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.

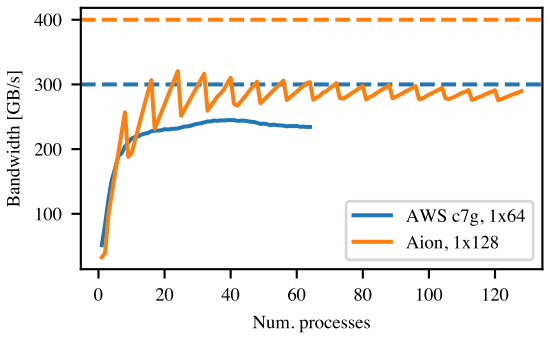


Fig. 2.1: Single-node STREAM benchmark. Theoretical peak bandwidth of each system show as dashed line.

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

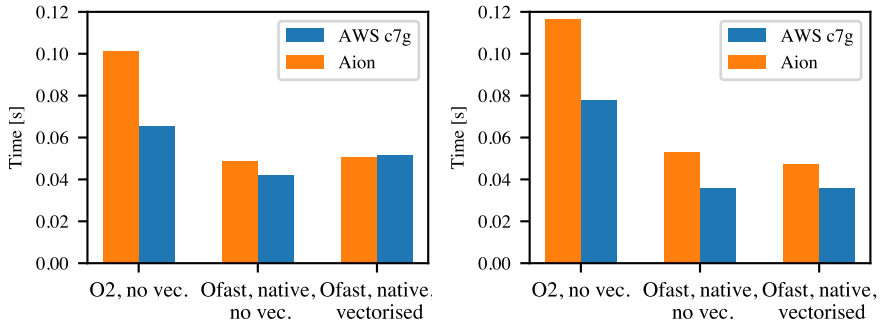
For the finite element kernel benchmarks we compiled the kernels with LLVM/clang 18.1.3 and GCC 13.2.0. Full details are given in Table 2.2.

	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.

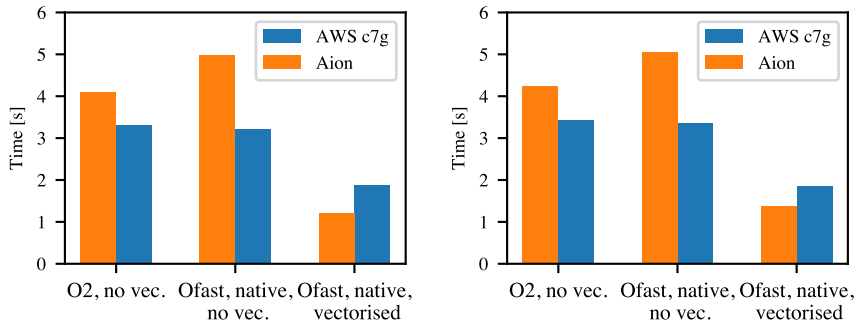
Results for kernel benchmarks are shown in Figure 2.2 and Figure 2.3. Low-order kernels (Figure 2.2) show no dependence on compiler vectorisation setup. On the other hand, AWS c7g shows 1.3x speed-up over Aion which we attribute to higher memory bandwidth for a single process.

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

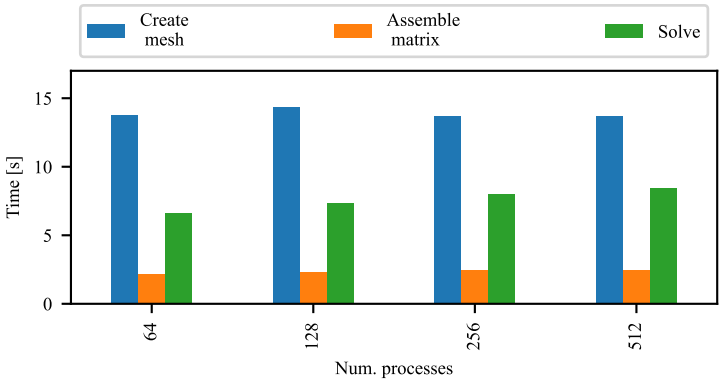
Benchmarks for memory bandwidth, local finite element kernels and parallel scalability of Poisson solver were executed on Aion nodes and on AWS c7g(n) instances.

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.

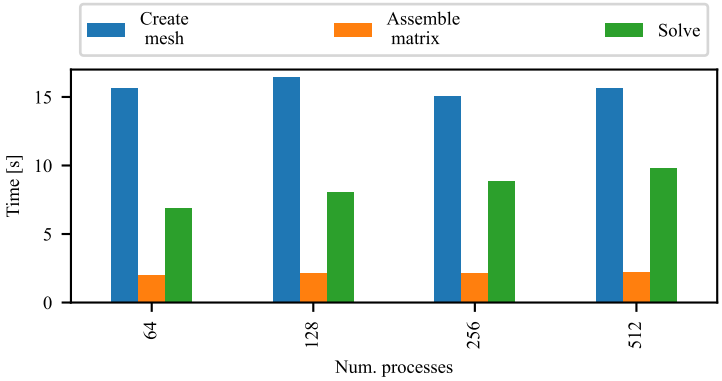
In terms of auto-vectorisation capabilities of GCC 13.2.0 and clang 18.1.3, both produced optimised instructions for the targeted microarchitectures (Zen 2 for Aion and Neoverse V1 for AWS c7g). This observation was confirmed with performance benchmarks based on local finite element kernels for the Laplace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

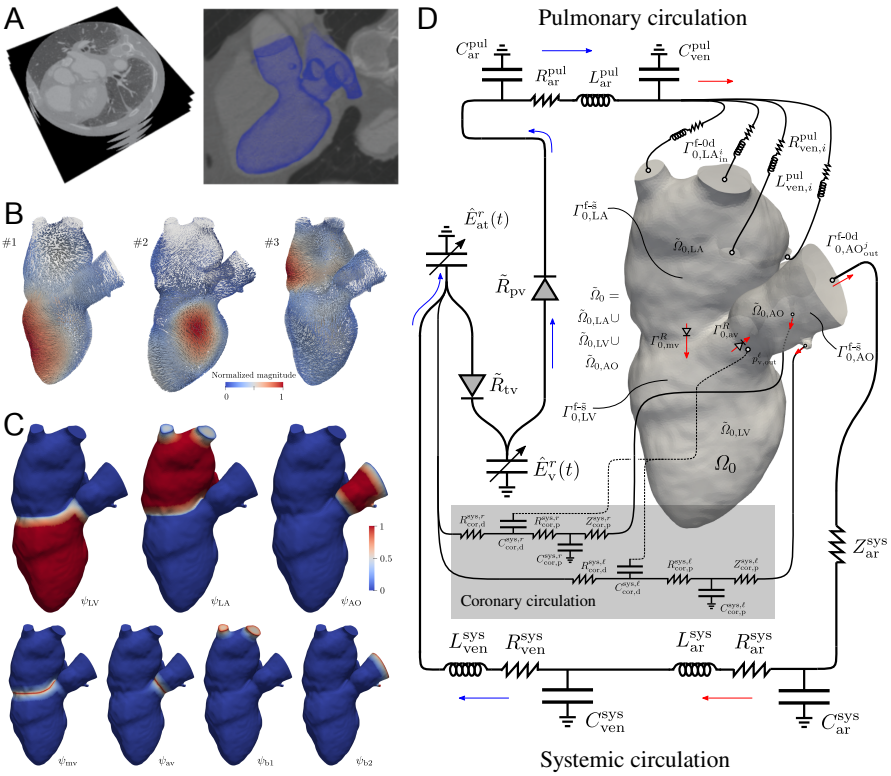


Fig. 3.1: **A.** Dynamic cardiac computed tomography (CT) images with contrast and dynamic segmentation of left heart lumen for subsequent finite element mesh generation, motion tracking of deformation over the heart cycle. **B.** Principal component analysis by means of Proper Orthogonal Decomposition (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,

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

then needs to be defined, with n_v^Q 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

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

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

subject to the essential boundary condition on $\Gamma_0^{\text{f-s̃}}$, 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,

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

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

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

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 $\boldsymbol{\Lambda}_{n+1}$, and domain displacements $\mathbf{d}_{n+1}$ satisfying

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

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 $\mathbf{v} = \mathbf{V}_v^{\boldsymbol{\Gamma}}\tilde{\mathbf{v}}$, 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 $k+1$:

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

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 $\Gamma_0^{\text{f-s̄}}$, 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 $\Gamma_0^{\text{f-s̄}}$.

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 $\mathbf{\Lambda}_{n+1}^{k}$) 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).

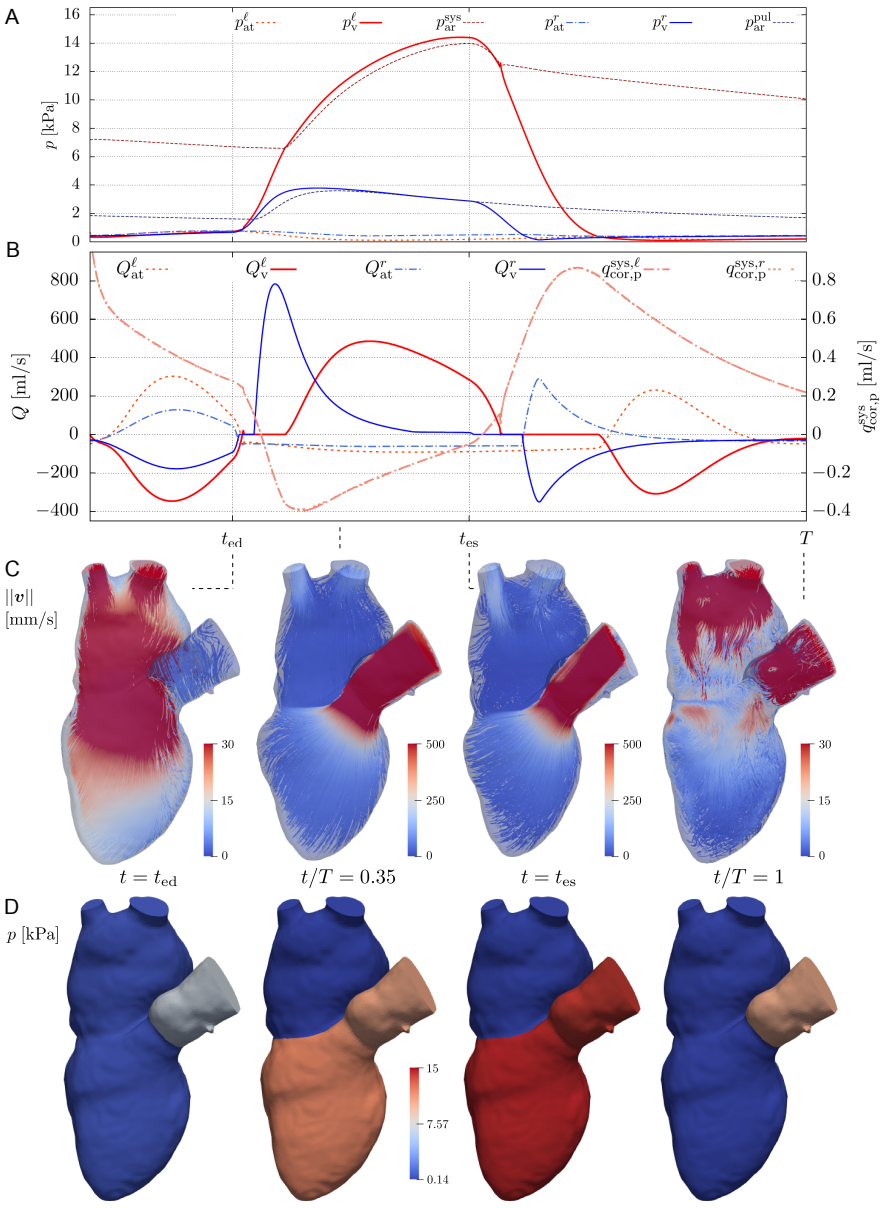


Fig. 3.2: Adapted from Hirschvogel et al. (2025): **A.** Left atrial, left ventricular, systemic arterial, right atrial, right ventricular, and pulmonary arterial pressures over time. **B.** Left atrial, left ventricular, right atrial, right ventricular, left and right proximal coronary fluxes over time. **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

Chapter 4
Growth and Remodelling Package in FEniCSx

Karl Munthe, Henrik Finsberg, Samuel Wall, Joakim Sundnes

Abstract The heart is a dynamic organ that changes its size and shape to regulate its behaviour to the demands of the body, which can change for example through body growth, exercise, or the onset of a disease. Different models have been proposed to try to capture various types of cardiac growth resulting from mechanical stimuli, but the models have rarely been compared systematically. In this manuscript we present a framework implemented in FEniCSx that allows one to quickly run simulations of growth and remodelling with different material models and different growth laws. We present and compare the growth predicted by each model for a set of simple experiments, and compare the results to the relevant literature in the field. All the code can be found at `https://github.com/karlfm/Growth-and-Remodeling-in-FEniCSx`

4.1 Introduction

In classical continuum mechanics one normally studies the mechanics of bodies where mass, linear momentum, angular momentum, and energy are conserved prop-

K. Munthe
Simula Research Laboratory, Oslo, Norway and
Department of Informatics, University of Oslo, Norway, e-mail: `karlfredrik@simula.no`

H. Finsberg
Simula Research Laboratory, Oslo, Norway

S. Wall
Simula Research Laboratory, Oslo, Norway

J. Sundnes
Simula Research Laboratory, Oslo, Norway and
Department of Informatics, University of Oslo, Norway

erties. This approach has been extremely successful and is the bedrock for traditional engineering disciplines, but it does not accurately capture aspects of how living organisms change with respect to their environment. One of the unique features of biological material is its ability to grow and evolve by adding or removing mass. Understanding how biological matter grows and what drives the growth is important, not only to understand normal growth and development, but also when and how growth may become non-compensatory and drive disease.

It has been known for a long time that the growth of organs, such as the heart, is regulated at least in part by the forces applied to it (Hsu, 1968). This understanding has led to the formulation of growth laws that link growth and remodeling to local stress or strain.

In this chapter we will introduce a package written in FEniCSx that allows one to easily model growth and remodelling of biological tissue, with the aim of quickly testing combinations of growth models and material models. Allowing researchers to systematically test combinations of material models and growth tensors will aid in the discovery of more accurate models of growth and remodelling phenomena of biological tissue.

4.2 Methods

4.2.1 Growth and Remodeling in Continuum Mechanics

Consider a solid body that is continuously and smoothly deforming from one configuration to another, and denote the initial configuration (also called the reference configuration) as $\mathcal{M}$ and the current configuration $\mathcal{N}$. We denote a point in $\mathcal{M}$ with uppercase letters $\mathbf{X} = (X, Y, Z)$ and a point in $\mathcal{N}$ with lowercase letters $\mathbf{x} = (x, y, z)$[1]. A point in $\mathcal{M}$ can be mapped to a point in $\mathcal{N}$ by a motion, $\phi(\mathbf{X}) : \mathbf{X} \rightarrow \mathbf{x}$, which is a diffeomorphism. We can map a vector from the reference configuration to the current configuration via the pushforward of ϕ, which is commonly denoted by $\mathbf{F}$ and is referred to as the deformation gradient, computed as $\mathbf{F} = \partial\mathbf{x}/\partial\mathbf{X}$. The displacement field $\mathbf{u} = \mathbf{x} - \mathbf{X}$ is a vector field describing the displacement of each point $\mathbf{X}$ in the reference configuration to its location $\mathbf{x}$ in the deformed configuration. Most mechanical models of the heart assume that the tissue is hyperelastic, meaning that deformations of the material conserve its energy, and when any load on the tissue is removed it returns to its original reference shape. Growth, on the other hand, represents a permanent change of the unloaded reference configuration, and cannot be modeled as an elastic deformation. Instead, it is commonly modeled using the framework of plastic or elasto-plastic deformations. The most common approach, introduced by (Rodriguez et al., 1994), is to multiplicatively split the deformation tensor into an elastic part and an inelastic growth part,

[1] Apart from the letters X, Y, and Z, all upper case letters represent tensors.

$$\mathbf{F} = \mathbf{F}_e \mathbf{F}_g, \tag{4.1}$$

where $\mathbf{F}_e$ is the elastic deformation and $\mathbf{F}_g$ is the plastic deformation that represents growth.

One interpretation of 4.1 is that the material first deforms by $\mathbf{F}_g$ in a way that does not cause stress, but might cause incompatibilities in the form of discontinuities or overlapping material. The deformation described by $\mathbf{F}_g$ leads to an unphysical intermediate configuration, which is then deformed by $\mathbf{F}_e$ in a way that removes the unphysical characteristics that occurred from $\mathbf{F}_g$, but adds residual stress. Sufficient conditions for the existence of intermediate configurations such as $\mathbf{F}_g$ are discussed in (Goodbrake et al., 2021). We assume that the deformation described by $\mathbf{F}_e$ is hyperelastic, such that we can obtain the first Piola–Kirchhoff stress tensor by differentiating a strain energy function,

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}_e}. \tag{4.2}$$

We typically describe the growth in terms of multiple growth steps, given by

$$\mathbf{F}_g^{i+1} = \mathbf{F}_g^i \mathbf{F}_g^{\text{inc}}.$$

Here, $\mathbf{F}_g^{\text{inc}}$ is the incremental growth tensor describing growth occuring in one step, and $\mathbf{F}_g^i$ is the cumulative growth after i steps. The initial growth tensor, $\mathbf{F}_g^0$, is set to the identity tensor. The cumulative growth deformation tensor after n steps is given by

$$\mathbf{F}_g^n = \mathbf{F}_g^{\text{inc}}|_{t=0} \mathbf{F}_g^{\text{inc}}|_{t=1} \cdots \mathbf{F}_g^{\text{inc}}|_{t=n},$$

where $\mathbf{F}_g^{\text{inc}}|_{t=i}$ means the $\mathbf{F}_g^{\text{inc}}$ at the i'th step. Note that the i'th incremental growth tensor is dependent on the stress or strain that occurred in the $i-1$'th growth step (see (Goriely and Amar, 2007)). It is common to assume that the growth tensor is diagonal, and to express the incremental growth tensor in terms of fiber, crossfiber, and normal directions

$$\mathbf{F}_g^{\text{inc}} = F_{g,f}^{\text{inc}} \mathbf{e}_f \otimes \mathbf{e}_f + F_{g,c}^{\text{inc}} \mathbf{e}_c \otimes \mathbf{e}_c + F_{g,n}^{\text{inc}} \mathbf{e}_n \otimes \mathbf{e}_n.$$

Here, $F_{g,i}^{\text{inc}}$ for $i = \{f, c, n\}$ are functions of either stress or strain, and $\mathbf{e}_i$ for $i = \{f, c, n\}$ are orthonormal basis vectors in the fiber, crossfiber, and normal directions respectively.

The incremental growth tensor depends on the local stress or strain, which is determined by solving for mechanical equilibrium at each growth step;

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_e \mathbf{F}_g && \text{in } \mathcal{M}, \\ \nabla \cdot \mathbf{P} &= 0 && \text{in } \mathcal{M}, \\ \mathbf{P} \cdot \nu &= 0 && \text{on } \partial\mathcal{M}_N, \\ \mathbf{u} &= g_D && \text{on } \partial\mathcal{M}_D, \end{aligned} \tag{4.3}$$

where ν is a surface normal vector, $\partial\mathcal{M}_N$ and $\partial\mathcal{M}_D$ denote the boundaries which are prescribed Neumann and Dirichlet boundary conditions respectively.

It remains to specify the growth laws that determine F_g and the strain energy Ψ in (4.2). We will define the growth laws in the next section, but for all the experiments we use a nearly incompressible neo-Hookean model, so (4.2) becomes

$$\mathbf{P} = \frac{\partial \Psi_{\text{iso}}}{\partial \mathbf{F}_e} + \frac{\partial \Psi_{\text{vol}}}{\partial \mathbf{F}_e},$$
$$\mathbf{P} = \frac{\partial}{\partial \mathbf{F}_e} \left[\frac{\mu}{2} \left(\text{tr}\bar{\mathbf{C}} - 3 \right) + \kappa (J-1)^2 \right],$$

where μ and κ material parameters. Ψ_{iso} and Ψ_{vol} are the isochoric (distortional) and volumetric (dilational) parts of the strain energy function. To decouple the energy stored in the body as a result of volume preserving deformation and non-volume preserving deformation, we introduce $\bar{\mathbf{F}}_e = \mathbf{F}_e J^{-1/3}$, whose determinant is equal to one. The isochoric right Cauchy-Green deformation tensor, $\bar{\mathbf{C}}$, is calculated as $\bar{\mathbf{C}} = \bar{\mathbf{F}}_e^\top \bar{\mathbf{F}}_e$. Now, $\partial\Psi_{\text{iso}}/\partial\mathbf{F}_e = 0$ only if the deformation preserves the shape, and $\partial\Psi_{\text{vol}}/\partial\mathbf{F}_e = 0$ only if the deformation preserves the volume (see Chapter 6 of (Holzapfel, 2002) for more details).

For further information about continuum mechanics we recommend (Marsden and Hughes, 1983) and (Holzapfel, 2002), and for further information about growth and remodelling we recommend (Goriely, 2017) and (Yavari, 2010).

4.2.2 Numerical Implementation

Algorithm 1 shows an overview of the steps involved in the solution of the growth model equations. For stress based growth, you would update the stress tensor rather than the strain tensor, and for additive growth laws, you would add the cumulative and incremental growth tensors instead of multiplying them together.

Constructing the weak form: We multiply $\nabla \cdot \mathbf{P}$ by a test function, which we set to be in the same function space as $\mathbf{u}$, and integrate over a discretization of $\mathcal{M}$. By applying integration by parts, we obtain

Algorithm 1: Growth tensor and stress/strain tensor are updated at each growth step. Both $\mathbf{F}_{\mathrm{e}}$ and $\mathbf{F}_g^{\mathrm{inc}}$ are dependent on $\mathbf{u}$.

for *each time step* **do**
    Solve (4.3) for the displacement $\mathbf{u}$;
    Update the stress/strain tensor using the obtained displacement $\mathbf{u}$.;
    Update the growth tensor using the stress/strain tensor from the previous line.;
end

$$\int_\Omega (\nabla \cdot \mathbf{P}) \cdot \eta d\mathbf{X} = 0$$
$$\int_\Omega \mathbf{P} : \nabla\eta d\mathbf{X} = \int_{\partial\Omega} \mathbf{P} \cdot \eta \cdot \nu dA. \tag{4.4}$$

Since we are using test functions, η, that vanish on $\partial_D \mathcal{M}$, and the normal component of $\mathbf{P}$ is zero on $\partial_N \mathcal{M}$ we can set the boundary integral to zero. (4.4) is solved using FEniCSx (Baratta et al., 30 December 2023).

Iteratively solving the conservation of momentum: We now solve (4.4) for the displacement $\mathbf{u}$, which we can use to compute all the necessary variables. We use tetrahedral, second order, continuous, Lagrange elements to approximate $\mathbf{u}$; and a first order, discontinuous, Lagrange elements to approximate $\mathbf{F}_e$ and $\mathbf{F}_g$. This is a common numerical scheme in cardiac mechanics which has been demonstrated to avoid locking (Oliveira and Sundnes, 2016).

4.2.3 Solving Growth Laws on the Unit Cube

In the simulations we have run we have aligned the x-axis with the fiber direction and the y-, and z-axis are the crossfiber and normal direction. To be consistent with the literature we will use $\mathbf{e}_f$, $\mathbf{e}_c$, and $\mathbf{e}_n$ to denote the unit vectors in the (x, y, z) directions respectively.

Boundary conditions: We set the following boundary conditions

$$g_D = \begin{cases} u &= \begin{cases} 0 & \text{on } x = 0, \\ u_D & \text{on } x = 1, \end{cases} \\ v &= 0 \quad \text{on } y = 0, \\ w &= 0 \quad \text{on } z = 0, \end{cases}$$

where u, v, and w are the displacement in the x, y, and z direction respectively. u_D specifies how much the body is displaced.

Numerical simulations: We ran two simulations, one with a 10% stretch and one with a 10% compression which corresponds to $u_D = 0.1$ and $u_D = -0.1$ respectively.

For the GCG model, $F_{g,c,\max}$ was set to 1.2 in the stretch simulation and 0.8 in the compression simulation (see table 4.1). We set $\mu = 15$ kPa and $\kappa = 100$ kPa.

4.2.4 The different growth models

In this paper we compare five growth models which we have taken from (Taber, 1998), (Kroon et al., 2009), (Göktepe et al., 2010), and (Kerckhoffs et al., 2012). The growth models are given in table 4.1 where LT2 is from (Taber, 1998), KFR is from (Kroon et al., 2009), GEG and GCG are from (Göktepe et al., 2010) and KOM is from (Kerckhoffs et al., 2012). Each growth model had a set point that either determined the homeostatic level of stress, stretch, or strain. When the stress, stretch, or strain reaches the set point, then growth will cease to occur. If this does not happen, the body will grow indefinitely, which we call runaway growth. We used the same variables as they were given in the original papers, except for the GCG, where we scaled the variables to more accurately fit with the shear modulus used here. The values are tabulated in table 4.2. In LT2, $\sigma_{p,0}$ and $\sigma_{a,0}$ are set points for

	$F^{i+1}_{g,f}$	$F^{i+1}_{g,n}$	$F^{i+1}_{g,c}$
LT2	$F^i_{g,f}\left(\frac{\sigma_{\theta p}-\sigma_{p,0}}{T\sigma_{p,0}}+1\right)$	$F^i_{g,n}\left(\frac{\sigma_{\theta a}-\sigma_{a,0}}{T\sigma_{a,0}}+1\right)$	1
KFR	$F^i_{g,f}(\beta(\sqrt{2E_{ff}+1}-1-s_{\text{hom}})+1)^{1/3}$	$F^i_{g,n}(\beta(\sqrt{2E_{ff}+1}-1-s_{\text{hom}})+1)^{1/3}$	$F^i_{g,c}(\beta(\sqrt{2E_{ff}+1}-1-s_{\text{hom}})+1)^{1/3}$
GEG	$\frac{1}{\tau}\left(\frac{F_{g,f,\max}-F^i_{g,f}}{F_{g,f,\max}-1}\right)^{\gamma}(F^i_{e,f}-\lambda^{\text{crit}})+F^i_{g,f}$	1	1
GCG	1	$\frac{1}{\tau}\left(\frac{F_{g,c,\max}-F^i_{g,c}}{F_{g,c,\max}-1}\right)^{\gamma}(\text{tr}\,(()\,\mathbf{M})-p^{\text{crit}})+F^i_{g,c}$	1
KOM	$\begin{cases} F^i_{g,f}k_{ff}\frac{f_{\text{ff,max}}\Delta t_{\text{growth}}}{1+\exp(-f_f(s_{\text{l}}-s_{l,50}))}+1, & s_{\text{l}}\geq 0 \\ F^i_{g,f}\frac{-f_{\text{ff,max}}\Delta t_{\text{growth}}}{1+\exp(f_f(s_{\text{l}}+s_{l,50}))}+1, & s_{\text{l}}<0 \end{cases}$	$\begin{cases} F^i_{g,c}\sqrt{k_{cc}\frac{f_{cc,\max}\Delta t_{\text{growth}}}{1+\exp(-c_{\text{f}}(s_{\text{t}}-s_{t,50}))}+1}, & s_{\text{t}}\geq 0 \\ F^i_{g,c}\sqrt{\frac{-f_{cc,\max}\Delta t_{\text{growth}}}{1+\exp(c_{\text{f}}(s_{\text{t}}+s_{t,50}))}+1}, & s_{\text{t}}<0 \end{cases}$	$\begin{cases} F^i_{g,c}\sqrt{k_{cc}\frac{f_{cc,\max}\Delta t_{\text{growth}}}{1+\exp(-c_{\text{f}}(s_{\text{t}}-s_{t,50}))}+1}, & s_{\text{t}}\geq 0 \\ F^i_{g,c}\sqrt{\frac{-f_{cc,\max}\Delta t_{\text{growth}}}{1+\exp(c_{\text{f}}(s_{\text{t}}+s_{t,50}))}+1}, & s_{\text{t}}<0 \end{cases}$

Table 4.1: $F_{g,f}$, $F_{g,c}$, $F_{g,n}$, for each of the five models. The parameters T, β, τ, and Δt, simply determine the rate of growth and can be tuned to match the growth rate of data obtained from experiments.

Model	Parameters
LT2	$\sigma_{a,0} = 30$ [kPa], $\sigma_{p,0} = 3$ [kPa], $T = 10^{-4}$
KFR	$s_{\text{hom}} = 0.13$, $\beta = 10^{-2}$
GEG	$F_{g,f,\text{max}} = 1.5$, $\lambda^{\text{crit}} = 1.01$, $\gamma = 2$, $\tau = 10^2$
GCG	$F_{g,c,\text{max}} = 1.2$ and 0.8, $p^{\text{crit}} = 0.12$, $\gamma = 2$, $\tau = 10^4$
KOM	$f_{\text{ff,max}} = 0.31$ [1/days], $f_f = 150$, $s_{l50} = 0.06$, $F_{ff50} = 1.35$, $f_{l,\text{slope}} = 40$, $f_{\text{ff,max}} = 0.1$ [1/days] $c_{\text{f}} = 75$, $s_{\text{t50}} = 0.07$, $F_{\text{cc50}} = 1.28$, $c_{\text{th,slope}} = 60$, $E_{ff,\text{set}} = 0$, $E_{\text{cross,set}} = 0$, $\Delta t = 10^{-2}$ [days]

Table 4.2: Model parameters for the growth models. T, β, τ, and Δt, determine the speed of growth.

the passive and active fiber stress at equilibrium, and $\sigma_{\theta p}$ and $\sigma_{\theta a}$ are the active and passive fiber stresses. In the simulations we have run, we have only used the passive component of σ, and have set $\sigma_a = 0$. For KFR, s_{hom} is the strain set point. For GEG and GCG, $F_{g,f,\text{max}}$ and $F_{g,c,\text{max}}$ is the maximum amount of growth allowed to occur. $\mathbf{M}$ is the Mandel stress, which is defined as

$$\mathbf{M} = \mathbf{F}^\top \mathbf{P}$$

and p^{crit} is the stress set point. λ^{crit} is the strain set point. For KOM, k_{ff} and k_{cc} are defined as

$$k_{ff} = \frac{1}{1 + \exp(f_{\text{length,slope}}(\mathbf{F}^i_{g,ff} - F_{ff,50}))}$$
$$k_{cc} = \frac{1}{1 + \exp(c_{\text{thickness,slope}}(\mathbf{F}^i_{g,cc} - F_{cc,50}))}$$

and s_{l}, and s_{t} are defined as

$$s_{\text{l}} = \max(E_{ff}) - E_{ff,\text{set}}$$
$$s_{\text{t}} = \min(E_{\text{cross, max}}) - E_{\text{cross,set}}$$

where E_{ij} is the Lagrange strain tensor, and E_{ff} is the strain in the fiber direction, and $E_{\text{cross, max}}$ is the maximum algebraic maximum principle strain of the matrix (see (Witzenburg and Holmes, 2018))

$$E_{\text{cross}} = \begin{pmatrix} E_{cc} & E_{cr} \\ E_{rc} & E_{rr} \end{pmatrix}$$

and $E_{ff,\text{set}}$ and $E_{\text{cross,set}}$ are set points.

Growth stops for the LT2 model when $\sigma_\theta = \sigma_0$. For KOM, since k_{cc} and k_{ff} are logistic functions, the growth is bounded from above and below inhibiting runaway growth. Finally, for KFR, it does not appear obvious that it will not obtain runaway growth, but other simulations setups that were tested did result in runaway growth, even though the one we present here does not.

4.3 Results

The data we collected from the simulations described in Section 4.2.3 were the stretch and growth that occurred in the middle of the cube. The results are depicted in figures 4.1 and 4.2. The top row of each figure displays the fiber and crossfiber components of the growth tensor, $\mathbf{F}_g$, and the bottom row displays the fiber and crossfiber components of the elastic deformation tensor $\mathbf{F}_e$. In the simulations we ran, $\mathbf{F}_e$ is diagonal, so the components of $\mathbf{F}_e$ are the principle stretches. This is because $\sqrt{\mathbf{e}_i^\top \mathbf{C}_e \mathbf{e}_i}$ is the principle stretch in the i'th direction, and $\sqrt{\mathbf{e}_i^\top \mathbf{C}_e \mathbf{e}_i} = \sqrt{\mathbf{e}_i^\top \mathbf{F}_e^\top \mathbf{F}_e \mathbf{e}_i} = \mathbf{F}_e \mathbf{e}_i$. By the same reasoning, the diagonal components of $\mathbf{F}_g$ (which are the only non-zero components), give the growth in fiber, crossfiber, and normal direction. Increasing κ did not yield qualitatively different results.

GCG seems to be converging to $F_{g,c,\max}$, and GEG seem to have converged because it reached λ^{crit}. The reason GCG is growing oppositely compared to GEG is probably because GEG was created to model growth triggered by volume overload while GCG was created to capture growth triggered by pressure overload. KFR grew an equal amount in each direction, and stabilized. It is not clear under what conditions KFR should be stable because s_{hom} is the same in each direction. When we ran simulations with other boundary conditions, the solution diverged. The KOM model was stable for many different types of boundary conditions, but is the most computationally expensive model to run.

4.4 Conclusion and future work

We have implemented a general growth and remodelling framework using the FEniCSx program in Python. The goal is to easily change material models and growth models. This will allow researchers to compare their models with other models in the field. Future work will include implementing more complex geometries, more growth laws, and more material models. The models we have used in this paper are not derived from the dissipation equation but are instead phenomenologically derived growth laws and future work should investigate whether or not they satisfy the laws of thermodynamics. Another avenue of future research we wish to pursue is to look into constrained mixture models which model how changes in the various constituents influence the characteristics of the tissue. We also wish to add models that have more mathematically sophisticated stopping criteria, such as the ones developed by Erlich et al. (Erlich and Recho, 2023) uses an energy penalty to construct a stopping criteria and (Erlich and Zurlo, 2024) looks into how curvature[2] in the reference configuration could be used as a stopping criteria. Future work will also implement the growth models on geometries with fibers. We tried running the

[2] The intrinsic three dimensional curvature, not the two dimensional curvature of the surface of the body.

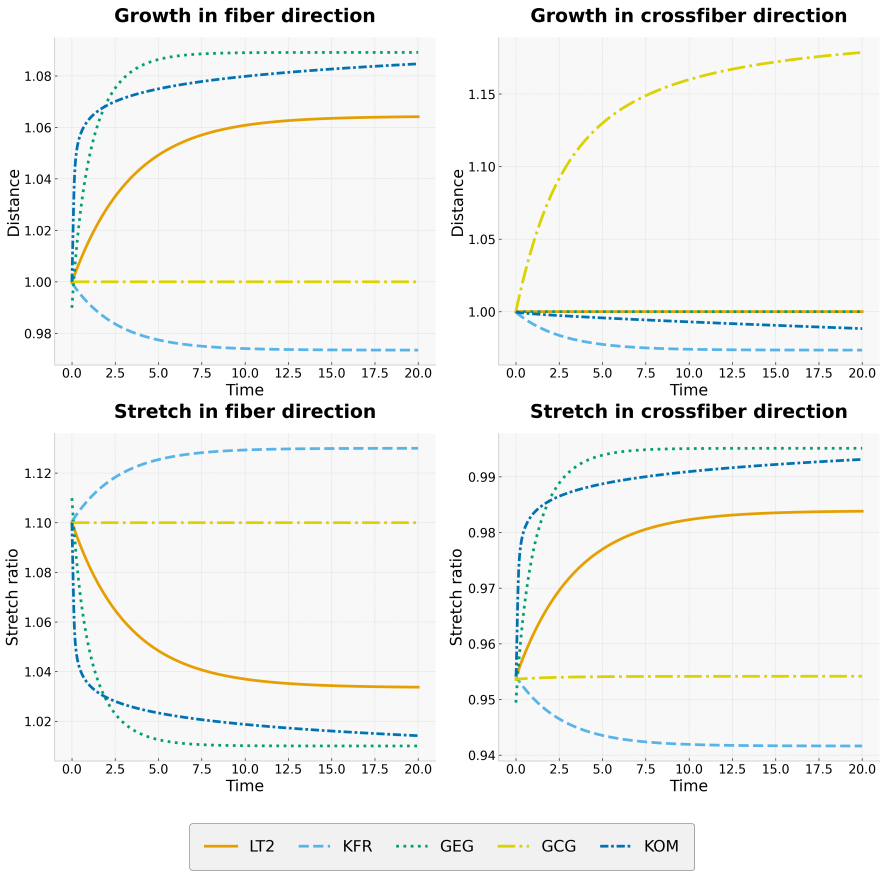


Fig. 4.1: Growth and stretch predicted by a 10% stretch in the fiber direction.

models on various fiber orientations and found them extremely sensitive to fibers that varied throughout the domain. Preliminary results indicate that some of the models do not converge to a steady state for relatively small perturbations of the variables or if the fibers are not well aligned with the body, something we plan on quantifying in the future.

When this package is further developed we aim to add it to the Pulse[3] package.

The models we compared have been developed to capture different aspects of growth and were tuned to be used on different material models, so an apples-to-apples comparison might not be fair. Furthermore, the growth models we have used are not taking into account residual stresses that exist within the material before or after growth occurs.

[3] https://github.com/finsberg/fenicsx-pulse

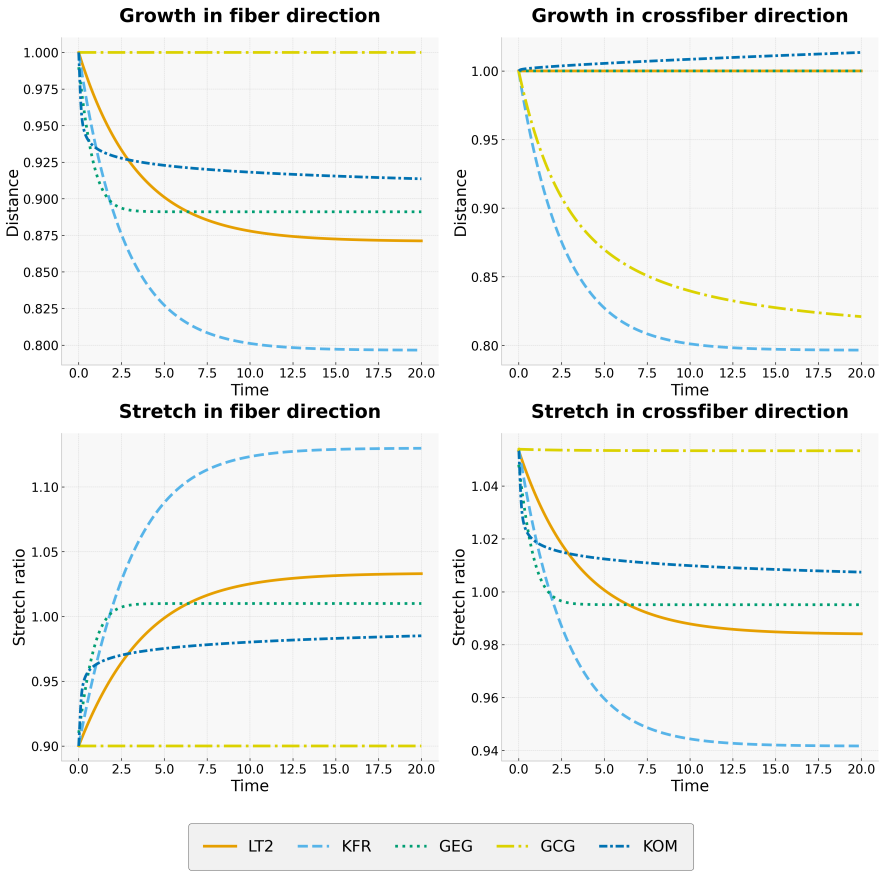


Fig. 4.2: Growth and stretch predicted by a 10% compression in the fiber direction.

Finally, experimental data is needed to verify which models are accurate or capture the correct phenomena of growing cardiac tissue.

References

Baratta IA, Dean JP, Dokken JS, HABERA M, HALE J, Richardson CN, Rognes ME, Scroggs MW, Sime N, Wells GN (30 December 2023) Dolfinx: The next generation fenics problem solving environment, doi:10.5281/zenodo.10447666

Erlich A, Recho P (2023) Mechanical feedback in regulating the size of growing multicellular spheroids. Journal of the Mechanics and Physics of Solids 178, doi:10.1016/j.jmps.2023.105342

Erlich A, Zurlo G (2024) Incompatibility-driven growth and size control during development. Journal of the Mechanics and Physics of Solids 188, doi:10.1016/j.jmps.2024.105660

Goodbrake C, Goriely A, Yavari A (2021) The mathematical foundations of anelasticity: Existence of smooth global intermediate configurations. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 477, doi:10.1098/rspa.2020.0462

Goriely A (2017) The Mathematics and Mechanics of Biological Growth, vol 45. Springer New York, doi:10.1007/978-0-387-87710-5

Goriely A, Amar MB (2007) On the definition and modeling of incremental, cumulative, and continuous growth laws in morphoelasticity. Biomechanics and Modeling in Mechanobiology 6:289–296, doi:10.1007/s10237-006-0065-7

Göktepe S, Abilez OJ, Parker KK, Kuhl E (2010) A multiscale model for eccentric and concentric cardiac growth through sarcomerogenesis. Journal of Theoretical Biology 265, doi:10.1016/j.jtbi.2010.04.023

Holzapfel GA (2002) Nonlinear solid mechanics: A continuum approach for engineering science. Meccanica 37:489–490, doi:10.1023/A:1020843529530, URL https://doi.org/10.1023/A:1020843529530

Hsu FH (1968) The influences of mechanical loads on the form of a growing elastic body. Journal of biomechanics 1:303–311, doi:10.1016/0021-9290(68)90024-9, objectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23

Kerckhoffs RC, Omens JH, McCulloch AD (2012) A single strain-based growth law predicts concentric and eccentric cardiac growth during pressure and volume overload. Mechanics Research Communications 42:40–50, doi:10.1016/j.mechrescom.2011.11.004

Kroon W, Delhaas T, Arts T, Bovendeerd P (2009) Computational modeling of volumetric soft tissue growth: Application to the cardiac left ventricle. Biomechanics and Modeling in Mechanobiology 8:301–309, doi:10.1007/s10237-008-0136-z

Marsden JE, Hughes TJR (1983) Mathematical Foundations of Elasticity. Prentice-Hall, URL https://books.google.no/books?id=mKgyzexDuL0C

Oliveira BLd, Sundnes J (2016) Comparison of tetrahedral and hexahedral meshes for finite element simulation of cardiac electro-mechanics. In: ECCOMAS Congress

Rodriguez EK, Hoger A, Mcculloch AD (1994) Stress-dependent finite growth in soft elastic tissues

Taber LA (1998) Biomechanical growth laws for muscle tissue

Witzenburg CM, Holmes JW (2018) Predicting the time course of ventricular dilation and thickening using a rapid compartmental model. Journal of Cardiovascular Translational Research 11:109–122, doi:10.1007/s12265-018-9793-1

Yavari A (2010) A geometric theory of growth mechanics. Journal of Nonlinear Science 20:781–830, doi:10.1007/s00332-010-9073-y

Chapter 5

Estimation of optimal inlet boundary conditions for blood flow assessment in abdominal aortic aneurysm using variational data assimilation

S. Paratico, R. Munafò, C. Trenti, P. Dyverfeldt, S. Saitta and E. Votta

Abstract

Blood fluid dynamics impacts vessel wall cells and tissue biomechanics, influencing thrombus formation and vessel wall remodeling. Accurate *in vivo* quantification can thus aid in understanding these mechanisms and patient stratification. Computational fluid dynamics (CFD) and 4D flow magnetic resonance imaging (MRI) are used for this purpose, but both have limitations: CFD involves assumptions and boundary condition (BC) simplifications, while 4D flow MRI suffers from low spatial resolution and noise. This study employs variational data assimilation (VarDA) to integrate CFD and 4D flow MRI, yielding a high-resolution, noise-free flow field closely aligned with 4D flow MRI velocity data. To enhance alignment, the optimal inlet velocity profile is determined iteratively via an incremental pressure correction scheme (IPCS). Previously tested in simple synthetic geometries and later in a complex patient-specific abdominal aortic aneurysm (AAA), this approach demonstrates improved reliability in patient-specific hemodynamic evaluation.

Introduction

Alterations in blood fluid dynamics often contribute to the progress of cardiovascular pathological conditions [Bappoo et al. (2021), Guzzardi et al. (2015)]. Hence, quantifying blood fluid dynamics on a patient-specific basis and non-invasively can support the understanding of pathological mechanisms or the stratification of patients based on the risk for adverse endpoints. To this aim, blood flow field can be reconstructed from clinical imaging, namely 4D flow magnetic resonance imaging (MRI) [Dyverfeldt et al. (2015)], or computed through patient-specific computational fluid dynamics (CFD) models [Kheyfets et al. (2015)]. However, 4D flow

S. Paratico e-mail: sara.paratico@mail.polimi.it
Politecnico di Milano

MRI provides indirect and noisy velocity measurements with low spatio-temporal resolution, which often violate mass conservation. CFD models solve discretized Navier-Stokes (NS) equations to compute well-resolved, noise-free velocity fields but are affected by numerical artifacts and rely on simplified boundary conditions (BCs), including inlet velocity BCs. Variational data assimilation (VarDA) integrates CFD-based NS equations with sparse, uncertain 4D flow MRI data by optimizing BCs to minimize discrepancies. In cardiovascular flows, it refines inlet velocity profiles but requires computing both velocity and pressure gradient fields, which is challenging in high-velocity arterial flows. Pressure-velocity coupling or correction schemes address this, yet MRI-induced noise can hinder proper pressure correction, affecting the accuracy of the solution.

Related works

Several studies have explored VarDA in hemodynamics, ranging from 2D steady-state to 3D transient conditions. D'Elia et al. (2012) showed that VarDA allows to reconstruct flow fields in geometrically complex 2D domains, such as the 2D representation of the aortic arch and carotid bifurcation, even with noisy velocity data. Subsequently, in D'Elia and Veneziani (2013), the same authors showed that, in 2D domains, noisy velocity data can be effectively managed by properly managing inlet BCs. In particular, they showed that the regularization of the inlet velocity profile through the use of a control variable also regularized the velocity field over the whole domain and allowed for successful pressure-velocity coupling. Tiago et al. (2017) extended VarDA to a 3D saccular aneurysm, demonstrating its flexibility with various BCs and optimization methods like gradient-based and genetic algorithms to improve accuracy. Koltukluoğlu and Blanco (2018) showed that VarDA, applied to 4D Flow MRI data, outperformed traditional CFD methods by dynamically adjusting BCs in real-time to maintain flow congruence near inlets. Funke et al. (2019) demonstrated the effectiveness of 4D (3D space + time) VarDA in capturing transient blood flow in aneurysms using phase contrast-MRI data. Finally, Dokken et al. (2020) proposed a multimesh finite element (FE) method that enhanced stability and accuracy by allowing flexible BC management across multiple mesh domains, which is key for simulating complex hemodynamics in realistic geometries.

Aim of the work

This study aims to implementing a method to compute *in vivo* blood fluid dynamics on a patient-specific basis with high space-resolution without using simplifying hypotheses on the inlet BCs. To achieve this, VarDA is used to estimate an optimal inlet BC for CFD, starting from a noisy, uncertain 4D flow MRI-based velocity profile while enforcing consistency between the CFD-computed velocity field and

sparse 4D flow MRI data in the bulk flow region. We benchmarked the method on ideal 2D and 3D geometries and then applied it to a patient-specific abdominal aortic aneurysm (AAA) geometry.

Methods

Data Assimilation Method

The VarDA approach was formulated as an optimization problem constrained by the NS equations, using the *dolfin-adjoint* library for the adjoint problem. The process follows three steps: running a first numerical simulation with tentative inlet BCs (which we refer to as the *forward model* or *Tape*), solving the optimization problem to identify inlet BCs, and running a final numerical simulation yielding the refined velocity and pressure fields (Fig. 5.1).

Forward problem definition

The weak and discretized form of NS equations for an incompressible fluid [Stokes (2009)] was solved using the FE platform *FEniCS* [Alnaes et al. (2015)] to compute velocity field $\mathbf{u}$ over a domain Ω given an initial condition $\mathbf{u} = \mathbf{u_0}$ on Ω at time t_0, a zero pressure condition at the outlet section Ω_N and a Dirichlet BC at the inlet section Ω_D in the form of a space- and time-dependent velocity profile $\mathbf{g}$. Through an in house Python script, 2D and 3D fluid domains Ω were discretized into triangular and tetrahedral elements, respectively, with 1-1.5 mm characteristic size and with linear and quadratic shape functions for nodal pressure and velocity, respectively. The semi-implicit Crank-Nicolson (CN) time-integration scheme was applied with a time increment of $\Delta t = 0.001$ s. The incremental pressure correction scheme (IPCS) proposed in [Goda (1979)] was implemented. A generalized minimal residual method (GMRES) was chosen as linear solver, with tolerances of 1×10^{-4} for momentum and continuity equations.

Optimization Problem definition

The optimization problem (eq. (5.1), constrained by the NS equations, aimed to minimize a functional $J(\mathbf{u})$ (eq. (5.2), defined as the difference between the computed and observed velocities.

$$\min_{\mathbf{c}} J(\mathbf{u}) + R(\mathbf{c}) \quad \text{s.t.} \quad F(\mathbf{u}, \mathbf{c}) = 0 \tag{5.1}$$

$$J(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}_{\text{obs}}\|_{L^2(\Omega)} \tag{5.2}$$

To address the ill-posedness of the problem, a Tikhonov regularization term $\mathbf{R}(\mathbf{c})$ was introduced, with respect to the controlled variable defined as c. It accounts for two terms that are scaled by parameters α and β, where β is set to 0 for steady-state conditions. This reformulation transforms the problem into an unconstrained optimization scenario, more suitable for gradient descent methods.

$$\begin{aligned} R(\mathbf{c}) &= \|\mathbf{c}\|_{L^2(\Omega)} \\ &\text{where} \\ \|c\|_{\Gamma\times(0,T]} &= \left(\int_0^T \int_\Omega \frac{\alpha}{2}\left(\left(|\mathbf{g}_D|^2 + |\nabla \mathbf{g}_D|^2\right) + \frac{\beta}{2}\left(|\dot{\mathbf{g}}_D|^2 + |(\nabla \mathbf{g})_D|^2\right)\right) dx\, dt\right)^{\frac{1}{2}} \end{aligned} \tag{5.3}$$

The adjoint approach efficiently computes the total derivative of the functional, yielding the adjoint NS equations that facilitate optimization. The iterative Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, in its L-BFGS variant [Liu and Nocedal (1989)], served as optimizer. The L-BFGS algorithm is already implemented in the SciPy library, which is automatically called by dolfin-adjoint library and which provides many user-friendly numerical routines, such as the routine for optimization.

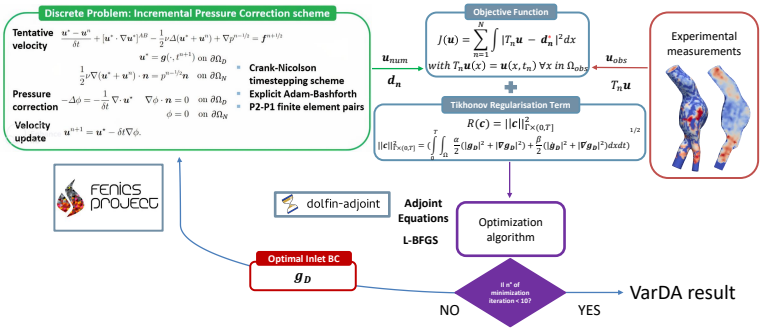


Fig. 5.1: **Top-left**: The finite element (FE) solver computes the Navier-Stokes (NS) equations with an initial guess for the inlet boundary conditions (BCs); **Top-right**: experimental velocity measurements are taken at discrete points in the domain; **Center**: discrepancy between the computational fluid dynamics (CFD) velocity field and experimental data is minimized by iteratively refining inlet velocity profile with a gradient-based method.

Convergence was ensured through Wolfe conditions [Nocedal and Wright (2006)], with maximum number of 10 iterations and ftol = 1×10^{-9} and gtol = 1×10^{-12} as tolerances. The performances of the method were assessed by $J(\mathbf{u})$ value before and after optimization, Root Mean Squared Error (RMSE) between $\mathbf{u}$ and $\mathbf{u}_{\text{obs}}$, and qualitative analysis of the effect on the velocity field through the software *Paraview*.

Benchmark Tests

Preliminary tests - First, preliminary tests were performed to compare the computational efficiency of IPCS vs. an alternative coupled scheme [Figueroa et al. (2006)] in a 2D straight conduit (which represents a case of 2D VarDA and can thus be addressed using the term I defined as *2DVar*). Simulations were run under laminar conditions at both low and high Reynolds numbers (Re), in order to evaluate the method in laminar and transitionally unstable regimes. The conduit was a longitudinal section of a cylinder with a diameter of 41 mm and a length of 200 mm, made of 4,967 mesh elements. Synthetic observations ($\mathbf{u}_{\text{obs}}$) were generated by an auxiliary CFD simulation, prescribing a parabolic velocity profile at the inlet with peak velocity $U_{\text{max}} = 600\,\text{mm/s}$ (Re = 6649) and with $U_{\text{max}} = 50\,\text{mm/s}$ (Re = 554) for turbulent and laminar conditions, respectively. In the Tape, the tentative inlet velocity profile was parabolic with $U_{\text{max}} = 800\,\text{mm/s}$ (Re = 8865) and $U_{\text{max}} = 100\,\text{mm/s}$ (Re = 1108). The iterative minimization of the discrepancy between $\mathbf{u}$ and $\mathbf{u}_{\text{obs}}$ was performed to determine the optimal velocity profile for CFD simulations, verifying if it matched the parabolic profile used to generate the synthetic observations.

Progressively demanding tests - Next, the method was benchmarked through three progressively more demanding tests:

1. 2D straight conduit in transient conditions (which represents a case of 2D VarDA also involving time and can thus be addressed using the term I defined as *2DVar+t* benchmark) - This benchmark shared the same domain of the 2DVar benchmark. However, both the auxiliary simulation for the generation of the experimental observations and Tape consisted in a sequence of two transient simulations: in the first simulation, velocity was initially equal to 0 mm/s everywhere in the domain, and at the inlet a parabolic velocity profile was imposed, whose peak velocity increased linearly from 0 to $\frac{U_{\text{max}}}{2}$ over 0.3 s; in the second simulation, the velocity field computed by the first one was used as IC and the inlet velocity parabolic profile was scaled by the time-dependent function $f(t)$:

$$f(t) = \begin{cases} \frac{U_{\text{max}}}{2} \cos\left(\frac{\pi}{T_s}\left(t - \frac{T_s}{2}\right)\right), & \text{if } t \leq T_s \\ \frac{U_{\text{max}}}{2}, & \text{if } T_s < t \leq T_d \end{cases} \tag{5.4}$$

   where $T_s = 300$ ms and $T_d = 540$ ms are cardiac cycle's systolic and diastolic phases [Katz (1977)]. Besides determining the optimal velocity profile for CFD simulations, spatial and temporal regularization terms [eq. (5.3)] were incorporated into the optimization process and subjected to a sensitivity analysis.

2. 3D straight conduit in steady-state conditions (which represents a case of 3D VarDA and can thus be addressed using the term I defined as *3DVar* benchmark) - This benchmark evaluated the computational cost increase when transitioning from a 2D to a 3D problem. The fluid domain was a 3D cylinder with a radius of

30 mm and a length of 200 mm, made of 74,968 mesh elements. IC and BCs, as well as the objective function, were identical to those in the 2DVar benchmark.

3. Patient-specific AAA geometry in steady-state conditions (which represents a case of 3DVar applied to a patient-specific AAA geometry and can thus be addressed using the term I defined as AAA benchmark) - This benchmark aimed to test VarDA in a complex 3D domain using real experimental observations. 4D flow imaging was acquired from an adult male with AAA using a 3T coronary magnetic resonance (CMR) system (Ingenia, Philips Healthcare, Netherlands) at Linköping University Hospital. The 4D flow data were processed with in-house Python [Saitta et al. (2024)] and CMR angiography was performed for 3D AAA geometry segmentation. Two tests were carried out with laminar flow in the AAA. In the first test, observations consisted in 4DFlow data acquired during early systole, corresponding to the third time frame (about 63 ms from the start of the cardiac cycle, with a 21 ms time step), while the Tape was generated by a CFD simulation fed by 4DFlow-based inlet velocity profiles. The second test assessed the method's robustness against noise, using an inlet velocity profile, scaled by 0.15, at peak systole to produce the Tape's output. Noisy observations were generated by processing the Tape's output according to medium noise setting of Saitta et al. (2024). In addition to already-mentioned metrics, wall shear stresses (WSSs) from final simulation we analyzed using custom Paraview filters.

The associated codes can be found at the following link: https://github.com/saraparatico/proceedingsCodes/tree/main.

Results

Computational costs

Numerical experiments were conducted on various setups: a workstation with 24 CPUs and 64 GB RAM for 2DVar and 3DVar and a high-performance computing system with 40 CPUs and 190 GB RAM for 2DVar+t and AAA benchmark. 2DVar took 30 minutes and 3DVar took 6 hours on 12 CPUs; on the other side, 6 hours were required for 2DVar+t and 17 hours for AAA benchmark.

Preliminary tests

In high Reynolds number tests, IPCS optimization reduced RMSE from 142.60 mm/s to 6.70 mm/s, while the coupled scheme faced convergence issues. In low Reynolds conditions, IPCS proved to be five times faster than the coupled scheme and achieved a final RMSE of 1.76 mm/s compared to 4.22 mm/s for the coupled scheme.

Progressively Demanding Tests

2DVar + t benchmark - When a zero-velocity field was imposed as IC, the post-optimization velocity field showed inconsistencies with respect to the observations. In particular, a high-velocity region just downstream of the inlet section was obtained, while low velocity values were computed in the rest of the domain. Moreover, these tests did not yield improvements by changing α, and increasing β further worsened the performance (Fig. 5.2).

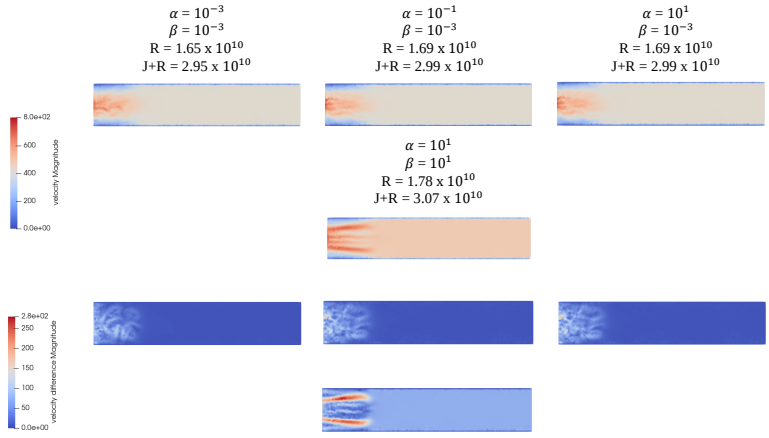


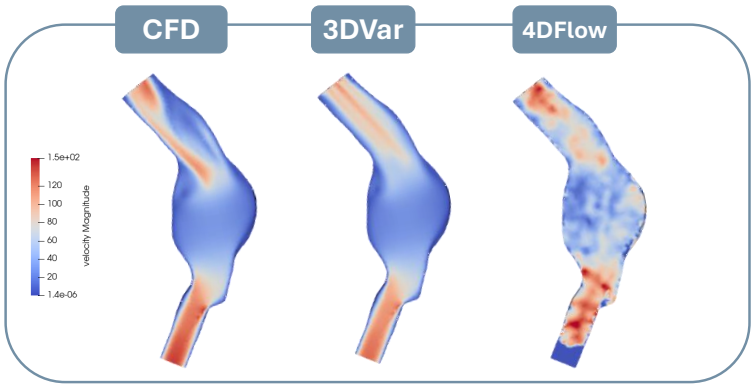
Fig. 5.2: Sensitivity analysis of regularization parameters at systolic peak. First row: 2DVar + t velocity magnitude for different α values (10^{-3}, 10^{-1}, 10^{1}), with $\beta = 10^{-3}$. Second row: test with $\beta = 10^{1}$ to assess time regularization. Third and fourth rows: velocity difference between reference and 3DVar results for each α and β.

When the initial velocity and pressure fields were set equal to those yielded by the previous post-optimization simulation, the results showed a more homogeneous flow better matching parabolic characteristics and with lower $J + R$ values.

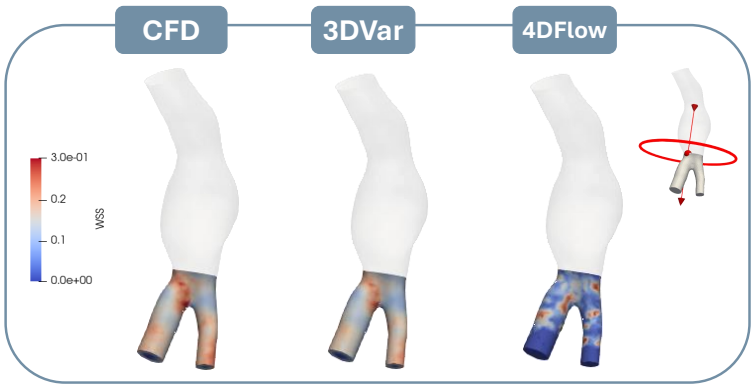
3DVar benchmark - VarDA was performed with spatial regularization terms set to $\alpha = 10^{-2}$, 10^{1}, and 10^{3}. The lowest value of $J + R$ was achieved with $\alpha = 10^{-2}$, but it did not correspond to the lowest RMSE. The velocity field exhibited a peak near the inlet, suggesting continuity loss. For $\alpha = 10^{1}$, the lowest RMSE was achieved, and the post-optimization velocity field more accurately reflected the observed data. Increasing α to 10^{3} resulted in significant deviations from the observations, with unexpected velocity behaviors and higher values of $J + R$ and RMSE. This suggests that lower α values improve RMSE, while higher α values lead to smoother solu-

tions, but can introduce inaccuracies when too large.

AAA benchmark - In first tests, $RMSE$ improved from 59.3 mm/s to 55.1 mm/s, indicating a better alignment with observations (Fig. 5.3a).



(a) AAA benchmark velocity magnitude maps.



(b) AAA benchmark WSS maps.

Fig. 5.3: (a) velocity and (b) wall shear stress (WSS) fields obtained on the abdominal aortic aneurysm (AAA) computed by computational fluid dynamics (CFD) (**left**), derived directly from 4DFlow MRI (**right**), and by data assimilation (**centre**).

Generating a Tape took about 25 minutes, while optimization required 17 hours with 80 Gb of memory. WSS distributions from Tape's output and 3DVar predictions were

consistent, identifying regions with high shear stress (Fig. 5.3b). WSS distributions from Tape's output and 3DVar predictions were consistent in terms of location of high WSS regions. Moreover, while enforcing consistency with 4DFlow-based velocity measurements, the 3DVar method yielded a regular and realistic WSS distribution. This is a major difference as compared to the WSS distribution estimated directly from 4DFlow data, which was unrealistic owing to their poor space-resolution and to the impact of noise in the near-wall region. In tests with noisy observations, 3DVar effectively reconstructed the velocity field, slightly reducing $RMSE$ from 36.8 mm/s to 36.4 mm/s, while maintaining WSS predictions consistent with CFD results, particularly at the iliac bifurcation, which is where the abdominal aorta splits into two smaller arteries, carrying blood to pelvis and legs.

Discussion

From 2DVar+t to 3DVar

The 2DVar+t reveals challenges due to inertial effects and short simulation durations, causing reconstruction defects from incomplete flow development. Extending simulation time for optimization is impractical due to high computational costs. A potential solution includes proper initialization of CFD simulations and implementing a checkpointing method to reduce computational costs by using only the last cardiac cycle for gradient calculations. Moreover, a key difference between 2DVar+t and 3DVar is the flow field's response to regularization. In 2DVar+t, increasing α has minimal effect due to dominant time-dependent effects, reducing spatial regularization's impact. Additionally, increasing β deteriorates the results, a challenge not present in 3DVar, emphasizing the difficulty of balancing temporal and spatial regularization in dynamic flows. Conversely, in 3DVar, moderate α values (10^1) significantly improve the velocity field, reducing inlet peaks and lowering $RMSE$. However, excessive regularization ($\alpha = 10^3$) causes unrealistic velocity patterns.

AAA benchmark

AAA benchmark effectively reconstructs the velocity field in AAA geometry, maintaining high consistency with data obtained from 4D flow imaging. It identifies high shear stress regions despite challenges due to lower resolution of 4D flow data near boundaries. The method remains robust against noise.

Conclusions

This study applies VarDA to estimate optimal inlet BCs for CFD using noisy 4D flow MRI data, minimizing mismatch with *in vivo* velocity measurements. The method yields a resolved, noise-free velocity field and is validated on 2D, 3D and patient-specific AAA geometries, demonstrating potential for personalized hemodynamic simulations. The IPCS framework enhances efficiency and accuracy in transient flow analyses. Despite challenges in transient cases, this work lays the groundwork for future VarDA advancements with clinical implications.

References

Alnaes 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(100):9–23, doi:10.11588/ANS.2015.100.20553

Bappoo N, Syed M, Khinsoe G, Kelsey L, Forsythe R, Powell J, Hoskins P, McBride O, Norman P, Jansen S, Newby D, Doyle B (2021) Low shear stress at baseline predicts expansion and aneurysm-related events in patients with abdominal aortic aneurysm. Circulation: Cardiovascular Imaging 14(12):1112–1121, doi:10.1161/CIRCIMAGING.121.013160

D'Elia M, Veneziani A (2013) Uncertainty quantification for data assimilation in a steady incompressible navier-stokes problem. ESAIM: Mathematical Modelling and Numerical Analysis 47(4):1037–1057, doi:10.1051/m2an/2012056

D'Elia M, Perego M, Veneziani A (2012) A variational data assimilation procedure for the incompressible navier-stokes equations in hemodynamics. Journal of Scientific Computing 52:340–359, doi:10.1007/s10915-011-9547-6

Dokken JS, Johansson A, Massing A, Funke SW (2020) A multimesh finite element method for the navier–stokes equations based on projection methods. Computer Methods in Applied Mechanics and Engineering 368:113129, doi:10.1016/j.cma.2020.113129

Dyverfeldt P, Bissell M, Barker AJea (2015) 4d flow cardiovascular magnetic resonance consensus statement. Journal of Cardiovascular Magnetic Resonance 17(1):72, doi:10.1186/s12968-015-0174-5

Figueroa CA, Vignon-Clementel IE, Jansen KE, Hughes TJ, Taylor CA (2006) A coupled momentum method for modeling blood flow in three-dimensional deformable arteries. Computer Methods in Applied Mechanics and Engineering 195(41):5685–5706, doi:https://doi.org/10.1016/j.cma.2005.11.011

Funke SW, Nordaas M, Øyvind Evju, Alnæs MS, Mardal KA (2019) Variational data assimilation for transient blood flow simulations: Cerebral aneurysms as an illustrative example. International Journal for Numerical Methods in Biomedical Engineering 35(1):e3152, doi:10.1002/cnm.3152

Goda K (1979) A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. Journal of Computational Physics 30(1):76–95, doi:10.1016/0021-9991(79)90088-3

Guzzardi D, Barker A, Van Ooij P, Malaisrie S, Puthumana J, Belke D, Mewhort H, Svystonyuk D, Kang S, Verma S, Collins J, Carr J, Bonow R, Markl M, Thomas J, McCarthy P, Fedak P (2015) Valve-related hemodynamics mediate human bicuspid aortopathy: Insights from wall shear stress mapping. Journal of the American College of Cardiology 66(8):892–900, doi:10.1016/j.jacc.2015.06.1310

Katz AM (1977) Physiology of the heart. URL `https://api.semanticscholar.org/CorpusID:262326887`

Kheyfets V, Rios L, Smith T, Schroeder T, Mueller J, Murali S, Lasorda D, Zikos A, Spotti J, ReillyJr J, Finol E (2015) Patient-specific computational modeling of blood flow in the pulmonary arterial circulation. Computer Methods and Programs in Biomedicine 120(2):88–101, doi:10.1016/j.cmpb.2015.04.005

Koltukluoğlu TS, Blanco PJ (2018) Boundary control in computational haemodynamics. Journal of Fluid Mechanics 847:329–364, doi:10.1017/jfm.2018.329

Liu DC, Nocedal J (1989) On the limited-memory bfgs method for large scale optimization. Mathematical Programming 45:503–528, doi:10.1007/BF01589116

Nocedal J, Wright SJ (2006) Numerical Optimization, 2nd edn. Springer, doi:10.1007/978-0-387-40065-5

Saitta S, Carioni M, Mukherjee S, Schönlieb CB, Redaelli A (2024) Implicit neural representations for unsupervised super-resolution and denoising of 4d flow mri. Computer Methods and Programs in Biomedicine 246:108057, doi:10.1016/J.CMPB.2024.108057

Stokes GG (2009) On the Theories of the Internal Friction of Fluids in Motion, and of the Equilibrium and Motion of Elastic Solids, Cambridge University Press, p 75–129. Cambridge Library Collection - Mathematics, doi:10.1017/CBO9780511702242.005

Tiago J, Guerra T, Sequeira A (2017) A velocity tracking approach for the data assimilation problem in blood flow simulations. International Journal for Numerical Methods in Biomedical Engineering 33(10):e2856, doi:10.1002/cnm.2856

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