Use the template dedic.tex *together with the Springer Nature document class SVMono for monograph-type books or SVMult for contributed volumes to style a quotation or a dedication at the very beginning of your book*

Foreword

Place *Foreword author*

Preface

Place, Date *Editors*

Acknowledgements

Use the template *acknow.tex* together with the document class SVMono (monograph-type books) or SVMult (edited books) if you prefer to set your acknowledgement section as a separate chapter instead of including it as last part of your preface.

Contents

List of Contributors

Acronyms

Use the template *acronym.tex* together with the document class SVMono (monograph-type books) or SVMult (edited books) to style your list(s) of abbreviations or symbols.

Lists of abbreviations, symbols and the like are easily formatted with the help of the Springer Nature enhanced `description` environment.

ABC Spelled-out abbreviation and definition

BABI Spelled-out abbreviation and definition

CABR Spelled-out abbreviation and definition

Chapter 1
Sample title for chapter 1

U. N. Owen and U. N. Owena

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Introduction

Sample references Alnæs et al. (2015); Baratta et al. (2023). Source code for this book is found at github.com/meg-simula/2024-fenics-proceedings

Acknowledgements Please including any acknowledgements, including funding, here.

U. N. Owen e-mail: email@place.edu
Institution

References

Alnæs MS, Blechta J, Hake JE, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The fenics project version 1.5. Archive of Numerical Software 3

Baratta IA, Dean JP, Dokken JS, Habera M, Hale JS, Richardson CN, Rognes ME, Scroggs MW, Sime N, Wells GN (2023) DOLFINx: The next generation FEniCS problem solving environment. doi:10.5281/zenodo.10447666, URL https://doi.org/10.5281/zenodo.10447666

Chapter 2
The FEniCS Project on AWS Graviton3

M. Habera and J. S. Hale

Abstract ARM architecture central processing units are increasingly prevalent in high performance computers due to their energy efficiency, scalability and cost-effectiveness. The overall goal of this study is to evaluate the suitability of ARM-based cloud computing instances for executing finite element computations. Specifically, we show performance results executing the FEniCS Project finite element software on Amazon Web Services (AWS) c7g and c7gn instances with Graviton3 processors. These processors support ARMv8.4-A instruction set with Scalable Vector Extensions (SVE) for Single Instruction Multiple Data operations and the Elastic Fabric Adaptor for communications between instances. Both clang 18 and GCC 13 compilers successfully generated optimized code using SVE instructions which ensures that users can achieve optimized performance without extensive manual tuning. Testing a distributed memory parallel DOLFINx Poisson solver with up to 512 Message Passing Interface processes, we found that the performance and scalability of the AWS instances are comparable to a dedicated AMD EPYC Rome cluster installed at the University of Luxembourg. These findings demonstrate that ARM-based cloud computing instances, exemplified by AWS Graviton3, can be competitive for distributed memory parallel finite element analysis.

Introduction

The FEniCS Project (Alnæs et al., 2015; Baratta et al., 2023b) has been used to write finite element solvers for problems arising in fields that involve the solution of partial

M. Habera e-mail: michal.habera@rafinex.com
Rafinex SARL, Luxembourg and Institute of Computational Engineering, Department of Engineering, Faculty of Science, Technology and Medicine, University of Luxembourg, Luxembourg.
J. S. Hale e-mail: jack.hale@uni.lu
Institute of Computational Engineering, Department of Engineering, Faculty of Science, Technology and Medicine, University of Luxembourg, Luxembourg.

differential equations (PDEs), including mathematics, biology, physics, engineering, geophysics and mechanics.

Exploring ARM-based processors and cloud computing instances for executing FEniCS Project-based solvers is worthwhile due to ARMs potential advantages in cost-effectiveness, energy efficiency and scalability with respect to x86-64-based machines Simakov et al. (2023); Suárez et al. (2024). Examples of adoption of ARM in the HPC space include the Isambard project (Isambard 3, NVIDIA Grace, (BCS, 2025)), Mont-Blanc project (Phase 3, Cavium Thunder X2, (Rajovic et al., 2016)), Fugaku supercomputer (Fujitsu A64FX, (Fujitsu, 2024)) and Astra supercomputer (Cavium Thunder X2, (Sandia, 2018)). The publically available cloud services with ARM instances include Amazon Web Services (AWS) (Graviton3 CPU based on Neoverse V1 and Graviton4 CPU with Neoverse V2), Google Cloud (Axion CPU based on Neoverse V2, (Google, 2025)) and Microsoft Azure (Azure Cobalt 100 based on Neoverse N2, (Microsoft, 2024)).

AWS Graviton3-based instances aim to provide cost effective compute resources for scientific computing and machine-learning applications by including both Scalable Vector Extension (SVE) instructions for Single Instruction Multiple Data (SIMD) parallelism and the Elastic Fabric Adaptor (EFA) interconnect for high-bandwidth low-latency communication between instances. This makes the AWS cloud offering particularly appealing for executing scientific computing codes, like the FEniCS Project.

A key technology in the FEniCS Project is the use of automatic code generation (compilation). The user expresses their finite element problem in the Unified Form Language (UFL) (Alnæs et al., 2014) and then the FEniCSx Form Compiler (FFCx) (Kirby and Logg, 2006) compiles the UFL description of the problem into a low-level C kernel for computing the cell-local finite element tensor.

One aspect of good performance of a compute-bound kernel is ensuring the assembly code of the compiled kernel contains calls to Single Instruction Multiple Data (SIMD) operations. SIMD operations can apply the same operation to multiple data items in a single CPU clock cycle. For a recent overview of SIMD programming strategies see e.g. (Rocke, 2023). The current strategy of FFCx with respect to SIMD is to ensure that its kernels are amenable to the compiler applying automatic vectorisation, a process that automatically converts a scalar program into a vectorised equivalent that uses SIMD operations.

Consequently for users to achieve good performance when using FEniCSx on Graviton3 it is important to verify that the latest compilers do automatically emit SVE and/or Neon SIMD instructions when compiling the generated C finite element kernels and that these kernels achieve reasonable runtime performance.

In addition to SIMD parallelisation at the kernel level, DOLFINx, the finite element problem solving environment of the FEniCS Project, also supports distributed memory parallel assembly of global finite element data structures (sparse matrices and vectors) using the Message Passing Interface (MPI), for full details see Baratta

et al. (2023b). For user's to run large-scale DOLFINx simulations on AWS it is necessary to verify that the EFA interconnect provides sufficient performance for parallel scalability.

In summary, the contribution of this chapter is to examine both SIMD performance and multi-node parallel scaling of the FEniCS Project software on Amazon's Graviton3 based instances.

Methodology and Results

Systems

AWS c7g and c7gn instances are compared to Aion computing instances available at the University of Luxembourg HPC facilities (Varrette et al., 2022). These instances have different hardware configuration, see Table 2.1 for full details.

The FEniCS Project components are written in a mixture of Python, modern-style C++20 and Standard C17. The Python interface is a wrapper around the core data structures and computationally intensive algorithms written in C and C++.

	Aion node	AWS c7g instance
Processor	2 x (AMD Epyc ROME 7H12, 64 cores @ 2.6 GHz)	1 x (Graviton3, 64 cores @ 2.6 GHz)
Architecture	x86_64, Zen 2 (AVX2)	ARMv8.4-A, Neoverse V1 (SVE)
Memory	256 GB DDR4 3200 MT/s = 25.6 GB/s 8 NUMA nodes	128 GB DDR5 4800 MT/s = 38.4 GB/s Unified Memory Access (no NUMA)
Total mem. bandwidth	2 x 200 GB/s	1 x 300 GB/s

Table 2.1: Configuration of the Aion nodes (University of Luxembourg HPC) and AWS c7g (Amazon) instances. The c7gn instance used in the Poisson weak scaling test has the same hardware as c7g with the addition of a 200 GB s^{-1} interconnect between instances for MPI-based communication.

Memory bandwidth

Low-order finite element methods are typically memory bandwidth constrained as the time taken to load and store data from main memory (e.g. the mesh geometry) dominates the time taken to perform the arithmetic operations to compute the fi-

nite element cell tensor. Understanding the memory bandwidth characteristics of a processor is therefore important for ensuring optimal performance.

STREAM (McCalpin, 1995, 1991-2007) is the industry standard benchmark for measuring sustained memory bandwidth performance. They estimate memory bandwidth from memory intense operations (copy, scale, add) on large contiguous arrays.

In Figure 2.1 results for the copy operation for single-node benchmark are shown. For the single-node benchmark 80 % of the theoretical peak memory bandwidth of 400 GB s^{-1} for Aion and 300 GB s^{-1} for AWS c7g is reached. This is considered a reasonable outcome of the STREAM benchmark, (McCalpin, 2023). Bandwidth saturation is observed at around 20 % of the node utilisation. Both curves show different characteristics of the saturation point due to different memory access configuration. On the Aion instances there are 8 non-unified memory access (NUMA) nodes of 16 cores each, while AWS c7g instances are setup with unified memory access.

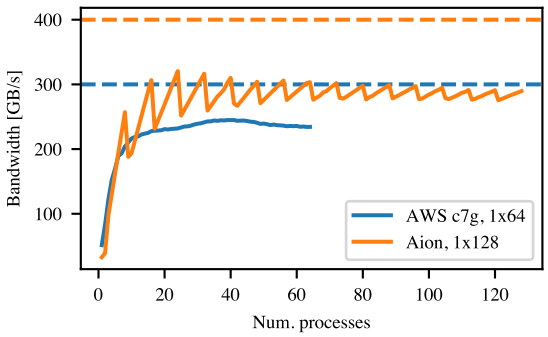


Fig. 2.1: Single-node STREAM benchmark. Theoretical peak bandwidth of each system show as dashed line.

Finite element kernels

In order to measure the performance of a standard FEniCS user finite element code we used the Local Finite Element Operator Benchmarks repository (Baratta et al., 2023a). The benchmark measures execution time for local finite element kernel generated by the FEniCS Form Compiler (FFCx) v0.9.0 (Kirby and Logg, 2006). We generate a matrix-free three-dimensional Laplace kernel representing a finite element discretisation of the action of Laplace operator A_{ij} with spatially varying material property $\kappa(x)$

$$v_i = A_{ij} w_j, \quad A_{ij} = \int_K \kappa J_{mk} J_{mn} \nabla_k \phi_i \nabla_n \phi_j |\det J| \mathrm{d}x, \tag{2.1}$$

where K is a fixed reference tetrahedron, $w_j \in \mathbb{R}^n$ is a fixed, prescribed vector, J is a Jacobian transformation matrix and ϕ are finite element basis functions.

The generated kernel calculates a double precision vector $v_i \in \mathbb{R}^n$, where $n = 4$ for first-order discretization (low-order) and $n = 165$ for eight-order discretization (high-order). Low-order kernels are expected to be memory bandwidth limited, while high-order kernels have higher arithmetic intensity. In addition, the matrix-free (operator action) version requires fewer load and store operations in comparison to the assembly of a matrix, increasing the ratio of floating-point operations to memory loads and stores. Consequently for the high-order kernels there is the scope for significant performance increases if the compiler can automatically emit SIMD operations.

Generated code structure

Compiler (loop) SIMD auto-vectorisation is usually performed for inner-most loops with compile-time known bounds. The analysis of FFCx autogenerated code is required to understand the potential and missed optimisations.

Code Listing 2.1: Abbreviated FFCx generated finite element kernel.

```
void kernel(double* restrict A, const double* restrict w, ...){
   // 1. Static arrays of basis functions and quadrature weights.
   // 2. Quadrature rule independent computations.

   for (int iq = 0; iq < NUM_QUAD_POINTS; ++iq) {
      // 3. Quadrature loop body.
      for (int ic = 0; ic < NUM_DOFS; ++ic){
         // 3.1 Coefficient evaluation.
         w1_d100 += w[4 + (ic)] * FE0_C0_D100_Q530[0][0][iq][ic];
         // ...
      }

      // 3.2 Scalar graph evaluation.
      double sv_530_0 = w1_d100 * sp_530_18;
      double sv_530_1 = w1_d010 * sp_530_22;
      // ...

      for (int i = 0; i < NUM_DOFS; ++i) {
         // 3.3 Tensor assignment loop.
         A[(i)] += fw0 * FE0_C0_D100_Q530[0][0][iq][i];
         // ...
      }
   }
}
```

An abbreviated example of generated C code is shown in Code Listing 2.1. Firstly, there are arrays defining finite element basis functions at quadrature points. These require no arithmetic operations. Computations independent of the quadrature loop contain more intense arithmetic operations (e.g. determinant of the Jacobian), but are executed only once. Non-affine geometry would require evaluation of geometric

quantities at each quadrature point, which would increase the arithmetic intensity and yield more opportunities for vectorisation.

The most performance critical part of the code is contained in the quadrature loop body. For the eight-order Laplace operator there is `NUM_QUAD_POINTS = 214` and `NUM_DOFS = 165`. There are two inner-most loops: coefficient evaluation and tensor assignment. Both contain a set of multiply-add operations which are candidates for automatic vectorisation via fused multiply-add operations in both SVE (Graviton3) and AVX2 (AMD EPYC).

Experimental results

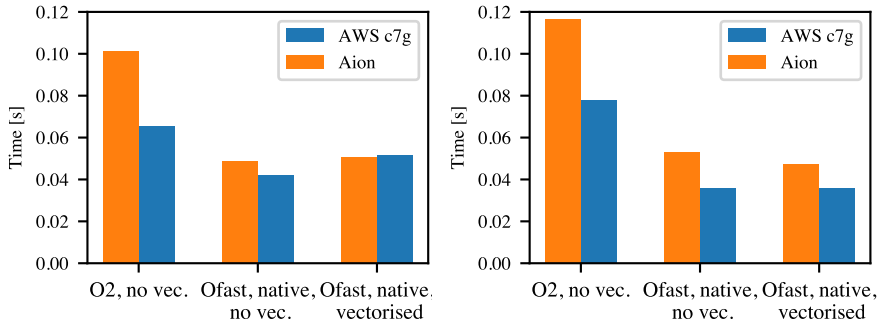
For the finite element kernel benchmarks we compiled the kernels with LLVM/clang 18.1.3 and GCC 13.2.0. Full details are given in Table 2.2.

	Compiler	Aion	AWS c7g
Ofast, native, vectorized	GCC 13.2.0	-Ofast -march=znver2 -mtune=znver2	-Ofast -mcpu=neoverse-v1
	clang 18.1.3	-Ofast -march=znver2 -mtune=znver2	-Ofast -mcpu=neoverse-v1
Ofast, native, no vec.	GCC 13.2.0	-Ofast -march=znver2 -mtune=znver2 -fno-tree-vectorize	-Ofast -mcpu=neoverse-v1 -fno-tree-vectorize
	clang 18.1.3	-Ofast -march=znver2 -mtune=znver2 -fno-slp-vectorize -fno-vectorize	-Ofast -mcpu=neoverse-v1 -fno-slp-vectorize -fno-vectorize
O2, no vec.	GCC 13.2.0	-O2 -fno-tree-vectorize	-O2 -fno-tree-vectorize
	clang 18.1.3	-O2 -fno-slp-vectorize -fno-vectorize	-O2 -fno-slp-vectorize -fno-vectorize

Table 2.2: Compiler versions and compilation flags used for finite element kernel benchmarks.

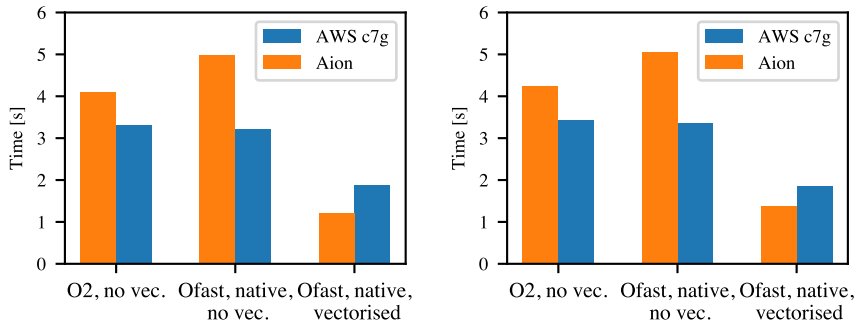
Results for kernel benchmarks are shown in Figure 2.2 and Figure 2.3. Low-order kernels (Figure 2.2) show no dependence on compiler vectorisation setup. On the other hand, AWS c7g shows 1.3x speed-up over Aion which we attribute to higher memory bandwidth for a single process.

High-order kernels (Figure 2.3), which are expected to benefit from SIMD operations, show a clear link between compiler settings and performance. Both clang and GCC auto-vectorisers perform well, producing a noticeable speed-up (>2x) in the most optimised setting. The vectorisation speed-up (>4x) is more significant with the Aion nodes.



(a) clang 18.1.3. (b) GCC 13.2.0.

Fig. 2.2: Low-order Laplace operator action assembly.



(a) clang 18.1.3. (b) GCC 13.2.0.

Fig. 2.3: High-order Laplace operator action assembly.

Optimisation reports (`-Rpass=loop-vectorize` for clang, `-fopt-info-vec-optimized` for GCC) and the inspection of the generated assembly reveal that the low-order operator action the compiler optimisation level `-Ofast` makes constant folding more effective and pre-computes more operations at compile-time (e.g. partial sums of the static constant arrays) (Godbolt, 2024e).

On Graviton3, both GCC and clang generate SVE FMLA instructions (Arm, 2024) for both the coefficient evaluation and tensor assignment loops (Godbolt, 2024b,a). FMLA, or Floating-point fused Multiply-Add, is a SIMD instruction that multiplies two vectors stored in SVE registers and adds the result to a third vector. The coefficient evaluation loop with no interdependencies between iterations is a perfect example for compiler auto-vectorisation. Moreover, for coefficients of higher order discretization, there is potential for exploiting wider SVE registers (up to 2048 bits).

An assembly excerpt for the coefficient evaluation is shown below.

```
ld1d {z0.d}, p0/z, [x7, x0, lsl #3]
ld1d {z25.d}, p0/z, [x3, x0, lsl #3]
fmla z3.d, p0/m, z25.d, z0.d
...
faddv d1, p1, z1.d
```

As expected, there are two contiguous loads LD1D into two of the available SVE Z0-Z31 registers followed by a fused Multiply-Add instruction. The result is accumulated into an SVE register Z3 which is then horizontally summed outside of the vectorised loop (FADDV). Here P0 is a predicate register without any constraints on the available elements.

On Aion, both GCC and clang vectorise both coefficient evaluation and tensor assignment loops and rely on the `VFMADD231PD` instructions on the YMM registers, i.e. vectorisation width of 4 doubles (Godbolt, 2024c,d).

Parallel scalability

Results for the parallel scalability were produced using performance test codes for FEniCSx (Wells and Richardson, 2023) built against DOLFINx 0.6.0 and PETSc 3.18 (Balay et al., 2023) with the Spack package manager setup to use GCC 12.2.0. We setup Spack to use a version of OpenMPI provided by AWS which includes the appropriate libfabric with native support for the EFA interconnect. Libfabric is a network communication library that abstracts networking technologies from fabric and hardware implementation, ensuring optimal data transfer across Amazon's proprietary EFA interconnect.

The Poisson equation solver benchmark consists of the following measured steps:

1. Create mesh. Create a unit cube mesh and discretise using linear tetrahedral cells. Partition the mesh with Parmetis 4.0.3 partitioner (Karypis and Kumar, 1998) and distribute.
2. Assemble matrix. Execute the local Poisson equation kernel over the mesh and assemble into a PETSc MATMPIAIJ (distributed compressed sparse row) matrix.

3. Solve linear system. Run Conjugate Gradient (CG) solver with a classical algebraic multigrid (BoomerAMG (Falgout and Yang, 2002)) preconditioner.

Creating the mesh (including partitioning), assembling matrices and solving the resulting linear system are typically the most expensive steps in a finite element solve. They also contain significant parallel communication steps that can highlight issues in either the finite element solver, or the underlying MPI hardware/software stack, leading to poor parallel scaling. Weak scaling results (constant workload of approx. 5×10^5 degrees-of-freedom per process) are shown in Figure 2.4. Both Aion and AWS c7gn show almost constant times for mesh creation (< 5% difference).

Matrix assembly is expected to have ideal weak parallel scalability due to the cell-local nature of the assembly loop and negligible amount of MPI communication during matrix finalisation. Aion and AWS c7gn show small increase in time (10-15 %) for 512 processes.

The time for the solve step increases by 40 % for 512 processes on AWS c7gn and by 27 % on Aion. However, the number of Krylov iterations of the preconditioned CG solver grows from 16 to 20 for 512 processes (25% increase) due to the inefficiency of the algebraic multigrid preconditioner on an unstructured 3D mesh. Taking this into account, the time per iteration is almost constant on Aion (< 5%) and a small increase of 15 % on AWS c7gn is observed.

Conclusions

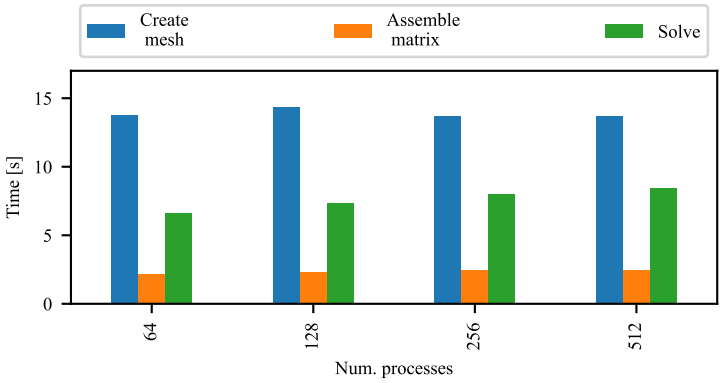
Benchmarks for memory bandwidth, local finite element kernels and parallel scalability of Poisson solver were executed on Aion nodes and on AWS c7g(n) instances.

Memory bandwidth measured using STREAM MPI confirms higher memory transfer rate of AWS c7g(n), but a superior total bandwidth of $310\,\mathrm{GB\,s^{-1}}$ per Aion node.

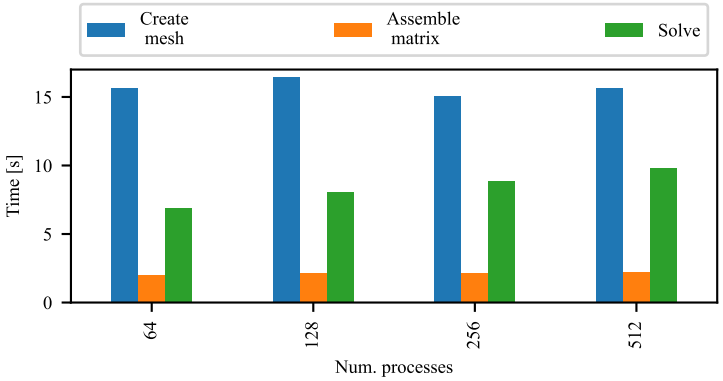
In terms of auto-vectorisation capabilities of GCC 13.2.0 and clang 18.1.3, both produced optimised instructions for the targeted microarchitectures (Zen 2 for Aion and Neoverse V1 for AWS c7g). This observation was confirmed with performance benchmarks based on local finite element kernels for the Laplace operator.

The MPI-based distributed memory Poisson equation solver shows weak scaling with 15 % time per iteration increase for 512 processes on the c7gn-based cluster. Results for the in-house University of Luxembourg Aion system are slightly superior with almost constant (< 5% difference) time per iteration for 512 processes.

Based on our results, we conclude that AWS Graviton3 instances are a viable alternative for high-performance computing tasks using the FEniCS Project automated finite element solver. These instances are likely to be particularly interesting for users



(a) Aion, 5×10^5 degrees-of-freedom per process, 25 % utilisation (32 processes per node).



(b) AWS c7gn, 5×10^5 degrees-of-freedom per process, 50 % utilisation (32 processes per node).

Fig. 2.4: Weak parallel scalability of the DOLFINx Poisson equation solver on Aion and AWS c7gn systems.

with infrequent or highly elastic large-scale computational demands Emeras et al. (2016).

In future work we plan to work on other more complex problems (e.g. linear elasticity) and performance benchmarks of direct solvers. Additionally, the latest generation Graviton4 instances provide an improved Neoverse V2 instruction set, which has a smaller SVE vector length of 128 bits, (Arm, 2025), which warrants further investigation.

Supplementary material

Raw data and plotting scripts are archived at (Habera and Hale, 2025).

Acknowledgements This project has received compute resources from Amazon Web Services (AWS) through the first and second collaborative University of Luxembourg and AWS Graviton3 call. The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg Varrette et al. (2022) – see https://hpc.uni.lu

This research was funded in whole, or in part, by the National Research Fund (FNR), grant reference COAT/17205623. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

Alnæs MS, Blechta J, Hake JE, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The FEniCS Project Version 1.5. Archive of Numerical Software 3, doi:10.11588/ans.2015.100.20553

Alnæs MS, Logg A, Ølgaard KB, Rognes ME, Wells GN (2014) Unified Form Language: A Domain-specific Language for Weak Formulations of Partial Differential Equations. ACM Trans Math Softw 40(2):9:1–9:37, doi:10.1145/2566630, URL http://doi.acm.org/10.1145/2566630

Arm (2024) Arm Architecture Reference Manual for A-profile architecture. https://developer.arm.com/documentation/ddi0487/ka, [Accessed 11-09-2024]

Arm (2025) Arm neoverse v2 core technical reference manual. https://developer.arm.com/documentation/102375/latest/, [Accessed 11-01-2025]

Balay S, et al. (2023) PETSc Web page. URL https://petsc.org/

Baratta I, Richardson C, Dokken JS, Hermano A (2023a) Local Finite Element Operator Benchmarks. URL https://github.com/IgorBaratta/local_operator

Baratta IA, Dean JP, Dokken JS, Habera M, Hale JS, Richardson CN, Rognes ME, Scroggs MW, Sime N, Wells GN (2023b) DOLFINx: The next generation FEniCS problem solving environment. doi:10.5281/zenodo.10447666

BCS (2025) Specs - Bristol Centre for Supercomputing Documentation — docs.isambard.ac.uk. https://docs.isambard.ac.uk/specs/, [Accessed 12-01-2025]

Emeras J, Varrette S, Bouvry P (2016) Amazon Elastic Compute Cloud (EC2) vs. In-House HPC Platform: A Cost Analysis. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pp 284–293, doi:10.1109/CLOUD.2016.0046, URL https://ieeexplore.ieee.org/document/7820283

Falgout RD, Yang UM (2002) hypre: A Library of High Performance Preconditioners. In: Sloot PMA, Hoekstra AG, Tan CJK, Dongarra JJ (eds) Computational Science — ICCS 2002, no. 2331 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp 632–641, doi:10.1007/3-540-47789-6_66

Fujitsu (2024) Supercomputer Fugaku retains first place worldwide in HPCG and Graph500 rankings — fujitsu.com. https://www.fujitsu.com/global/about/resources/news/press-releases/2024/1119-01.html, [Accessed 10-01-2025]

Godbolt M (2024a) Compiler Explorer - (ARM64 gcc 13.2.0). https://godbolt.org/z/sxGo17Wq9, [Accessed 11-09-2024]

Godbolt M (2024b) Compiler Explorer - high-order (armv8-a clang 18.1.0). URL https://godbolt.org/z/WzYEefEGK, [Accessed 11-09-2024]

Godbolt M (2024c) Compiler Explorer - high-order (x86-64 clang 18.1.0). https://godbolt.org/z/fEz64zzWx, [Accessed 11-09-2024]

Godbolt M (2024d) Compiler Explorer - high-order (x86-64 gcc 13.2). https://godbolt.org/z/aYeYcb6z1, [Accessed 11-09-2024]

Godbolt M (2024e) Compiler Explorer - low-order (armv8-a clang 18.1.0). https://godbolt.org/z/4Mdbvndrf, [Accessed 11-09-2024]

Google (2025) Arm VMs on Compute | Compute Engine Documentation | Google Cloud — cloud.google.com. https://cloud.google.com/compute/docs/instances/arm-on-compute, [Accessed 12-01-2025]

Habera M, Hale JS (2025) Supplementary material: The FEniCS Project on AWS Graviton3. doi:10.5281/zenodo.13748404

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1):359–392, doi:10.1137/s1064827595287997, URL http://dx.doi.org/10.1137/S1064827595287997

Kirby RC, Logg A (2006) A Compiler for Variational Forms. ACM Trans Math Softw 32(3):417–444, doi:10.1145/1163641.1163644, URL http://doi.acm.org/10.1145/1163641.1163644

McCalpin JD (1991-2007) STREAM: Sustainable memory bandwidth in high performance computers. Tech. rep., University of Virginia, Charlottesville, Virginia, URL http://www.cs.virginia.edu/stream/

McCalpin JD (1995) Memory Bandwidth and Machine Balance in Current High Performance Computers. IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter pp 19–25

McCalpin JD (2023) The evolution of single-core bandwidth in multicore processors — sites.utexas.edu. https://sites.utexas.edu/jdm4372/2023/04/25/the-evolution-of-single-core-bandwidth-in-multicore-processors/, [Accessed 10-01-2025]

Microsoft (2024) Announcing the preview of new Azure VMs based on the Azure Cobalt 100 processor | Microsoft Community Hub — techcommunity.microsoft.com. https://techcommunity.microsoft.com/blog/azurecompute/announcing-the-preview-of-new-azure-vms-based-on-the-azure-cobalt-100-processor/4146353, [Accessed 12-01-2025]

Rajovic N, Rico A, Mantovani F, Ruiz D, Vilarrubi JO, Gomez C, Backes L, Nieto D, Servat H, Martorell X, Labarta J, Ayguade E, Adeniyi-Jones C, Derradji S, Gloaguen H, Lanucara P, Sanna N, Mehaut JF, Pouget K, Videau B, Boyer E, Allalen M, Auweter A, Brayford D, Tafani D, Weinberg V, Brommel D, Halver R, Meinke JH, Beivide R, Benito M, Vallejo E, Valero M, Ramirez A (2016) The mont-blanc prototype: An alternative approach for hpc systems. In: SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, p 444–455, doi:10.1109/sc.2016.37, URL http://dx.doi.org/10.1109/SC.2016.37

Rocke FJ (2023) Evaluation of C++ SIMD Libraries. Bachelor's thesis, Der Ludwig-Maximilians-Universität München, URL https://www.mnm-team.org/pub/Fopras/rock23/

Sandia NL (2018) Astra supercomputer at Sandia Labs is fastest Arm-based machine on TOP500 list — sandia.gov. https://www.sandia.gov/labnews/2018/11/21/astra-2/, [Accessed 10-01-2025]

Simakov NA, Deleon RL, White JP, Jones MD, Furlani TR, Siegmann E, Harrison RJ (2023) Are we ready for broader adoption of ARM in the HPC community: Performance and Energy Efficiency Analysis of Benchmarks and Applications Executed on High-End ARM Systems. In: Proceedings of the HPC Asia 2023 Workshops, Association for Computing Machinery, New York, NY, USA, HPCAsia '23 Workshops, pp 78–86, doi:10.1145/3581576.3581618

Suárez D, Almeida F, Blanco V (2024) Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs. The Journal of Supercomputing 80(9):12771–12789, doi:10.1007/s11227-024-05946-9

Varrette S, Cartiaux H, Peter S, Kieffer E, Valette T, Olloh A (2022) Management of an Academic HPC & Research Computing Facility: The ULHPC Experience 2.0. In: Proc. of the 6th ACM High Performance Computing and Cluster Technologies Conf. (HPCCT 2022), Association for Computing Machinery (ACM), Fuzhou, China, doi:10.1145/3560442.3560445

Wells G, Richardson C (2023) Performance test codes for FEniCSx. URL `https://github.com/FEniCS/performance-test`

Chapter 3

Blood flow in the beating heart: Coupling fluid dynamics to reduced wall and circulation models for data-driven cardiac FSI

Marc Hirschvogel, Mia Bonini, Maximilian Balmus, David Nordsletten

Abstract We present a fluid-reduced-solid interaction (FrSI) approach suitable for modeling blood flow in the beating left heart. The method uses image-derived model data to construct a suitable boundary motion space, enhanced by a reduced solid mechanics wall model to enable adaptive fluid motion. The method combines the efficiency of fluid dynamics models with features from full FSI approaches, uniquely integrating motion data to predict cardiac hemodynamics over a full heart cycle. The approach is presented for a patient-specific left heart model coupled to a lumped circulatory system, showing physiological flow behavior and pressure-volume relations.

Introduction

Computational fluid dynamics (CFD) provides a valuable tool to predict blood flow in the cardiovascular system Schwarz et al. (2023). Models of blood flow in the heart have become relevant to predict various cardiovascular conditions, where motion states from imaging are used Bonini et al. (2022); Zingaro et al. (2023); García-Villalba et al. (2021) or even fully-coupled fluid-solid interaction (FSI) models are employed Nordsletten et al. (2011); McCormick et al. (2011). However, prescribed cavity motion reduces the model's ability to adapt under varying loads, and full FSI models are complex, computationally demanding, and difficult to constrain (uncertain boundary conditions and sparse patient data for reliable geometry reconstruction). The fluid-reduced-solid interaction (FrSI) method closes the gap between model complexity and efficiency. This is a data-informed model reduction approach, particularly suited for cardiac FSI Hirschvogel et al. (2024). The method combines physics- with projection-based model reduction techniques that

Marc Hirschvogel e-mail: marc.hirschvogel@polimi.it
MOX, Dipartimento di Matematica, Politecnico di Milano, Milan, Italy

leverage Proper Orthogonal Decomposition (POD) modes derived from imaging (or some high-fidelity model) to build a reduced-order model (ROM), combined with a structural model of the ventricular wall defined on a 2D manifold. In this contribution, we show the FrSI method's applicability to a complex, patient-specific left heart model, with a particular focus on monolithic solver implementations in FEniCSx Alnæs et al. (2015); Baratta et al. (2023). This method encompasses the implementation of an Arbitrary Lagrangian-Eulerian (ALE) fluid mechanics problem subject to non-local constraints (Galerkin ROM, 3D-0D coupling to lumped circulation models). The solver and preconditioning aspects of this model and other fluid dynamics problems under non-local boundary conditions have been introduced in Hirschvogel et al. (2025).

Methods

The fluid-reduced-solid interaction (FrSI) problem of a 3D left heart model (atrium, ventricle, aortic outflow tract) along with the underlying data sources is depicted in Fig. 3.1. In particular, domain and motion data are retrieved from time-resolved dynamic computed tomography (CT), which are subsequently mapped to a finite element mesh in order to generate a discrete space of modes using POD Rathinam and Petzold (2003). The model is further coupled to a closed-loop systemic, pulmonary, and coronary circulation system Hirschvogel et al. (2017); Arthurs et al. (2016) in order to provide physiologically meaningful cardiovascular loads to the 3D model.

Preprocessing of patient data

The FrSI approach relies on external data sourced either from some high-dimensional model or from patient-specific imaging data. Here, we build a patient-specific model of the left heart by segmenting a dynamic cardiac CT data set using 3D Slicer Kikinis et al. (2014), cf. Fig. 3.1A. Subsequently, a diastolic frame is meshed with SimModeler Simmetrix Inc. (2023), and a motion tracking algorithm is used to extract the wall velocities at each frame and map them to the finite element mesh (with velocity degree of freedom space of size n_v). Thereafter, the wall velocity data for $m = 19$ frames is collected into a snapshot matrix $\hat{\mathbf{S}} \in \mathbb{R}^{n_v \times m}$, and the eigenvalue problem

$$(\hat{\mathbf{S}}^{\mathrm{T}}\hat{\mathbf{S}})\,\boldsymbol{\psi}_i = \lambda_i\,\boldsymbol{\psi}_i, \quad i = 1, \ldots, m, \tag{3.1}$$

is solved, with eigenvalues λ and eigenvectors $\boldsymbol{\psi} \in \mathbb{R}^m$. The first r_v POD modes $\boldsymbol{\phi} \in \mathbb{R}^{n_v}$ then can be computed as follows:

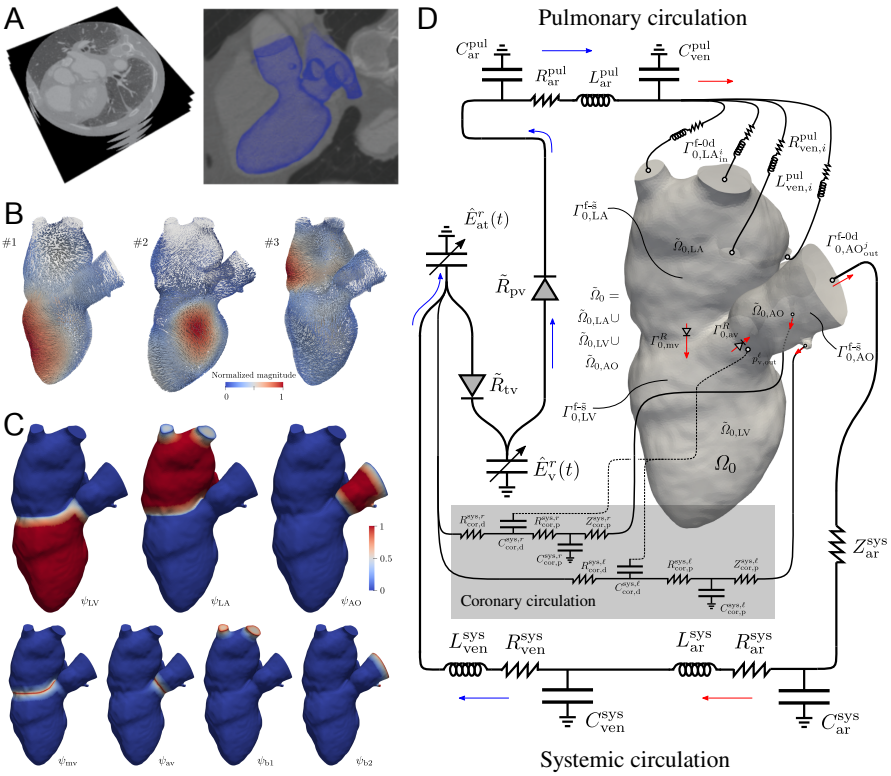


Fig. 3.1: **A.** Dynamic cardiac computed tomography (CT) images with contrast and dynamic segmentation of left heart lumen for subsequent finite element mesh generation, motion tracking of deformation over the heart cycle. **B.** Principal component analysis by means of Proper Orthogonal Decomposition (POD) of wall motion space. The first three most dominant POD modes are shown (10 are used). **C.** Partition of unity fields for regional decomposition of POD space: Atrium, ventricle, and aortic outflow tract—as well as their junctions and truncations around the in-/outflows—can exhibit independent kinematics. **D.** 3D-0D coupled FrSI model of the left heart.

$$\boldsymbol{\phi}_j = \frac{1}{\sqrt{\lambda_j}} \hat{\mathbf{S}} \boldsymbol{\psi}_j, \quad j = 1, \ldots, r_v, \tag{3.2}$$

of which the first three are shown in Fig. 3.1B. A suitable Galerkin model reduction operator,

$$\mathbf{V}_v^{\Gamma} \in \mathbb{R}^{n_v \times (r_v + n_v^{\Omega})}, \tag{3.3}$$

then needs to be defined, with n_v^{Ω} as the size of the space of bulk (non-boundary) velocities. In Eq. (3.3), POD modes Eq. (3.2) have to be incorporated such that

velocity degrees of freedom on $\Gamma_0^{\text{f-s}}$ are confined to the lower-dimensional subspace, but those associated to the bulk domain remain unconstrained Hirschvogel et al. (2024). Furthermore, the POD space is decomposed with a partition of unity approach such that each region can exhibit independent kinematics, cf. Fig. 3.1C.

Strong form problem statement

Here, we briefly state the strong problem of FrSI—fluid dynamics in an ALE reference frame Donea et al. (1982); Duarte et al. (2004) with a reduced structural wall model—subject to non-local flux-dependent tractions at the in- and outflows. The boundary subspace projection is then performed on the discrete system presented in a later section.

The incompressible non-conservative ALE Navier-Stokes equations, defining the conservation of linear momentum and mass over the domain Ω is written as:

$$\rho\left(\left.\frac{\partial \boldsymbol{v}}{\partial t}\right|_{\boldsymbol{x}_0} + \nabla\boldsymbol{v}(\boldsymbol{v}-\boldsymbol{w})\right) = \nabla\cdot\boldsymbol{\sigma} \qquad \text{in } \Omega\times[0,T], \tag{3.4}$$

$$\nabla\cdot\boldsymbol{v} = 0 \qquad \text{in } \Omega\times[0,T], \tag{3.5}$$

where ∇ is the gradient operator with respect to physical space ($\nabla\boldsymbol{v} := \frac{\partial v_i}{\partial x_j}\boldsymbol{e}_i\otimes\boldsymbol{e}_j$) and $\left.\frac{\partial(\bullet)}{\partial t}\right|_{\boldsymbol{x}_0}$ the time derivative in the ALE frame. Further, $\boldsymbol{v}$ and p are the fluid's velocity and pressure, $\boldsymbol{w}$ is the ALE domain velocity, and $\rho = 1.025\cdot10^{-6}\ \frac{\text{kg}}{\text{mm}^3}$ the blood density. The ventricular blood is assumed to be a Newtonian fluid, making the Cauchy stress $\boldsymbol{\sigma} = -p\boldsymbol{I} + \mu\left(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^{\text{T}}\right)$, with the dynamic viscosity $\mu = 4\cdot10^{-6}$ kPa $\cdot$ s. The fluid's boundary wall $\Gamma_0^{\text{f-s}}$ is assumed deformable and is described by a reduced solid mechanics model governed by the balance of linear momentum of finite strain elastodynamics, which entails the physics component of the FrSI method Hirschvogel et al. (2024). Since the displacement field at the boundary can be entirely derived from the fluid's velocity field, kinematic compatibility and continuity of tractions are readily fulfilled by incorporating the following boundary traction, cf. comparable derivations for small strain solid wall models Colciago et al. (2014):

$$\boldsymbol{t}_0^{\text{f-s}} = -h_0\left(\rho_{0,\text{s}}\frac{\partial\boldsymbol{v}}{\partial t} - \tilde{\nabla}_0\cdot\tilde{\boldsymbol{P}}\right) \qquad \text{on } \Gamma_0^{\text{f-s}}\times[0,T], \tag{3.6}$$

where $\tilde{\nabla}_0$ is a Nabla operator with respect to the reference frame (considering only in-plane derivatives), h_0 a wall thickness parameter (here 10 mm for ventricle, 5 mm for atrium, and 1 mm for aortic arch), $\rho_{0,\text{s}} = 10^{-6}\ \frac{\text{kg}}{\text{mm}^3}$ the reduced solid's density, and $\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P}}(\boldsymbol{u}_{\text{f}}(\boldsymbol{v})+\boldsymbol{u}_{\text{pre}},\boldsymbol{v})$ the first Piola-Kirchhoff stress—a general function of the fluid velocity and the fluid displacement at the boundary:

$$\boldsymbol{u}_{\mathrm{f}}(\boldsymbol{v}) = \int_0^t \boldsymbol{v}\,\mathrm{d}\bar{t}. \tag{3.7}$$

The first Piola-Kirchhoff stress is mapped from its material counterpart, the second Piola-Kirchhoff stress, $\tilde{\boldsymbol{P}} = (\boldsymbol{F}_{\mathrm{f}} - \boldsymbol{F}_{\mathrm{f}}\,\boldsymbol{n}_0 \otimes \boldsymbol{n}_0)\tilde{\boldsymbol{S}}$, with the fluid deformation gradient $\boldsymbol{F}_{\mathrm{f}} = \boldsymbol{I} + \nabla_0 \boldsymbol{u}_{\mathrm{f}}$, using Eq. (3.7). By eliminating its normal components and redefining the out-of-plane stretch on assumptions of incompressibility, a membrane right Cauchy-Green tensor, $\tilde{\boldsymbol{C}}$, for the surface as well as its time derivative can be defined. Finally, $\boldsymbol{u}_{\mathrm{pre}}$ is a prestress displacement computed by methods described in Schein and Gee (2021); Gee et al. (2010). More details on the kinematics and prestress for FrSI can be found in Hirschvogel et al. (2024). The constitutive equation for the reduced solid second Piola-Kirchhoff stress is

$$\tilde{\boldsymbol{S}} = 2\frac{\partial \Psi(\tilde{\boldsymbol{C}})}{\partial \tilde{\boldsymbol{C}}} + 2\frac{\partial \Psi_{\mathrm{v}}(\dot{\tilde{\boldsymbol{C}}})}{\partial \dot{\tilde{\boldsymbol{C}}}} + \tau_{\mathrm{a}}(t)\boldsymbol{A}_0, \tag{3.8}$$

with a structural tensor $\boldsymbol{A}_0 = \boldsymbol{I}$ for the atrium (isotropic active stress), $\boldsymbol{A}_0 = \tilde{\boldsymbol{M}}_0$ for the ventricle (active stress in directions of a reduced structural tensor Hirschvogel et al. (2024)), or $\boldsymbol{A}_0 = \boldsymbol{0}$ for the aortic arch (no active stress). The active stress $\tau_{\mathrm{a}}(t)$ follows the solution of an evolution equation, cf. Hirschvogel et al. (2017). The passive elastic model is of isotropic-exponential type[1] Demiray (1972), and a typical viscous pseudo-potential is used Chapelle et al. (2012).

The coupling to the circulatory system is expressed via $n_{0\mathrm{d}}^{\mathrm{b}} = 7$ non-local constraints, enforcing consistency between the flux over the 3D-0D boundary $\Gamma_i^{\text{f-0d}}$ and the flux variable $q_i^{0\mathrm{d}}$ from the 0D model:

$$\int_{\Gamma_i^{\text{f-0d}}} (\boldsymbol{v} - \widehat{\boldsymbol{w}}) \cdot \boldsymbol{n}\,\mathrm{d}A = \alpha_i q_i^{0\mathrm{d}}(\{\Lambda\}_{n_{0\mathrm{d}}^{\mathrm{b}}}) \qquad \text{in } [0,T]. \tag{3.9}$$

Therein, $\boldsymbol{n}$ is a unit outward normal of the current frame, and scaling parameters, α_i, account for the directionality of flow, i.e. should take the value of -1 if a 0D flux variable is imposed as an inflow to the fluid domain, and 1 otherwise. The multipliers $\{\Lambda\}_{n_{0\mathrm{d}}^{\mathrm{b}}}$ impose normal tractions (pressure loads) on their respective in-/outflow boundaries:

$$\boldsymbol{t}_i^{\text{f-0d}} = -\Lambda_i\,\boldsymbol{n} \qquad \text{on } \Gamma_i^{\text{f-0d}} \times [0,T]. \tag{3.10}$$

[1] While the myocardium typically exhibits highly anisotropic passive properties Holzapfel and Ogden (2009), it remains inconclusive how its transmurally varying fiber, sheet, and sheet-normal architecture—governing its anisotropic stiffness—can be consistently homogenized throughout the wall and mapped to a 2D surface representation. Since our focus primarily addresses adaptive fluid motion, we prefer an isotropic 2D model whose parameters easily can be calibrated to observed (diastolic) pressure-volume data.

The deformability of the fluid domain here is described by a pseudo-solid's displacement field $\boldsymbol{d}$ governed by

$$\nabla_0 \cdot \boldsymbol{\sigma}_\mathrm{g} = \boldsymbol{0} \qquad \text{in } \Omega_0 \times [0,T], \tag{3.11}$$

$$\boldsymbol{d} = \boldsymbol{u}_\mathrm{f}(\boldsymbol{v}) \qquad \text{on } \Gamma_0^{\text{f-}\tilde{\text{s}}} \times [0,T], \tag{3.12}$$

subject to the essential boundary condition on $\Gamma_0^{\text{f-}\tilde{\text{s}}}$, requiring $\boldsymbol{d}$ to take the value of the fluid displacement Eq. (3.7). Here, we use a fully nonlinear ALE model of coupled Neo-Hookean type Holzapfel (2000), which, on the discrete space, is scaled by the inverse of the reference cell's Jacobian determinant Shamanskiy and Simeon (2021). This scaling allows allocating stiffness to more anisotropic boundary elements and have the more regularly shaped bulk elements bear most of the deformation. The ALE deformation gradient and its determinant—as well as the grid/ALE convective velocity in Eq. (3.4)—are given by

$$\widehat{\boldsymbol{F}} = \boldsymbol{I} + \nabla_0 \boldsymbol{d}, \quad \widehat{J} = \det \widehat{\boldsymbol{F}}, \quad \text{and} \quad \widehat{\boldsymbol{w}} = \frac{\partial \boldsymbol{d}}{\partial t}. \tag{3.13}$$

Weak form and linearization

In the following, we define the continuous weak forms of the strong problem statements suitable for a monolithic finite element implementation in FEniCSx. For this purpose, all integrals are formulated over the respective reference domains, and all gradient operators relate to the undeformed configuration Ω_0. The general weak problem can be stated as follows:

Find fluid velocity $\boldsymbol{v}$, pressure p, multiplier variables $\{\Lambda\}_{n_{0\mathrm{d}}^\mathrm{b}}$, and ALE domain displacements $\boldsymbol{d}$ such that conservation of linear momentum,

$$\begin{aligned} R_{\delta v}\left(\boldsymbol{v}, p, \{\Lambda\}_{n_{0\mathrm{d}}^\mathrm{b}}, \boldsymbol{d}; \delta\boldsymbol{v}\right) &:= \int_{\Omega_0} \widehat{J}\rho \left(\left.\frac{\partial \boldsymbol{v}}{\partial t}\right|_{\boldsymbol{x}_0} + \left(\nabla_0 \boldsymbol{v}\, \widehat{\boldsymbol{F}}^{-1}\right)\, (\boldsymbol{v} - \widehat{\boldsymbol{w}}) \right) \cdot \delta\boldsymbol{v}\, \mathrm{d}V_0 \\ &+ \int_{\Omega_0} \widehat{J}\, \boldsymbol{\sigma}(\boldsymbol{v}, p, \boldsymbol{d}) : \nabla_0 \delta\boldsymbol{v}\, \widehat{\boldsymbol{F}}^{-1}\, \mathrm{d}V_0 + \sum_{i=1}^{n_{0\mathrm{d}}^\mathrm{b}} \Lambda_i \int_{\Gamma_{0,i}^{\text{f-0d}}} \widehat{J}\widehat{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{n}_0 \cdot \delta\boldsymbol{v}\, \mathrm{d}A_0 \\ &+ \int_{\Gamma_0^{\text{f-}\tilde{\text{s}}}} h_0 \left(\rho_{0,\mathrm{s}} \frac{\partial \boldsymbol{v}}{\partial t} \cdot \delta\boldsymbol{v} + \tilde{\boldsymbol{P}}\left(\boldsymbol{u}_\mathrm{f}(\boldsymbol{v}) + \boldsymbol{u}_\mathrm{pre}, \boldsymbol{v}\right) : \tilde{\nabla}_0 \delta\boldsymbol{v} \right) \mathrm{d}A_0 \\ &+ R_{\delta v}^R(\boldsymbol{v}, \boldsymbol{d}; \delta\boldsymbol{v}) + S_{\delta v}^\mathrm{D}(\boldsymbol{v}, p, \boldsymbol{d}; \delta\boldsymbol{v}) + S_{\delta v}^\mathrm{out}(\boldsymbol{v}, \boldsymbol{d}; \delta\boldsymbol{v}) = 0, \end{aligned} \tag{3.14}$$

conservation of mass,

$$R_{\delta p}(\boldsymbol{v}, \boldsymbol{d}; \delta p) := \int_{\tilde{\Omega}_0} \widehat{J} \nabla_0 \boldsymbol{v} : \widehat{\boldsymbol{F}}^{-\mathrm{T}} \delta p \, \mathrm{d}V_0 + S^{\mathrm{D}}_{\delta p}(\boldsymbol{v}, p, \boldsymbol{d}; \delta p) = 0, \tag{3.15}$$

constraints enforcing consistency between 0D and 3D models,

$$\begin{aligned} &R_{\delta \Lambda}\left(\boldsymbol{v}, \{\Lambda\}_{n^{\mathrm{b}}_{0\mathrm{d}}}, \boldsymbol{d}; \{\delta\Lambda\}_{n^{\mathrm{b}}_{0\mathrm{d}}}\right) := \\ &\sum_{i=1}^{n^{\mathrm{b}}_{0\mathrm{d}}} \left(\int_{\Gamma^{\text{f-0d}}_{0,i}} (\boldsymbol{v} - \widehat{\boldsymbol{w}}) \cdot \widehat{J}\widehat{\boldsymbol{F}}^{-\mathrm{T}} \boldsymbol{n}_0 \, \mathrm{d}A_0 - \alpha_i \, q^{0\mathrm{d}}_i \left(\{\Lambda\}_{n^{\mathrm{b}}_{0\mathrm{d}}} \right) \right) \delta\Lambda_i = 0, \end{aligned} \tag{3.16}$$

as well as ALE domain motion,

$$R_{\delta d}(\boldsymbol{d}, \boldsymbol{v}; \delta \boldsymbol{d}) := \int_{\Omega_0} \boldsymbol{\sigma}_{\mathrm{g}}(\boldsymbol{d}) : \nabla_0 \delta \boldsymbol{d} \, \mathrm{d}V_0 = 0, \tag{3.17}$$

hold true, for all fluid velocity and pressure test functions $(\delta\boldsymbol{v}, \delta p)$, ALE domain motion test functions $(\delta\boldsymbol{d})$, as well as multiplier test functions $(\{\delta\Lambda\}_{n^{\mathrm{b}}_{0\mathrm{d}}})$. The ALE problem is further subject to the essential boundary condition Eq. (3.12) at the deformable interface where the reduced solid is defined. The constitutive equation for the Cauchy stress is written with respect to the reference frame, $\boldsymbol{\sigma}(\boldsymbol{v}, p, \boldsymbol{d}) = -p\boldsymbol{I} + \mu \left(\nabla_0 \boldsymbol{v} \, \widehat{\boldsymbol{F}}^{-1} + \widehat{\boldsymbol{F}}^{-\mathrm{T}} (\nabla_0 \boldsymbol{v})^{\mathrm{T}} \right)$. In Eq. (3.14), $R^{R}_{\delta v}(\boldsymbol{v}, \boldsymbol{d}; \delta \boldsymbol{v})$ is a Robin term used to impose pressure jump-dependent tractions at the mitral and aortic valve planes. This represents a particular challenge, since effects of the mitral and aortic valves need pressure discontinuities across the interfaces of atrium and ventricle as well as ventricle and aortic root. For this purpose, very recently introduced *mixed-dimensional* functionality of FEniCSx is leveraged. This allows to create sub-discretizations (of equal or lower dimension) and make use of functions defined on different but related meshes within one finite element form. Here, the function space for the fluid pressure is defined on each sub-mesh (atrium, ventricle, aorta), and hence is allowed to jump across their respective interfaces. More details on the valve models can be found in Hirschvogel et al. (2025). Furthermore, the terms $S^{\mathrm{D}}_{\delta v}(\boldsymbol{v}, p, \boldsymbol{d}; \delta \boldsymbol{v})$ in Eq. (3.14) and $S^{\mathrm{D}}_{\delta p}(\boldsymbol{v}, p, \boldsymbol{d}; \delta p)$ in Eq. (3.15) refer to stabilization operators suitable for first-order approximations of both fluid velocity and pressure. Here, we make use of a variant of the G2 stabilization method Johnson (1998); Hoffman and Johnson (2003); Hessenthaler et al. (2017). Furthermore, to prevent backflow-induced divergence, all Neumann/3D-0D coupling boundaries are subject to an outflow stabilization Moghadam et al. (2011), referred to by the term $S^{\mathrm{out}}_{\delta v}(\boldsymbol{v}, \boldsymbol{d}; \delta \boldsymbol{v})$ in Eq. (3.14).

Flux variables $q^{0\mathrm{d}}_j = \mathbf{y} \cdot \mathbf{e}_j$ in Eq. (3.16) are, in general, solutions to a set of $n^{\mathrm{e}}_{0\mathrm{d}}$ 0D algebraic and first-order ordinary differential equations in time, with the vector of state variables $\mathbf{y}$ and $\mathbf{e}_j$ as the j-th $n^{\mathrm{e}}_{0\mathrm{d}}$-dimensional unit vector. We may state the 0D problem as follows: Find 0D model variables $\mathbf{y}$ such that

$$R_{0\mathrm{d}}\left(\mathbf{y}, \{\tilde{}\}_{n_{0\mathrm{d}}^{\mathrm{b}}}; \delta\mathbf{y}\right) := \left(\dot{\mathbf{g}}(\mathbf{y}, \{\tilde{}\}_{n_{0\mathrm{d}}^{\mathrm{b}}}) + \mathbf{f}(\mathbf{y}, \{\tilde{}\}_{n_{0\mathrm{d}}^{\mathrm{b}}})\right) \cdot \delta\mathbf{y} = 0, \tag{3.18}$$

for all $\delta\mathbf{y}$, where $\mathbf{g}$ is a linear ("left-hand side") and $\mathbf{f}$ a possibly nonlinear ("right-hand side") function in the variable vector $\mathbf{y}$ and/or multipliers $\{\Lambda\}_{n_{0\mathrm{d}}^{\mathrm{b}}}$.

The linearizations of the weak forms Eq. (3.14)—(3.17), being the derivatives in the direction of the velocity, pressure, multiplier, and domain displacement trial functions $\Delta\boldsymbol{v}$, Δp, $\{\Delta\Lambda\}_{n_{0\mathrm{d}}^{\mathrm{b}}}$, and $\Delta\boldsymbol{d}$, respectively,

$$K_{\delta(\cdot)_i\Delta(\cdot)_j} := D_{\Delta(\bullet)_j}\left[R_{\delta(\cdot)_i}\right], \tag{3.19}$$

are computed using symbolic automatic differentiation in FEniCSx, where $D_{\Delta(\bullet)_j}$ is the Gâteaux operator with respect to the trial function $\Delta(\bullet)_j$. Due to the Dirichlet conditions on the ALE problem, a special consideration is needed for the derivative of the ALE residual with respect to the fluid velocity. Due to the nature of how Dirichlet conditions are applied in FEniCSx, this is taken care of post-discretization.

Discretization and solution

The problem is discretized with finite elements of piecewise linear Lagrange polynomials in space and a single-step implicit finite difference scheme in time (One-step-θ scheme, with $\theta \in]0;1]$). The projection-based component of the FrSI method requires the reduced solid boundary to be projected to a lower-dimensional subspace spanned by POD modes, cf. Fig. 3.1B depicting the first three modes of this space. This is done by the boundary Galerkin projection operator Eq. (3.3). At the discrete assembled stage, at the current time-step indexed by $n+1$, we seek to find the discrete velocity $\mathbf{v}_{n+1}$, pressure $\mathbf{p}_{n+1}$, 3D-0D coupling multipliers $\tilde{}_{n+1}$, and domain displacements $\mathbf{d}_{n+1}$ satisfying

$$\mathbf{r}_{n+1} = \begin{bmatrix} \grave{\mathbf{V}}_v^{\mathrm{T}}\mathbf{r}_v(\grave{\mathbf{V}}_v\tilde{\mathbf{v}}, \mathbf{p}, \tilde{}, \mathbf{d}) \\ \mathbf{r}_p(\mathbf{p}, \grave{\mathbf{V}}_v\tilde{\mathbf{v}}, \mathbf{d}) \\ \mathbf{r}_{\tilde{}}(\tilde{}, \grave{\mathbf{V}}_v\tilde{\mathbf{v}}, \mathbf{d}) \\ \mathbf{r}_d(\mathbf{d}, \grave{\mathbf{V}}_v\tilde{\mathbf{v}}) \end{bmatrix}_{n+1} = \mathbf{0}, \tag{3.20}$$

where $\mathbf{r}_v$, $\mathbf{r}_p$, $\mathbf{r}_\Lambda$, and $\mathbf{r}_d$ are the assembled discrete counterparts of Eq. (3.14)–(3.17), respectively. The trial space projection is $\mathbf{v} = \grave{\mathbf{V}}_v\tilde{\mathbf{v}}$, where $\tilde{\mathbf{v}}$ is the (partly) reduced-dimensional velocity vector. In order to solve Eq. (3.20), a monolithic Newton scheme is employed, resulting in the linearized system of equations to solve for the variable increments in each nonlinear iteration indexed by $k+1$:

$$\begin{bmatrix} \mathbf{V}_v^{\Gamma\mathrm{T}}\mathbf{K}_{vv}\mathbf{V}_v^{\Gamma} & \mathbf{V}_v^{\grave{}\mathrm{T}}\mathbf{K}_{vp} & \mathbf{V}_v^{\Gamma\mathrm{T}}\mathbf{K}_{v\Lambda} & \mathbf{V}_v^{\grave{}\mathrm{T}}\mathbf{K}_{vd} \\ \mathbf{K}_{pv}\mathbf{V}_v^{\Gamma} & \mathbf{K}_{pp} & \mathbf{0} & \mathbf{K}_{pd} \\ \mathbf{K}_{\Lambda v}\mathbf{V}_v^{\Gamma} & \mathbf{0} & \mathbf{K}_{\Lambda\Lambda} & \mathbf{K}_{\Lambda d} \\ \mathbf{K}_{dv}\mathbf{V}_v^{\Gamma} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{dd} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta\tilde{\mathbf{v}} \\ \Delta\mathbf{p} \\ \Delta\tilde{} \\ \Delta\mathbf{d} \end{bmatrix}_{n+1}^{k+1} = - \begin{bmatrix} \mathbf{V}_v^{\grave{}\mathrm{T}}\mathbf{r}_v \\ \mathbf{r}_p \\ \mathbf{r}_{\Lambda} \\ \mathbf{r}_d \end{bmatrix}_{n+1}^{k}, \tag{3.21}$$

where sub-block matrices $\mathbf{K}_{ij}$ are obtained from the assembled discrete counterparts of Eq. (3.19), i.e. the derivatives of the residuals in the direction of the trial functions. Due to the lifting of Dirichlet conditions—ALE domain displacements prescribed to equal the fluid displacements on $\Gamma_0^{\text{f-s̃}}$, cf. Eq. (3.12)—special considerations have to be carried out to assemble $\mathbf{K}_{dv}$. This matrix yields

$$\mathbf{K}_{dv} = \gamma \left[(\mathbf{I} - \mathbf{I}_f)\mathbf{K}_{dd}\mathbf{I}_f - \mathbf{I}_f \right], \tag{3.22}$$

where γ is a time-integration factor stemming from the derivative of the fluid displacement with respect to the velocity, and $\mathbf{I}_f$ is a rank-deficient identity matrix with entries only at indices relating to boundary degrees of freedom of $\Gamma_0^{\text{f-s̃}}$.

Within one global Newton iteration, prior to solving Eq. (3.21), nonlinear sub-iterations (indexed by l) are carried out to find an equilibrium 0D flux (given the current nonlinear iterate $\tilde{}_{n+1}^{k}$) solving the time-discrete version of Eq. (3.18), meaning repeated solves of the linearized 0D model system,

$$\mathbf{K}_{n+1}^{0\mathrm{d},k,l}\Delta\mathbf{y}_{n+1}^{k,l+1} = -\mathbf{r}_{n+1}^{0\mathrm{d},k,l}, \tag{3.23}$$

followed by the solution of Eq. (3.21). In Eq. (3.23), $\mathbf{K}^{0\mathrm{d}}$ is computed with symbolic differentiation using SymPy Meurer et al. (2017).

Results

Figure 3.2 shows the results of a full heart cycle simulation. The physical time of the simulation is $T = 1$ s, and a time step size of $\Delta t = 0.00125$ s was used, hence $N = 800$ time steps were done. The mid-point single step time-integration method was set to Backward Euler, hence $\theta = 1$. The 3D computational domain consists of 265 722 nodes (1 487 039 finite elements), and the overall problem size is 1 790 303 degrees of freedom. The resulting linear system Eq. (3.21) was solved with a FGMRES Saad (1993) algorithm, preconditioned by our recently proposed BGS-S3×3 preconditioner Hirschvogel et al. (2025). All methods are implemented in the open-source FEniCSx- Baratta et al. (2023) and PETSc-based Balay et al. (2022) solver Ambit Hirschvogel (2024).

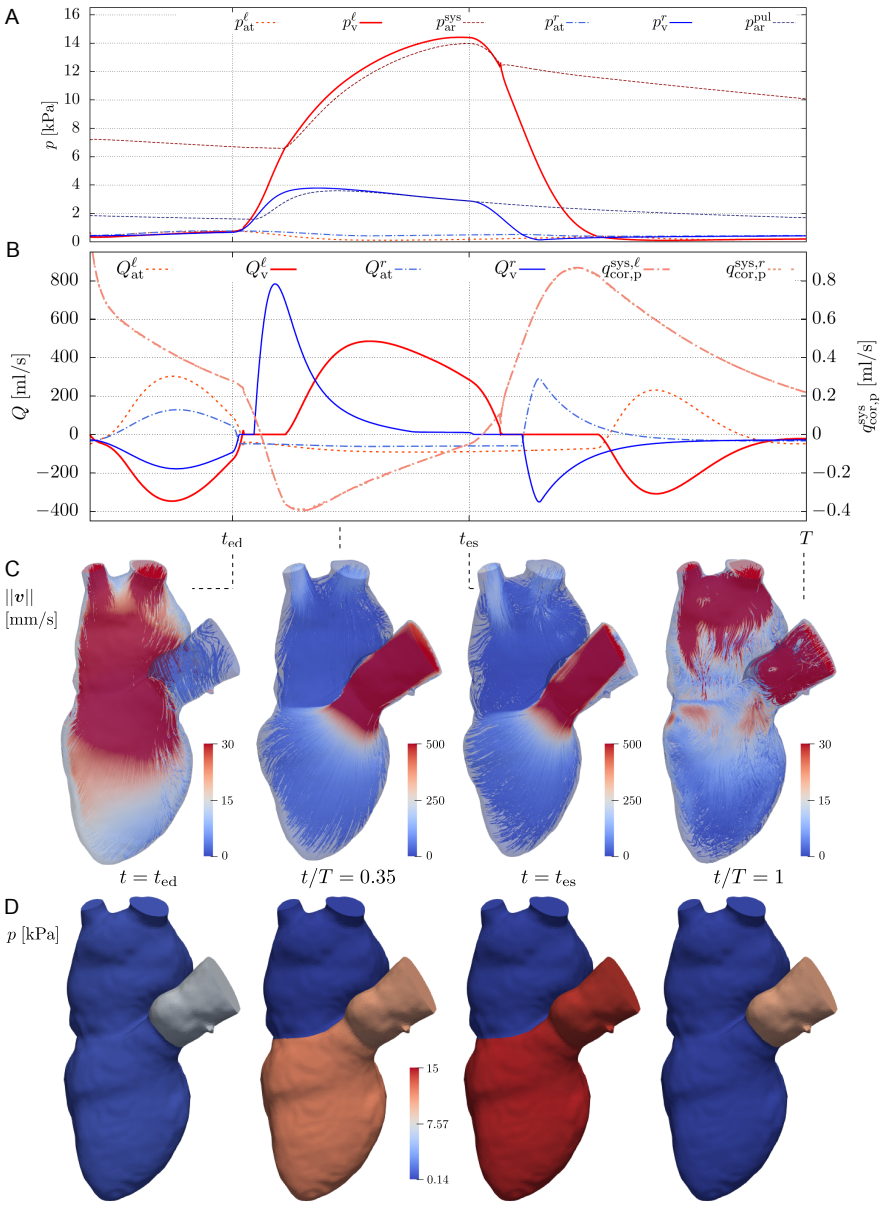


Fig. 3.2: Adapted from Hirschvogel et al. (2025): **A.** Left atrial, left ventricular, systemic arterial, right atrial, right ventricular, and pulmonary arterial pressures over time. **B.** Left atrial, left ventricular, right atrial, right ventricular, left and right proximal coronary fluxes over time. **C.** Magnitude of fluid velocity $\boldsymbol{v}$, streamlines on longitudinal cut through deformed domain Ω. Note the different scales for diastolic and systolic snapshots. **D.** Fluid pressure p, plotted on undeformed reference domain $\tilde{\Omega}_0$.

Conclusion

We presented the fluid-reduced-solid interaction (FrSI) method for a patient-specific, large-scale, left heart model, with a focus on a monolithic implementation in a FEniCSx software environment. The model can represent physiologic quantities throughout a heart cycle, and may be used to predict hemodynamics under varying cardiovascular conditions, e.g. for mitral valve regurgitation and repair.

Software and data availability

The presented results all are computed using Ambit Hirschvogel (2024) release version 1.3, cf. `https://github.com/marchirschvogel/ambit`. All data needed to run the model—the Ambit code as well as a medium (rf1) and fine discretization (rf2, used for generating the results presented), as well as the Ambit input file—are published at `https://zenodo.org/records/14631793`. Running Ambit requires FEniCSx to be installed (installation instructions at `https://github.com/FEniCS/dolfinx`), specifically the dolfinx development version dating to Git hash `4392bc84f440d7418ec4491a4a827d50720cb7d7` (Nov 28, 2024). The model might as well run with newer dolfinx versions, however it is not tested.

Acknowledgements DN acknowledges funding from the Engineering and Physical Sciences Research Council Healthcare Technology Challenge Award (EP/R003866/1), and support from the Wellcome Trust EPSRC Centre of Excellence in Medical Engineering (WT 088641/Z/09/Z) and the NIHR Biomedical Research Centre at Guy's and St. Thomas' NHS Foundation Trust and KCL.

References

Alnæs MS, Blechta J, Hake JE, Johansson A, Kehlet B, Logg A, Richardson C, Ring J, Rognes ME, Wells GN (2015) The FEniCS project version 1.5. Archive of Numerical Software 3

Arthurs CJ, Lau KD, Asrress KN, Redwood SR, Figueroa CA (2016) A mathematical model of coronary blood flow control: simulation of patient-specific three-dimensional hemodynamics during exercise. Am J Physiol Heart Circ Physiol 310(9):H1242–H1258, doi:10.1152/ajpheart.00517.2015

Balay S, Abhyankar S, Adams MF, Benson S, Brown J, Brune P, Buschelman K, Constantinescu E, Dalcin L, Dener A, Eijkhout V, Gropp WD, Hapla V, Isaac T, Jolivet P, Karpeev D, Kaushik D, Knepley MG, Kong F, Kruger S, May DA, McInnes LC, Mills RT, Mitchell L, Munson T, Roman JE, Rupp K, Sanan P, Sarich J, Smith BF, Zampini S, Zhang H, Zhang H, Zhang J (2022) PETSc/TAO users manual. Tech. Rep. ANL-21/39 - Revision 3.17, Argonne National Laboratory

Baratta IA, Dean JP, Dokken JS, Habera M, Hale JS, Richardson CN, Rognes ME, Scroggs MW, Sime N, Wells GN (2023) DOLFINx: The next generation FEniCS problem solving environment. doi:10.5281/zenodo.10447666, URL `https://doi.org/10.5281/zenodo.10447666`

Bonini M, Hirschvogel M, Ahmed Y, Xu H, Young A, Tang PC, Nordsletten D (2022) Hemodynamic modeling for mitral regurgitation. The Journal of Heart and Lung Transplantation 41(4 (Supplement)):S218–S219, doi:10.1016/j.healun.2022.01.1685

Chapelle D, Tallec PL, Moireau P, Sorine M (2012) Energy-preserving muscle tissue model: formulation and compatible discretizations. Journal for Multiscale Computational Engineering 10(2):189–211, doi:10.1615/IntJMultCompEng.2011002360

Colciago CM, Deparis S, Quarteroni A (2014) Comparisons between reduced order models and full 3D models for fluid-structure interaction problems in haemodynamics. Journal of Computational and Applied Mathematics 265:120–138, doi:10.1016/j.cam.2013.09.049

Demiray H (1972) A note on the elasticity of soft biological tissues. Journal of Biomechanics 5(3):309–311, doi:10.1016/0021-9290(72)90047-4

Donea J, Giuliani S, Halleux J (1982) An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. Computer Methods in Applied Mechanics and Engineering 33(1–3):689–723, doi:10.1016/0045-7825(82)90128-1

Duarte F, Gormaz R, Natesan S (2004) Arbitrary Lagrangian-Eulerian method for Navier-Stokes equations with moving boundaries. Computer Methods in Applied Mechanics and Engineering 193(45–47):4819–4836, doi:10.1016/j.cma.2004.05.003

García-Villalba M, Rossini L, Gonzalo A, Vigneault D, Martinez-Legazpi P, Durán E, Flores O, Bermejo J, McVeigh E, Kahn AM, Álamo JC (2021) Demonstration of patient-specific simulations to assess left atrial appendage thrombogenesis risk. Front Physiol 12(596596), doi:10.3389/fphys.2021.596596

Gee MW, Förster C, Wall WA (2010) A computational strategy for prestressing patient-specific biomechanical problems under finite deformation. International Journal for Numerical Methods in Biomedical Engineering 26(1):52–72, doi:10.1002/cnm.1236

Hessenthaler A, Röhrle O, Nordsletten D (2017) Validation of a non-conforming monolithic fluid-structure interaction method using phase-contrast MRI. Int J Numer Method Biomed Eng 33(8):e2845, doi:10.1002/cnm.2845

Hirschvogel M (2024) Ambit – A FEniCS-based cardiovascular multi-physics solver. Journal of Open Source Software 9(93):5744, doi:10.21105/joss.05744

Hirschvogel M, Bassilious M, Jagschies L, Wildhirt SM, Gee MW (2017) A monolithic 3D-0D coupled closed-loop model of the heart and the vascular system: Experiment-based parameter estimation for patient-specific cardiac mechanics. Int J Numer Method Biomed Eng 33(8):e2842, doi:10.1002/cnm.2842

Hirschvogel M, Balmus M, Bonini M, Nordsletten D (2024) Fluid-reduced-solid interaction (FrSI): Physics- and projection-based model reduction for cardiovascular applications. Journal of Computational Physics 506:112921, doi:10.1016/j.jcp.2024.112921, URL https://www.sciencedirect.com/science/article/pii/S0021999124001700

Hirschvogel M, Bonini M, Balmus M, Nordsletten D (2025) Effective block preconditioners for fluid dynamics coupled to reduced models of a non-local nature. Computer Methods in Applied Mechanics and Engineering 435:117541, doi:10.1016/j.cma.2024.117541

Hoffman J, Johnson C (2003) Adaptive Finite Element Methods for Incompressible Fluid Flow, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 97–157. doi:10.1007/978-3-662-05189-4_3

Holzapfel GA (2000) Nonlinear Solid Mechanics – A Continuum Approach for Engineering. Wiley Press Chichester

Holzapfel GA, Ogden RW (2009) Constitutive modelling of passive myocardium: A structurally based framework for material characterization. Phil Trans R Soc A 367(1902):3445–3475, doi:10.1098/rsta.2009.0091

Johnson C (1998) Adaptive finite element methods for conservation laws, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 269–323. doi:10.1007/BFb0096354

Kikinis R, Pieper SD, Vosburgh KG (2014) 3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support, Springer New York, New York, NY, pp 277–289. doi:10.1007/978-1-4614-7657-3_19

McCormick M, Nordsletten D, Kay D, Smith N (2011) Modelling left ventricular function under assist device support. International Journal for Numerical Methods in Biomedical Engineering 27(7):1073–1095, doi:10.1002/cnm.1428

Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, Rathnayake T, Vig S, Granger BE, Muller RP, Bonazzi F, Gupta H, Vats S, Johansson F, Pedregosa F, Curry MJ, Terrel AR, Roučka v, Saboo A, Fernando I, Kulal S, Cimrman R, Scopatz A (2017) SymPy: symbolic computing in Python. PeerJ Computer Science 3:e103, doi:10.7717/peerj-cs.103, URL https://doi.org/10.7717/peerj-cs.103

Moghadam ME, Bazilevs Y, Hsia TY, Vignon-Clementel IE, Marsden AL, Modeling of Congenital Hearts Alliance (MOCHA) (2011) A comparison of outlet boundary treatments for prevention of backflow divergence with relevance to blood flow simulations. Computational Mechanics 48(3):277–291, doi:10.1007/s00466-011-0599-0

Nordsletten DA, McCormick M, Kilner PJ, Hunter P, Kay D, Smith NP (2011) Fluid-solid coupling for the investigation of diastolic and systolic human left ventricular function. International Journal for Numerical Methods in Biomedical Engineering 27(7):1017–1039, doi:10.1002/cnm.1405

Rathinam M, Petzold LR (2003) A new look at Proper Orthogonal Decomposition. SIAM Journal on Numerical Analysis 41(5):1893–1925, doi:10.1137/S0036142901389049

Saad Y (1993) A flexible inner-outer preconditioned GMRES algorithm. SIAM J Sci and Stat Comput 14(2):461–469, doi:10.1137/0914028

Schein A, Gee MW (2021) Greedy maximin distance sampling based model order reduction of prestressed and parametrized abdominal aortic aneurysms. Advanced Modeling and Simulation in Engineering Sciences 8(18), doi:10.1186/s40323-021-00203-7

Schwarz EL, Pegolotti L, Pfaller MR, Marsden AL (2023) Beyond CFD: Emerging methodologies for predictive simulation in cardiovascular health and disease. Biophys Rev (Melville) 4(1):011301, doi:10.1063/5.0109400

Shamanskiy A, Simeon B (2021) Mesh moving techniques in fluid-structure interaction: robustness, accumulated distortion and computational efficiency. Computational Mechanics 67:583–600, doi:10.1007/s00466-020-01950-x

Simmetrix Inc (2023) SimModeler (Version 2023.0) [Computer Software]. URL http://simmetrix.com

Zingaro A, Bucelli M, Fumagalli I, Dede' L, Quarteroni A (2023) Modeling isovolumetric phases in cardiac flows by an Augmented Resistive Immersed Implicit Surface method. International Journal for Numerical Methods in Engineering 39(12):e3767, doi:10.1002/cnm.3767

Glossary

Use the template *glossary.tex* together with the Springer Nature document class SVMono (monograph-type books) or SVMult (edited books) to style your glossary in the Springer Nature layout.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

glossary term Write here the description of the glossary term. Write here the description of the glossary term. Write here the description of the glossary term.

Index