging is the use of a similar approach for cardiac conductivities [Yang 2015], requiring nonstandard procedures.

A directional model reduction procedure called HiMod (*Hierarchical Model Reduction* - see [Perotto et al. 2010; Aletti et al. 2015; Perotto 2014; Perotto and Veneziani 2014; Blanco et al. 2015; Mansilla Alvarez et al. 2016]) to accelerate the computation of advection diffusion reaction problems as well as incompressible fluids in pipe-like domains (or generally domains with a clear dominant direction, like in arteries) has been implemented in LifeV [Aletti et al. 2015; Guzzetti 2014]. These modules will be released in the library soon.

6.7. Darcy equations and porous media

Single and multi-phase flow simulators in fractured porous media are of paramount importance in many fields like oil exploration and exploitation, CO_2 sequestration, nuclear waste disposal and geothermal reservoirs. A single-phase flow solver is implemented with the standard finite element spaces Raviart-Thomas for the Darcy velocity and piecewise constant for the pressure. A global pressure-total velocity formulation for the two-phase flow is developed and presented in [Fumagalli and Scotti 2011; Fumagalli 2012] where the equations are solved using an IMPES-like technique. To handle fractures and faults in an efficient and accurate way the extended finite element method is adopted to locally enrich the cut elements, see [Iori 2011; Del Pra et al. 2015], and a one-codimensional problem for the flow is considered for these objects, see [Ferroni et al. 2016].

6.8. Ice sheets

Another application that has used the LifeV library (at Sandia Nat Lab, Albuquerque, NM) is the simulation of ice sheet flow. Ice behaves like a highly viscous shear-thinning incompressible fluid and can be modeled by nonlinear Stokes equations. In order to reduce computational costs several simplifications has been made to the Stokes model, exploiting the fact that ice sheets are very shallow. LifeV has been used to implement some of these models, including the Blatter-Pattyn (also known as First Order) approximation and the L1L2 approximation. The former model is a three-dimensional nonlinear elliptic PDE and the latter a depth-integrated integro-differential equations (see [Perego et al. 0000]). In all models, nonlinearity has been solved with Newton method, coupling LifeV with Trilinos NOX package. The LifeV based ice sheet implementation has been mentioned in [Evans et al. 2012] as an example of modern solver design for the solution of earth system models. Further, in [Tezaur et al. 2015] the authors verified the results of another ice sheet code with those obtained using LifeV.

LifeV ice sheet module has been coupled with the climate library MPAS and used in inter-comparison studies to asses sensitivities to different boundary conditions and forcing terms, see [Edwards et al. 2014] and [Shannon et al. 2013]. The latter ([Shannon et al. 2013]) has been considered in the IPCC (Intergovernmental Panel on Climate Change) report of 2014.

In [Perego et al. 2014] the authors perform a large scale PDE-constrained optimization to estimate the basal friction field (70K parameters) in Greenland ice sheet. For this purpose LifeV has been coupled with the Trilinos package ROL to perform a reduced-gradient optimization using BFGS. The assembly of state and adjoint equations and the computation of objective functional and its gradient have been performed in LifeV.

7. PERSPECTIVES

LifeV has proved to be a versatile library for the study of numerical techniques for large scale and multiphysics computations with finite elements. The code is in continuous development. The latest introduction of expression templates has increased the easiness of usage at the high level. Unfortunately, due to the fact the way the code has been developed, mainly by PhD students, not all the applications have been ported yet to this framework. Work is ongoing in that direction. The library has been coded using the C++98 yet porting is ongoing to exploit new features of the C++11 and C++14 standard, which can make the code more readable, user friendly and efficient.

As for the parallelization issues, LifeV relies strongly on the tools provided by the Trilinos libraries. We are following their development closely and we will be ready to integrate all the new features the library will offer, in particular with respect to hybrid type parallelism.

A target use of the library the team is working on is running on cloud facilities [Slawinski et al. 2012; Guzzetti et al. 2016] and GPL architectures.

ACKNOWLEDGMENTS

Besides the funding agencies cited in the introduction of this paper, we have to acknowledge all the developers of LifeV, who have contributed with new numerical methods, provided help to beginners, and struggled for the definition of a common path. Porting and fixing bugs is also a remarkable task to which they have constantly contributed. It is difficult to name people, the list would be anyway incomplete, a good representation of the contributors is given in the references below and of course on the developers website. Among other contributors we wish to mention Gilles Fourestey who has invested a lot of efforts to parallelise the code by introducing Trilinos and many of the concepts discribed in Section 2 and Daniele Di Pietro who actively worked on the Expression Templates part at the very beginning. A. Veneziani wishes to thank specifically (i) T. Passerini for the constant generous development and support not only in the development but also in the dissemination of LifeV among Emory students; (ii) M. Perego for reporting on the Ice-Sheet activity; (iii) U. Villa for the seminal and fundamental work in the Navier-Stokes modules that provided the basis of the current CFD in LifeV; (iv) L. Mirabella for bridging LifeV with the community of computational electrocardioologists.

Fausto Saleri introduced the name "LiFE" many years ago, in a 2D, Fortran 77 serial code for advection diffusion problems. Over the years, we made changes and additions, yet we do hope to have followed his enthusiasm, passion and dedication to scientific computing.

References

James Ahrens, Berk Geveci, and Charles Law. 2005. 36 - ParaView: An End-User Tool for Large-Data Visualization. In *Visualization Handbook*, Charles D. HansenChris R. Johnson (Ed.). Butterworth-Heinemann, Burlington, 717 – LXXII. DOI:http://dx.doi.org/10.1016/B978-012387582-2/50038-1

Matteo Aletti, Simona Perotto, and Alessandro Veneziani. 2015. *An educated-basis approach for the hierarchical model reduction of elliptic problems*. Tech report. Dept. Math and CS, Emory University.

Boris Andreianov, Mostafa Bendahmane, Alfio Quarteroni, and Ricardo Ruiz-Baier. 2015. Solvability analysis and numerical approximation of linearized cardiac electromechanics. *Math. Models Meth. Appl. Sci.* 25, 05 (2015), 959–993. DOI:http://dx.doi.org/10.1142/S0218202515500244

Luca Antiga, Marina Piccinelli, Lorenzo Botti, Bogdan Ene-Iordache, Andrea Remuzzi, and David A Steinman. 2008a. An image-based modeling framework for patient-specific computational hemodynamics. *Medical & biological engineering & computing* 46, 11 (2008), 1097–1112.

Luca Antiga, Marina Piccinelli, Lorenzo Botti, Bogdan Ene-Iordache, Andrea Remuzzi, and David A. Steinman. 2008b. An image-based modeling framework for patient-specific computational hemodynamics. *Medical & Biological Engineering & Computing* 46, 11 (2008), 1097–1112. DOI:http://dx.doi.org/10.1007/s11517-008-0420-1

F. Auricchio, M. Conti, A. Lefieux, S. Morganti, A. Reali, F. Sardanelli, F. Secchi, S. Trimarchi, and A. Veneziani. 2014. Patient-specific analysis of post-operative aortic hemodynamics: a focus on Thoracic Endovascular Repair (TEVAR). (2014). DOI:http://dx.doi.org/10.1007/s00466-014-0976-6

Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, Oliver Rheinbach, and Joerg Schrder. 2015. Numerical modeling of fluidstructure interaction in arteries with anisotropic polyconvex hyperelastic and anisotropic viscoelastic material models at finite strains. *International Journal for Numerical Methods in Biomedical Engineering* (2015), n/a–n/a. DOI:http://dx.doi.org/10.1002/cnm.2756 cnm.2756.

Eric Bavier, Mark Hoemmen, Sivasankaran Rajamanickam, and Heidi Thornquist. 2012. Amesos2 and Belos: Direct and Iterative Solvers for Large Sparse Linear Systems. *Scientific Programming* 20, 3 (2012), 241–255. DOI:http://dx.doi.org/10.3233/SPR-2012-0352

Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, and G. Scovazzi. 2007. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 197, 14 (2007), 173 – 201. DOI:http://dx.doi.org/10.1016/j.cma.2007.07.016

L Bertagna, A Quaini, and A Veneziani. 2015. Deconvolution-based nonlinear filtering for incompressible flows at moderately large Reynolds numbers. *Int J Num Meth Fluids* (2015).

Luca Bertagna and Alessandro Veneziani. 2014. A model reduction approach for the variational estimation of vascular compliance by solv- ing an inverse fluid-structure interaction problem. 30 (2014). DOI:http://dx.doi.org/10.1088/0266-5611/30/5/055006

PJ Blanco, LA Mansilla Alvarez, and RA Feijóo. 2015. Hybrid element-based approximation for the Navier–Stokes equations in pipe-like domains. *Computer Methods in Applied Mechanics and Engineering* 283 (2015), 971–993.

Pablo Javier Blanco, Simone Deparis, and Adelmo Cristiano Innocenza Malossi. 2013. On the continuity of mean total normal stress in geometrical multiscale cardiovascular problems. *Journal of Computational Physics* 251 (2013), 136–155. DOI:http://dx.doi.org/10.1016/j.jcp.2013.05.037

Jean Bonnemain, Simone Deparis, and Alfio Quarteroni. 2012. Connecting Ventricular Assist Devices to the Aorta: a Numerical Model. In *Imagine Math. Between Culture and Mathematics*, Michele Emmer (Ed.). Springer, 211–224.

Jean Bonnemain, Adelmo Cristiano Innocenza Malossi, Matteo Lesinigo, Simone Deparis, Alfio Quarteroni, and Ludwig von Segesser. 2013. Numerical simulation of left ventricular assist device implantations: comparing the ascending and the descending aorta cannulations. *Medical Engineering and Physics* 35, 10 (2013), 1465–1475. DOI:http://dx.doi.org/10.1016/j.medengphy.2013.03.022

D. Bonomi, C. Vergara, E. Faggiano, M. Stevanella, C. Conti, A. Redaelli, G. Puppini, G. Faggian, L. Formaggia, and G. B. Luciani. 2015. Influence of the aortic valve leaflets on the fluid-dynamics in aorta in presence of a normally functioning bicuspid valve. *Biomechanics and Modeling in Mechanobiology* 14, 6 (2015), 1349–1361. DOI:http://dx.doi.org/10.1007/s10237-015-0679-8

Abigail L. Bowers and Leo G. Rebholz. 2012. Increasing accuracy and efficiency in FE computations of the Leray-Deconvolution model. *Numerical Methods for Partial Differential Equations* 28, 2 (2012), 720–736. DOI:http://dx.doi.org/10.1002/num.20653

E. Burman and M. A. Fernández. 2007. Continuous interior penalty finite element method for the time-dependent Navier-Stokes equations: space discretization and convergence. *Numer. Math.* 107, 1 (2007), 39–77. DOI:http://dx.doi.org/10.1007/s00211-007-0070-5

J Cahouet and J-P Chabard. 1988. Some fast 3D finite element solvers for the generalized Stokes problem. *International Journal for Numerical Methods in Fluids* 8, 8 (1988), 869–895.

Antonio Cervone, Nur A. Fadel, and Luca Formaggia. 2012. Simulations of large three-dimensional sedimentary basin dynamics through domain decomposition technniques. In *ECCOMAS 2012 e-book*, J. Eberhardsteiner, H.J. Bhm, and F.G. Rammerstorfer (Eds.). Vienna University of Technology, 2652–2663.

Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. 2012. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. 357–372.

RH Clayton, Olivier Bernus, EM Cherry, Hans Dierckx, FH Fenton, L Mirabella, AV Panfilov, Frank B Sachse, G Seemann, and H Zhang. 2011. Models of cardiac tissue electrophysiology: progress, challenges and open questions. *Progress in biophysics and molecular biology* 104, 1 (2011), 22–48.

C. M. Colciago, S. Deparis, and A. Quarteroni. 2014. Comparisons between reduced order models and full 3D models for fluid-structure interaction problems in haemodynamics. *Journal Of Computational And Applied Mathematics* 265 (2014), 120–138. DOI:http://dx.doi.org/10.1016/j.cam.2013.09.049

Computational Engineering International, Inc. 2011. EnSight User Manual. (2011). https://www.ensight.com/.

P. Crosetto, S. Deparis, G. Fourestey, and A. Quarteroni. 2011a. Parallel algorithms for fluid-structure interaction problems in haemodynamics. *SIAM J. Sci. Comput.* 33, 4 (2011), 1598–1622. DOI:http://dx.doi.org/10.1137/090772836

P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, and A. Quarteroni. 2011b. Fluid-structure interaction simulation of aortic blood flow. *Comput. & Fluids* 43 (2011), 46–57. DOI:http://dx.doi.org/10.1016/j.compfluid.2010.11.032

Marco Del Pra, Alessio Fumagalli, and Anna Scotti. 2015. Well posedness of fully coupled fracture/bulk Darcy flow with XFEM. (May 2015). Submitted to: SIAM Journal on Numerical Analysis.

Marta D'Elia, Mauro Perego, and Alessandro Veneziani. 2012. A Variational Data Assimilation Procedure for the Incompressible Navier-Stokes Equations in Hemodynamics,. 52, 2 (2012), 340–359.

S. Deparis, M. Discacciati, G. Fourestey, and A. Quarteroni. 2006. Fluid-structure algorithms based on Steklov-Poincaré operators. *Comput. Methods Appl. Mech. Engrg.* 195, 41-43 (2006), 5797–5812. DOI:http://dx.doi.org/10.1016/j.cma.2005.09.029

S. Deparis, D. Forti, G. Grandperrin, and A. Quarteroni. 2015. FaCSI: A block parallel preconditioner for fluid-structure interaction in hemodynamics. (2015). Submitted.

S. Deparis, D. Forti, and A. Quarteroni. 2014. A Rescaled Localized Radial Basis Function Interpolation on Non-Cartesian and Nonconforming Grids. *SIAM Journal on Scientific Computing* 36 (2014), A2745–A2762.

S. Deparis, G. Grandperrin, and A. Quarteroni. 2013. Parallel preconditioners for the unsteady Navier–Stokes equations and applications to hemodynamics simulations. *Computers & Fluids* 0 (2013), –. DOI:http://dx.doi.org/10.1016/j.compfluid.2013.10.034

Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson, and Courtenay Vaughan. 2002. Zoltan Data Management Services for Parallel Dynamic Applications. *Computing in Science and Engineering* 4, 2 (2002), 90–97.

Daniele Antonio Di Pietro and Alessandro Veneziani. 2009a. Expression templates implementation of continuous and discontinuous Galerkin methods. *Computing and Visualization in Science* 12, 8 (2009), 421–436.

Daniele Antonio Di Pietro and Alessandro Veneziani. 2009b. Expression Templates Implementation of Continuous and Discontinuous Galerkin Methods. *Comput. Vis. Sci.* 12, 8 (Sept. 2009), 421–436. DOI:http://dx.doi.org/10.1007/s00791-008-0117-x

Marie Dupraz, Simonetta Filippi, Alessio Gizzi, Alfio Quarteroni, and Ricardo Ruiz-Baier. 2015. Finite element and finite volume-element simulation of pseudo-ECGs and cardiac alternans. *Math. Methods Appl. Sci.* 38, 6 (2015), 1046–1058. DOI:http://dx.doi.org/10.1002/mma.3127

T. L. Edwards, X. Fettweis, O. Gagliardini, F. Gillet-Chaulet, H. Goelzer, J. M. Gregory, M. Hoffman, P. Huybrechts, A. J. Payne, M. Perego, S. Price, A. Quiquet, and C. Ritz. 2014. Effect of uncertainty in surface mass balanceelevation feedback on projections of the future sea level contribution of the Greenland ice sheet. *The Cryosphere* 8, 1 (2014), 195–208. DOI:http://dx.doi.org/10.5194/tc-8-195-2014

H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. 2008. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *J. Comput. Phys.* 227, 3 (2008), 1790–1808. DOI:http://dx.doi.org/10.1016/j.jcp.2007.09.026

Howard C Elman, David J Silvester, and Andrew J Wathen. 2014. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press (UK).

Alexandre Ern and Jean Luc Guermond. 2006. *Theory and Practice of Finite Elements*. Springer.

Katherine J. Evans, Andrew G. Salinger, Patrick H. Worley, Stephen F. Price, William H. Lipscomb, Jeffrey A. Nichols, James B. White, Mauro Perego, Mariana Vertenstein, James Edwards, and Jean-Franois Lemieux. 2012. A modern solver interface to manage solution algorithms in the Community Earth Sys-

tem Model. *International Journal of High Performance Computing Applications* 26, 1 (2012), 54–62. DOI:http://dx.doi.org/10.1177/1094342011435159

E. Faggiano, L. Antiga, G. Puppini, A. Quarteroni, G.B. Luciani, and C. Vergara. 2013. Helical Flows and Asymmetry of Blood Jet in Dilated Ascending Aorta with Normally Functioning Bicuspid Valve. *Biomechanics and Modeling in Mechanobiology* 12(4) (2013), 801–813.

Alberto Ferroni, Luca Formaggia, and Alessio Fumagalli. 2016. Numerical analysis of Darcy problem on surfaces. *ESAIM: Mathematical Modelling and Numerical Analysis* (2016). in press, DOI 10.1051/m2an/2015095.

Luca Formaggia, Jean-Frédéric Gerbeau, Fabio Nobile, and Alfio Quarteroni. 2001. On the coupling of 3D and 1D Navier–Stokes equations for flow problems in compliant vessels. *Computer Methods in Applied Mechanics and Engineering* 191, 6 (2001), 561–582.

L. Formaggia, J. F. Gerbeau, F. Nobile, and A. Quarteroni. 2002. Numerical Treatment of Defective Boundary Conditions for the Navier–Stokes Equations. *SIAM J. Numer. Anal.* 40, 1 (2002), 376–401. DOI:http://dx.doi.org/10.1137/S003614290038296X

Luca Formaggia, Fausto Saleri, and Alessandro Veneziani. 2012. *Solving Numerical PDEs: Problems, Applications, Exercises: Problems, Applications, Exercises*. Springer Science & Business Media.

Luca Formaggia, Alessandro Veneziani, and Christian Vergara. 2008. A new approach to numerical solution of defective boundary value problems in incompressible fluid dynamics. *SIAM J. Numer. Anal.* 46, 6 (2008), 2769–2794.

Luca Formaggia, Alessandro Veneziani, and Christian Vergara. 2010. Flow rate boundary problems for an incompressible fluid in deformable domains: formulations and solution methods. *Computer Methods in Applied Mechanics and Engineering* 199, 9 (2010), 677–688.

D. Forti and L. Dedè. 2015. Semi–implicit BDF time discretization of the Navier–Stokes equations with VMS–LES modeling in High Performance Computing framework. *Computer & Fluids* 117 (2015), 168–182.

Pascal Frey. 2001. *MEDIT: An interactive mesh visualization software*. Technical Report RT-0253. Institut National de Recherche en Informatique et en Automatique.

Alessio Fumagalli. 2012. *Numerical Modelling of Flows in Fractured Porous Media by the XFEM Method*. Ph.D. Dissertation. Politecnico di Milano.

Alessio Fumagalli and Anna Scotti. 2011. Numerical modelling of multiphase subsurface flow in the presence of fractures. *Communications in Applied and Industrial Mathematics* 3, 1 (2011). DOI:http://dx.doi.org/10.1685/journal.caim.380

Geoffrey Furnish. 1997. Disambiguated glommable expression templates. 11 (1997), 263–269.

Alain Gauthier, Fausto Saleri, and Alessandro Veneziani. 2004. A fast preconditioner for the incompressible Navier Stokes Equations. *Computing and Visualization in science* 6, 2-3 (2004), 105–112.

M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. 2006. *ML 5.0 Smoothed Aggregation User's Guide*. Technical Report SAND2006-2649. Sandia National Laboratories.

Luca Gerardo-Giorda, L Mirabella, Fabio Nobile, Mauro Perego, and Alessandro Veneziani. 2009. A model-based block-triangular preconditioner for the Bidomain system in electrocardiology. *J. Comput. Phys.* 228, 10 (2009), 3625–3639. DOI:http://dx.doi.org/10.1016/j.jcp.2009.01.034

Luca Gerardo-Giorda, Mauro Perego, and Alessandro Veneziani. 2011. Optimized Schwarz coupling of Bidomain and Monodomain models in electrocardiology. *ESAIM: Mathematical Modelling and Numerical Analysis* 45, 02 (2011), 309–334.

Paola Gervasio. 2008. Convergence analysis of high order algebraic fractional step schemes for time-dependent Stokes equations. *SIAM J. Numer. Anal.* 46, 4 (2008), 1682–1703.

Paola Gervasio and Fausto Saleri. 2006. Algebraic fractional-step schemes for time-dependent incompressible Navier–Stokes equations. *Journal of Scientific Computing* 27, 1-3 (2006), 257–269.

Paola Gervasio, Fausto Saleri, and Alessandro Veneziani. 2006. Algebraic splitting methods for the incompressible Navier-Stokes Equations. 214, 1 (2006), 347–365.

Christophe Geuzaine and Jean-Franois Remacle. 2009. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Internat. J. Numer. Methods Engrg.* 79, 11 (2009), 1309–1331. DOI:http://dx.doi.org/10.1002/nme.2579

Alessio Gizzi, Ricardo Ruiz-Baier, Simone Rossi, Aymen Laadhari, Christian Cherubini, and Simonetta Filippi. 2015. A three-dimensional continuum model of active contraction in single cardiomyocytes. In *Modeling the Heart and the Circulatory System*, Alfio Quarteroni (Ed.). Springer-Verlag, Milano, 157–176. DOI:http://dx.doi.org/10.1007/978-3-319-05230-4_6

B. Gogas, L. Timmins, T. Passerini, Kim Piccinelli M, Molony S., Giddens D. D., S. King, and H. Samady. 2013. Biomechanical Assessment of Bioresorbable Devices. 6, 7 (2013), 760–761.

S. Guzzetti. 2014. *Hierarchical Model Reduction for the Incompressible Navier-Stokes Equations*. Master Thesis.

Sofia Guzzetti, Tiziano Passerini, Jaroslaw Slawinski, Umberto Villa, Alessandro Veneziani, and Vaidy Sunderam. 2016. Platform and algorithm effects on computational fluid dynamics applications in life sciences. *Future Generation Computer Systems* (2016).

S. Guzzetti, A. Veneziani, and V. Sunderam. 2015. *Performances of Overlapping Domain Decomposition Methods*. Tech rep. Dept. Math and CS, Emory University.

J. Härdtlein, C. Pflaum, A. Linke, and C. H. Wolters. 2009. Advanced expression templates programming. *Computing and Visualization in Science* 13, 2 (2009), 59–68. DOI:http://dx.doi.org/10.1007/s00791-009-0128-2

F. Hecht. 2012. New development in FreeFem++. *J. Numer. Math.* 20, 3-4 (2012), 251–265.

Michael A. Heroux. 2009. *Epetra Performance Optimization Guide*. Technical Report. Sandia National Laboratories.

Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. 2005. An overview of the Trilinos project. *ACM Trans. Math. Softw.* 31, 3 (2005), 397–423. DOI:http://dx.doi.org/10.1145/1089014.1089021

J.G. Heywood, R. Rannacher, and S. Turek. 1996. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *Int J Num Meth Fl* 22 (1996), 325–352.

Jonathan J. Hu, Andrey Prokopenko, Christopher M. Siefert, Raymond S. Tuminaro, and Tobias A. Wiesner. 2014. MueLu multigrid framework. http://trilinos.org/packages/muelu. (2014).

Klaus Iglberger, Georg Hager, Jan Treibig, and Ulrich Rüde. 2012. Expression templates revisited: a performance analysis of current methodologies. *SIAM Journal on Scientific Computing* 34, 2 (2012), C42–C69.

Guido Iori. 2011. *Una metodologia XFEM per problemi ellittici 3D con superfici di discontinuità*. Master's thesis. Politecnico di Milano. Advisor: Luca Formaggia. Co-advisor: Alessio Fumagalli.

G. Karypis and V. Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392.

Rocco M. Lancellotti. 2015. *Large eddy simulations in haemodynamics: models and applications*. Ph.D. Dissertation. Department of Mathematics, Politecnico di Milano, Italy.

Adelmo Cristiano Innocenza Malossi, Pablo Javier Blanco, Paolo Crosetto, Simone Deparis, and Alfio Quarteroni. 2013. Implicit coupling of one-dimensional and three-dimensional blood flow models with compliant vessels. *Multiscale Modeling and Simulation* 11, 2 (2013), 474–506. DOI:http://dx.doi.org/10.1137/120867408

Adelmo Cristiano Innocenza Malossi, Pablo Javier Blanco, and Simone Deparis. 2012. A two-level time step technique for the partitioned solution of one-dimensional arterial networks. *Computer Methods in Applied Mechanics and Engineering* 237-240 (2012), 212–226. DOI:http://dx.doi.org/10.1016/j.cma.2012.05.017

Adelmo Cristiano Innocenza Malossi, Pablo Javier Blanco, Simone Deparis, and Alfio Quarteroni. 2011. Algorithms for the partitioned solution of weakly coupled fluid models for cardiovascular flows. *International Journal for Numerical Methods in Biomedical Engineering* 27, 12 (2011), 2035–2057. DOI:http://dx.doi.org/10.1002/cnm.1457

Luis Mansilla Alvarez, Pablo Blanco, Carlos Bulant, Enzo Dari, Alessandro Veneziani, and Raúl Feijóo. 2016. Transversally enriched pipe element method (TEPEM): An effective numerical approach for blood flow modeling. *International Journal for Numerical Methods in Biomedical Engineering* (2016).

Lucia Mirabella, Christopher Haggerty, Passerini T., M. Piccinelli, Pedro J. Del Nido, Alessandro Veneziani, and Ajit P. Yoganathan. 2013. Treatment planning for a TCPC test case: a numerical investigation under rigid and moving wall assumptions. *Int J Num Meth Biomed Eng* 29, 2 (2013), 197–216.

L Mirabella, F Nobile, and A Veneziani. 2011. An a posteriori error estimator for model adaptivity in electrocardiology. *Computer Methods in Applied Mechanics and Engineering* 200, 37 (2011), 2727–2737.

F. Nobile, M. Pozzoli, and C. Vergara. 2013. Time accurate partitioned algorithms for the solution of fluid-structure interaction problems in haemodynamics. *Computer & Fluids* 86 (2013), 470–482.

F. Nobile, M. Pozzoli, and C. Vergara. 2014. Inexact accurate partitioned algorithms for fluid-structure interaction problems with finite elasticity in haemodynamics. *J. Comput. Phys.* 273 (2014), 598–617.

Fabio Nobile, Alfio Quarteroni, and Ricardo Ruiz-Baier. 2012. An active strain electromechanical model for cardiac tissue. *Int. J. Numer. Methods Biomed. Engrg.* 28, 1 (2012), 52–71. DOI:http://dx.doi.org/10.1002/cnm.1468

F. Nobile and C. Vergara. 2008. An effective fluid-structure interaction formulation for vascular dynamics by generalized Robin conditions. *SIAM J Sc Comp* 30, 2 (2008), 731–763.

MeshLab Web Page. 2016a. (2016). http://meshlab.sourceforge.net/

Netfabb Web Page. 2016b. (2016). https://www.netfabb.com/

TetGen Web Page. 2016c. (2016). http://wias-berlin.de/software/tetgen/

Tiziano Passerini, Mariarita de Luca, Luca Formaggia, Alfio Quarteroni, and Alessandro Veneziani. 2009. A 3D/1D geometrical multiscale model of cerebral vasculature. *Journal of Engineering Mathematics* 64, 4 (2009), 319–330.

Tiziano Passerini, Annalisa Quaini, Umberto Villa, Alessandro Veneziani, and Suncica Canic. 2013. Validation of an open source framework for the simulation of blood flow in rigid and deformable vessels. 29, 11 (2013), 1192–1213.

Tiziano Passerini, Laura M. Sangalli, Simone Vantini, Marinia Piccinelli, Susanna Bacigaluppi, Luca Antiga, Edoardo Boccardi, Piercesare Secchi, and Alessandro Veneziani. 2012. An Integrated Statistical Investigation of the Internal Carotid Ar- teries hosting Cerebral Aneurysms. 3, 1 (2012), 26–40.

S. V. Patankar and D. B. Spalding. 1972. A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows. *International J. on Heat and Mass Transfer* 15 (1972), 1787–1806. DOI:http://dx.doi.org/doi:10.1016/0017-9310(72)90054-3

Mauro Perego, Max Gunzburger, and John Burkardt. 2012-02-01T00:00:00. Parallel finite-element implementation for higher-order ice-sheet models. *Journal of Glaciology* 58, 207 (2012-02-01T00:00:00), 76–88. DOI:http://dx.doi.org/doi:10.3189/2012JoG11J063

Mauro Perego, Stephen Price, and Georg Stadler. 2014. Optimal initial conditions for coupling ice sheet models to Earth system models. *Journal of Geophysical Research: Earth Surface* 119, 9 (2014), 1894–1917. DOI:http://dx.doi.org/10.1002/2014JF003181

Mauro Perego and Alessandro Veneziani. 2009. An efficient generalization of the Rush-Larsen method for solving electro-physiology membrane equations. *Electronic Transactions on Numerical Analysis* 35 (2009), 234–256.

M. Perego, A. Veneziani, and C. Vergara. 2011. A variational approach for estimating the compliance of the cardiovascular tissue: an inverse fluid-structure interaction problem. *SIAM J Sc Comp* 33(3) (2011), 1181–1211.

Blair Perot. 1993. Algebraic splitting methods for the incompressible Navier-Stokes Equations. (1993).

Simona Perotto. 2014. A survey of hierarchical model (Hi-Mod) reduction methods for elliptic problems. In *Numerical simulations of coupled problems in engineering*. Springer, 217–241.

Simona Perotto, Alexandre Ern, and Alessandro Veneziani. 2010. Hierarchical Local Model Reduction for Elliptic Problems I: A Domain Decomposition Approach. 8, 4 (2010), 1102–1127.

Simona Perotto and Alessandro Veneziani. 2014. Coupled model and grid adaptivity in hierarchical reduction of elliptic problems. *Journal of Scientific Computing* 60, 3 (2014), 505–536.

Christoph Pflaum. 2001. Expression templates for partial differential equations. *Computing and Visualization in Science* 4, 1 (2001), 1–8.

R. Ponzini, C. Vergara, A. Redaelli, and A. Veneziani. 2006. Reliable CFD-based estimation of flow rate in haemodynamics measures. *Ultrasound in Med. and Biol.* 32(10) (2006), 1545–1555.

Radu Popescu. 2013. *Parallel algorithms and efficient implementation techniques for finite element approximations*. Ph.D. Dissertation. SB, Lausanne. DOI:http://dx.doi.org/10.5075/epfl-thesis-5980

Andrey Prokopenko, Jonathan J. Hu, Tobias A. Wiesner, Christopher M. Siefert, and Raymond S. Tuminaro. 2014. *MueLu Users Guide 1.0*. Technical Report SAND2014-18874. Sandia National Labs.

Alfio Quarteroni, Simone Rossi, Toni Lassila, and Ricardo Ruiz-Baier. 2015. Integrated Heart – Coupling multiscale and multiphysics models for simulation of the total heart function. *In preparation* (2015).

Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. 2000. Factorization methods for the numerical approximation of Navier–Stokes equations. *Computer methods in applied mechanics and engineering* 188, 1 (2000), 505–526.

Alfio Quarteroni, Alessandro Veneziani, and Christian Vergara. 2016. Geometric multiscale modeling of the cardiovascular system, between theory and practice. *Computer Methods in Applied Mechanics and Engineering* 302 (2016), 193–252.

Samuel Quinodoz. 2012. *Numerical Simulation of Orbitally Shaken Reactors*. Ph.D. Dissertation. SB. DOI:http://dx.doi.org/10.5075/epfl-thesis-5477

Maria Restrepo, Mark Luffel, Jake Sebring, Kirk Kanter, Pedro Del Nido, Alessandro Veneziani, Jarek Rossignac, and Ajit Yoganathan. 2015. Surgical planning of the total cavopulmonary connection: robustness analysis. *Annals of biomedical engineering* 43, 6 (2015), 1321–1334.

Simone Rossi, Toni Lassila, Ricardo Ruiz-Baier, Adelia Sequeira, and Alfio Quarteroni. 2014. Thermodynamically consistent orthotropic activation model capturing ventricular systolic wall thickening in cardiac electromechanics. *Eur. J. Mech. A/Solids* 48 (2014), 129–142. DOI:http://dx.doi.org/10.1016/j.euromechsol.2013.10.009

Simone Rossi, Ricardo Ruiz-Baier, Luca F. Pavarino, and Alfio Quarteroni. 2011. Active strain and activation models in cardiac electromechanics. *Proc. Appl. Math. Mech.* 11, 1 (2011), 119–120. DOI:http://dx.doi.org/10.1002/pamm.201110051

Simone Rossi, Ricardo Ruiz-Baier, Luca F. Pavarino, and Alfio Quarteroni. 2012. Orthotropic active strain models for the numerical simulation of cardiac biomechanics. *Int. J. Numer. Methods Biomed. Engrg.* 28, 6–7 (2012), 761–788. DOI:http://dx.doi.org/10.1002/cnm.2473

Ricardo Ruiz-Baier. 2015. Primal-mixed formulations for reaction-diffusion systems on deforming domains. *J. Comput. Phys.* in press (2015).

Ricardo Ruiz-Baier, Davide Ambrosi, Simone Pezzuto, Simone Rossi, and Alfio Quarteroni. 2013. Activation models for the numerical simulation of cardiac electromechanical interactions. In *Computer Models in Biomechanics: From Nano to Macro*, Gerhard Holzapfel and Ellen Kuhl (Eds.). Springer, Netherlands, 189–201. DOI:http://dx.doi.org/10.1007/978-94-007-5464-5_14

Ricardo Ruiz-Baier, Alessio Gizzi, Simone Rossi, Christian Cherubini, Aymen Laadhari, Simonetta Filippi, and Alfio Quarteroni. 2014. Mathematical modeling of active contraction in isolated cardiomyocytes. *Math. Medicine Biol.* 31, 3 (2014), 259–283. DOI:http://dx.doi.org/10.1093/imammb/dqt009

M. Sala and M. Heroux. 2005. *Robust Algebraic Preconditioners with IFPACK 3.0*. Technical Report SAND-0662. Sandia National Laboratories.

M. Sala, K. Stanley, and M. Heroux. 2006a. Amesos: A Set of General Interfaces to Sparse Direct Solver Libraries. In *Proceedings of PARA'06 Conference, Umea, Sweden*.

M. Sala, K. Stanley, and M. Heroux. 2006b. On the Design of Interfaces to Sparse Direct Solvers. *submitted* (2006).

Fausto Saleri and Alessandro Veneziani. 2005. Pressure Correction Algebraic Splitting Methods for the Incompressible Navier–Stokes Equations. *SIAM journal on numerical analysis* 43, 1 (2005), 174–194.

Frank R. Schaefer. 2007. GetPot, Powerful Input File and Command Line Parser. (2007). http://getpot.sourceforge.net/.

Joachim Schöberl. 1997. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science* 1, 1 (1997), 41–52. DOI:http://dx.doi.org/10.1007/s007910050004

Joachim Schöberl, Ken Martin, and Bill Lorensen. 2006. *The Visualization Toolkit*. Kitware. 4th edition.

S. R. Shannon, A. J. Payne, I. D. Bartholomew, M. R. Van Den Broeke, T. L. Edwards, X. Fettweis, O. Gagliardini, F. Gillet-chaulet, H. Goelzer, M. J. Hoffman, P. Huybrechts, D. W. F. Mair, P. W. Nienow, M. Perego, S. F. Price, C. J. P. P. Smeets, A. J. Sole, R. S. W. Van De Wal, and T. Zwinger. 2013. Enhanced basal lubrication and the contribution of the Greenland ice sheet to future sea-level rise. *Proceedings of the National Academy of Sciences of the United States of America - PNAS* 110, 35 (2013), 1415614161. DOI:http://dx.doi.org/10.1073/pnas.1212647110

Jaroslaw Slawinski, Tiziano Passerini, Umberto Villa, Alessandro Veneziani, and Vaidy Sunderam. 2012. Experiences with Target- Platform Heterogeneity in Clouds, Grids, and On-Premises Resources. In *Proceedings of HCW 2012 21st International Heterogeneity in Computing Workshop*, TO INSERT (Ed.).

Elaine Tang, Zhenglun Alan Wei, Kevin K Whitehead, Alessandro Veneziani, Mark A Fogel, and Ajit P Yoganathan. 2015. Respiratory pulsations affect fontan connection power loss: using real time velocity mapping to improve the accuracy of computational simulations. *Journal of Cardiovascular Magnetic Resonance* 17, 1 (2015), 1.

R. L. Taylor. 2014. FEAP - Finite Element Analysis Program. (2014). http://www.ce.berkeley/feap

I. K. Tezaur, M. Perego, A. G. Salinger, R. S. Tuminaro, and S. F. Price. 2015. Albany/FELIX: a parallel, scalable and robust, finite element, first-order Stokes approximation ice sheet solver built for advanced analysis. *Geoscientific Model Development* 8, 4 (2015), 1197–1220. DOI:http://dx.doi.org/10.5194/gmd-8-1197-2015

The HDF Group. 1997. Hierarchical Data Format, version 5. (1997). http://www.hdfgroup.org/HDF5/.

Paolo Tricerri, Luca Dede', Simone Deparis, Alfio Quarteroni, Anne M Robertson, and Adlia Sequeira. 2015. Fluid-structure interaction simulations of cerebral arteries modeled by isotropic and anisotropic constitutive laws. *Computational Mechanics -International Journal then Research Journal-* 55, 3 (2015), 479–498. DOI:http://dx.doi.org/10.1007/s00466-014-1117-y

G. H. W. van Bogerijen, F. Auricchio, M. Conti, A. Lefiueux, A. Reali, A. Veneziani, J.L. Tolenaar, V. Rampoldi F.L. Moll, and S. Trimarchi. 2014. Aortic Hemodynamics After Thoracic Endovascular Aortic Repair With the Role of Bird-Beak. 21 (2014), 791–802.

Todd Veldhuizen. 1995. Expression Templates. *C++ Report Magazine* 7 (1995), 26–31.

Alessandro Veneziani. 1998. *Mathematical and Numerical Modeling of Blood Flow Problems*. PhD.

Alessandro Veneziani. 2003. Block factorized preconditioners for high-order accurate in time approximation of the Navier-Stokes equations. *Numerical Methods for Partial Differential Equations* 19, 4 (2003), 487–510.

Alessandro Veneziani. 2009. A note on the consistency and stability properties of Yosida fractional step schemes for the unsteady stokes equations. *SIAM J. Numer. Anal.* 47, 4 (2009), 2838–2843.

Alessandro Veneziani. 2015. *Stent design and improvement: a matter of Mathematics too*. A. Gruenztig Center, Emory University.

A Veneziani and C Vergara. 2005. Flow rate defective boundary conditions in haemodynamics simulations. *International Journal for Numerical Methods in Fluids* 47, 8-9 (2005), 803–816.

A. Veneziani and C. Vergara. 2007. An approximate method for solving incompressible Navier-Stokes problems with flow rate conditions. *CMAME* 196(9-12) (2007), 1685–1700.

Alessandro Veneziani and Christian Vergara. 2013. Inverse problems in Cardiovascular Mathematics: toward patient-specific data assimilation and optimization. *International journal for numerical methods in biomedical engineering* 29, 7 (2013), 723–725.

A. Veneziani and U. Villa. 2012. ALADINS: an ALgebraic splitting time ADaptive solver for the Incompressible Navier-Stokes equations. *Journal Comput Phys.* (2012).

Christian Vergara, Matthias Lange, Simone Palamara, Toni Lassila, Alejandro F. Frangi, and Alfio Quarteroni. 2016. A coupled 3D1D numerical monodomain solver for cardiac electrical activation in the myocardium with detailed Purkinje network. *J. Comput. Phys.* 308 (2016), 218 – 238. DOI:http://dx.doi.org/10.1016/j.jcp.2015.12.016

C. Vergara, F. Viscardi, L. Antiga, and G.B. Luciani. 2012. Influence of bicuspid valve geometry on ascending aortic fluid-dynamics: a parametric study. *Artificial Organs* 36(4) (2012), 368–378.

Umberto Villa. 2011. *Efficient Numerical Methods*. PhD.

F. Viscardi, C. Vergara, L. Antiga, S. Merelli, A. Veneziani, G. Puppini, G. Faggian, A. Mazzucco, and G.B. Luciani. 2010. Comparative finite element model analysis of ascending aortic flow in bicuspid and tricuspid aortic valve. *Artificial organs* 34, 12 (Dec. 2010), 1114–20. DOI:http://dx.doi.org/10.1111/j.1525-1594.2009.00989.x

Boyi Yang, Bill Gogas, Gaetano Esposito, Olivia Hung, Emad Rasoul Arzrumly, Marina Piccinelli, Spencer King, Don Giddens, Alessandro Veneziani, and Habib Samady. 2015. Novel in-human four dimensional wall shear stress calculation of a coronary bioresorbable scaffold using optical coherence tomography images and blood flow simulations. *Journal of the American College of Cardiology* 65, 10_S (2015).

Huanhuan Yang. 2015. *Parameter Estimation and Reduced-order Modeling in Electrocardiology*. Ph.D. Dissertation. Emory University.

Huanhuan Yang and Alessandro Veneziani. 2013. *Variational Estimation of Cardiac Conductivities by a Data Assimilation Procedure*. Tech rep. Dept. of Math and CS, Emory University.

Received September 2016; revised -; accepted -

Online Appendix to: The LifeV library: engineering mathematics beyond the proof of concept

Luca Bertagna, Department of Mathematics and Computer Science, Emory University, Atlanta (GA) 30322 USA
Simone Deparis, CMCS–MATHICSE–SB, École Polytechnique Fédérale de Lausanne, Station 8, Lausanne, CH–1015, Switzerland (simone.deparis@epfl.ch)
Luca Formaggia, MOX, Dipartimento di Matematica, Politecnico di Milano, Italy (luca.formaggia@polimi.it)
Davide Forti, CMCS–MATHICSE–SB, École Polytechnique Fédérale de Lausanne, Station 8, Lausanne, CH–1015, Switzerland (davide.forti@epfl.ch)
Alessandro Veneziani, Department of Mathematics and Computer Science, Emory University, Atlanta (GA) 30322 USA (avenez2@emory.edu) .

DOI: 0000001.0000001