AN ENSEMBLE SOLVER FOR SEGREGATED CARDIOVASCULAR FSI

A PREPRINT

Xue Li
Department of Applied and Computational Mathematics and Statistics
University of Notre Dame
Notre Dame, IN 46556
xli26@nd.edu

Daniele E. Schiavazzi
Department of Applied and Computational Mathematics and Statistics
University of Notre Dame
Notre Dame, IN 46556
dschiavazzi@nd.edu

May 23, 2022

ABSTRACT

Computational models are increasingly used for diagnosis and treatment of cardiovascular disease. To provide a quantitative hemodynamic understanding that can be effectively used in the clinic, it is crucial to quantify the variability in the outputs from these models due to multiple sources of uncertainty. To quantify this variability, the analyst invariably needs to generate a large collection of high-fidelity model solutions, typically requiring a substantial computational effort. In this paper, we show how an explicit-in-time *ensemble* cardiovascular solver offers superior performance with respect to the embarrassingly parallel solution with implicit-in-time algorithms, typical of an inner-outer loop paradigm for non-intrusive uncertainty propagation. We discuss in detail the numerics and efficient distributed implementation of a segregated FSI cardiovascular solver on both CPU and GPU systems, and demonstrate its applicability to idealized and patient-specific cardiovascular models, analyzed under steady and pulsatile flow conditions.

***K*eywords** Ensemble solvers · Uncertainty Quantification · Computational Hemodynamics · Explicit Time Integration · Biomechanics

1 Introduction

In this study we focus on the development of efficient solvers for complex fluid-structure interaction (FSI) phenomena arising in cardiovascular (CV) hemodynamics. For this and many other applications, output variability is induced by uncertainty or ignorance in the input processes, e.g., material property distribution, physiologically sound boundary conditions or model anatomy, resulting from operator-dependent image volume segmentation. In this context, the new paradigm of Uncertainty Quantification (UQ) is rapidly becoming an integral part of the modeling exercise, and an indispensable tool to rigorously quantify confidence in the simulation outputs, enabling robust predictions of greater clinical impact. However, running a complete UQ study on a large scale cardiovascular model is typically associated with a substantial computational cost. Non-intrusive approaches for the solution of the forward problem in uncertainty quantification (also known as *uncertainty propagation*) typically consider the underlying deterministic solver as a black box (see Figure 1), requiring model solutions at various parameter realizations to be *independently* (and possibly simultaneously) computed. The scalability of this paradigm is limited due to two main reasons. First, in the embarrassingly parallel solution of multiple instances of the same problem, a large number of operations is repeated.

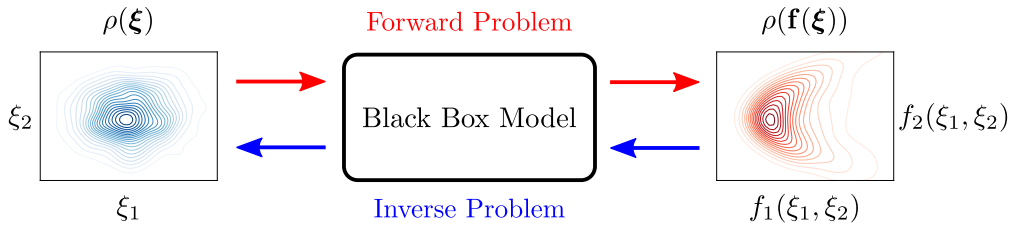


Figure 1: Schematic representation of the forward and inverse problems in UQ.

Second, the solution of large linear systems of equations from numerical integration in time with implicit schemes presents, in general, a less than ideal scalability when computed on large multi-core architectures.

To tackle these challenges, we propose an efficient computational approach for solving multiple instances of the same model (so-called model *ensemble*) at the same time, in a highly scalable fashion, enabled by CPU/GPU implementations of numerical integration schemes that rely heavily on distributed sparse matrix-vector products. We validate our approach by characterizing the effect of variability in vessel wall thickness and elastic modulus on the mechanical response of ideal and patient-specific cardiovascular models analyzed under steady and pulsatile flow conditions. A stochastic model for the spatial distribution of thickness and elastic modulus is provided, in this study, by approximating a Gaussian random field with Matérn covariance through the solution of a stochastic partial differential equation on a triangular finite element mesh of the vessel lumen [1]. Additionally, we follow a one-way coupled, segregated approach to fluid-structure interaction, where the wall stress is computed by an implicit Variational Multiscale fluid solver and passed to a structural, three-d.o.f.s shell model of the vascular walls [2]. Unique contributions of our approach are:

1. We propose, for the first time, an ensemble solver in the context of cardiovascular hemodynamics for UQ, with the aim of drastically reduce the computational effort to perform campaigns of high-fidelity model solutions.
2. Our approach paves the way to a fully explicit treatment of fluid structure interaction in cardiovascular modeling, whose potential has not yet been fully explored in the literature. In our opinion, this new paradigm will provide simpler approaches to simulating complex physiological dysfunctions (e.g. aortic dissection, involving large vessel deformations and contact/auto-contact phenomena, see [3]).
3. The combination of explicit time integration schemes with the solution of model ensembles leads to an efficient distribution of computing load and memory usage in the GPU, enabling scalability in a way that is currently not possible with embarrassingly parallel model solutions and implicit solvers.

We motivate the computational advantage of explicit-in-time ensemble solvers through a back-of-the-envelope argument. It is well known how explicit numerical integration schemes are only conditionally stable. For structural problems, the larger stable time step is determined through a CFL condition as $0.9 \cdot \Delta t$ where Δt is the amount of time required by an elastic wave to cross the smallest element in the mesh. A reasonable value of Δt for CV modeling is $\Delta t = l_{e,\text{min}}/c$, where $l_{e,\text{min}}$ is the diameter of the circle/sphere inscribed in the smallest element in the mesh, $c = \sqrt{E/\rho}$ is the elastic wave speed, E and ρ the elastic modulus and density of the vascular tissue (assumed homogeneous and isotropic in the current argument). Typical values of $l_{e,\text{min}} = 1.0 \times 10^{-3}$ m, $E = 0.7$ MPa and $\rho = 1.06$ kg/m^3 lead to a stable time step equal to approximately $0.9 \cdot 1.2{\times}10^{-6} \approx 1.0{\times}10^{-6}$. In contrast, the typical time step adopted by implicit CV FSI solvers is one millisecond (an upper bound, typically much smaller). At every time step, these solvers require multiple linear systems to be solved with iterative methods, each consisting in several matrix-vector products per iteration. Thus, if we assume an average of 10 non-linear iterations consisting of 10 linear solver iterations each (with two matrix-vector multiplications per iteration), an explicit solver is only a factor of five more expensive. However, the cost of explicit methods can be further reduced through several means, for example, increasing the critical time step via mass scaling (see, e.g., [4]), selectively updating the stiffness, damping and mass matrices between successive time steps to avoid the repeated assembly of element matrices (in the linear regime), and by solving multiple realizations of the boundary conditions, material properties and geometry at the same time. Additionally, large matrix-vector operations needed by the explicit solution of model ensembles are particularly well suited for GPU computing. In summary, explicit time integration schemes have many advantages over implicit approaches in the simulation of phenomena occurring over small time intervals and, combined with the solution of model ensembles, bear substantial potential to boost the efficiency of solving CV models on modern GPU-based systems. Even though our work focuses on CV hemodynamics, the proposed solver paradigm is applicable to study fluid-structure interaction phenomena in other fields, but its efficiency is affected by the stiffness and mass properties in the selected application.

Use of explicit structural solvers in cardiovascular flow problems is mainly related to their flexibility in modeling complex contact configurations. Studies involving coronary stent deployment following endoscopic balloon inflation

are proposed, e.g., in [5, 6], and coupling with an implicit fluid solver is discussed in [7]. Implementation of structural explicit solvers on GPU are discussed in various studies in the literature. In [8] the authors describe in detail an application involving thin shells, while an overview on applications in biomechanics is discussed in [9]. Additionally, Ensemble methods for fluid problems have been recently proposed by [10, 11, 12, 13, 14] in the context of the Navier-Stokes equations with distinct initial conditions and forcing terms. This is based on the observation that solution of linear systems is responsible for a significant fraction of the overall running time for linearly implicit methods, and that is far more efficient to solve multiple times a system of equation with the same coefficient matrix and different right-hand-side than different systems altogether. Extensions have also been proposed to magnetohydrodynamics [15], natural convection problems [16] and parametrized flow problems [17, 18]. We note how, in our case, there is no approximation introduced in the formulation of the ensemble numerical scheme. In addition, no ensemble method appears to be available from the literature in the context of fluid-structure interaction problems.

The basic methodology behind the proposed solver is discussed in Section 2, including the generation of random field material properties, the structural finite element formulation, the variational multiscale fluid solver, thier weak coupling and CPU/GPU implementation. Validation of the proposed approach is discussed in Section 3 with reference to an idealized model of the thoracic aorta and a patient-specific coronary model. Performance and scalability of the approach is discussed in Section 3.3 followed by conclusions and future work in Section 4.

2 Methodology

2.1 Random field material properties

A homogeneous and isotropic vascular tissue with uncertain elastic modulus and thickness is assumed in this study, modeled through a Gaussian random field with Matérn covariance. A Gaussian marginal distribution appears to be the simplest idealized distribution compatible with the scarce experimental observations, while the choice of a Matérn covariance relates to its finite differentiability, which make this model more desirable than other kernels [19]. Let $\|\cdot\|$ denote the Euclidean distance in $\mathbb{R}^d$. The Matérn covariance between two points at distance $\|\boldsymbol{h}\|$ is

$$r(\|\boldsymbol{h}\|) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa\|\boldsymbol{h}\|)^\nu K_\nu(\kappa\|\boldsymbol{h}\|),\ \ \boldsymbol{h} \in \mathbb{R}^d \tag{1}$$

where $\Gamma(\cdot)$ is the gamma function, K_ν is the modified Bessel function of the second kind, σ^2 is the marginal variance, ν is a scaling parameter which determines the mean square differentiability of the underling process, and κ is related to the correlation length $\rho = \sqrt{8\nu}/\kappa$, i.e., the distance which corresponds to a correlation of approximately 0.1, for all ν. It is known from the literature [20, 21] how Gaussian random fields with Matérn covariance can be obtained as solutions of a linear fractional stochastic partial differential equation (SPDE) of the form

$$(\kappa^2 - \Delta)^{\alpha/2}\, x(\boldsymbol{s}) = \mathcal{W}(\boldsymbol{s}),\ \ \boldsymbol{s} \in \mathbb{R}^d, \tag{2}$$

where $\alpha = \nu + d/2$, $\kappa > 0$, $\nu > 0$, $\mathcal{W}(\boldsymbol{s})$ is a white noise spatial process, $\Delta = \Sigma_i\, \partial^2/\partial\, s_i^2$, and the marginal variance is

$$\sigma^2 = \frac{\Gamma(\nu)}{\Gamma(\nu + d/2)(4\pi)^{d/2}\kappa^{2\nu}}.$$

Since the lumen wall is modelled with a surface of triangular elements, we are interested in generating realizations from discretely indexed Gaussian random fields. This is achieved through an approximate stochastic weak solution of the SPDE (2), as discussed in [1]. We construct a discrete approximation of the solution $x(\boldsymbol{s})$ using a linear combination of basis functions, $\{\psi_k\}, k = 1, \ldots, n$, and appropriate weights, $\{w_k\}, k = 1, \ldots, n$, i.e., $x(\boldsymbol{s}) = \sum_{k=1}^n \psi_k(\boldsymbol{s})\, w_k$. We then introduce an appropriate Sobolev space with inner product $\langle \cdot, \cdot \rangle$, a family of *test functions* $\{\varphi_k\}, k = 1, \ldots, n$, and derive a Galerkin functional for (2) of the form

$$\langle \varphi_i, (\kappa^2 - \Delta)^{\alpha/2}\, \psi_j \rangle\, w_j = \langle \varphi_i, \mathcal{W} \rangle \tag{3}$$

We then choose $\varphi_k = (\kappa^2 - \Delta)^{1/2}\, \psi_k$ for $\alpha = 1$ and $\varphi_k = \psi_k$ for $\alpha = 2$, leading to precision matrices $\boldsymbol{Q}_\alpha$ expressed as

$$\begin{aligned} \boldsymbol{Q}_\alpha &= \varkappa^2 \boldsymbol{C} + \boldsymbol{G} && \text{for } \alpha = 1 \\ \boldsymbol{Q}_\alpha &= (\varkappa^2 \boldsymbol{C} + \boldsymbol{G})^T \boldsymbol{C}^{-1} (\varkappa^2 \boldsymbol{C} + \boldsymbol{G}) && \text{for } \alpha = 2 \\ \boldsymbol{Q}_\alpha &= (\varkappa^2 \boldsymbol{C} + \boldsymbol{G})^T \boldsymbol{C}^{-1} \boldsymbol{Q}_{\alpha-2} \boldsymbol{C}^{-1} (\varkappa^2 \boldsymbol{C} + \boldsymbol{G}) && \text{for } \alpha > 2, \end{aligned} \tag{4}$$

where, for $\alpha \geq 3$ a recursive Galerkin formulation is used, letting $\alpha = 2$ on the left-hand side of equation (2) and replacing the right-hand side with a field generated by $\alpha - 2$, assigning $\varphi_k = \psi_k$, $k = 1, \ldots, n$. Note how the use of

piecewise linear basis functions $\{\psi_k\}, k = 1, \ldots, n$, lead to matrices

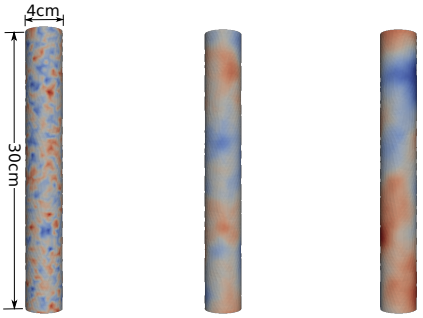
$$\boldsymbol{G}_{ij} = \langle \nabla\psi_i, \nabla\psi_j \rangle \text{ and } \boldsymbol{C}_{ij} = \langle \psi_i, \psi_j \rangle, \tag{5}$$

that are *sparse*, and often found in the finite element discretization of second order elliptic PDEs. However, the precision matrices $\boldsymbol{Q}_\alpha$ are, in general, not sparse as they contain the inverse $\boldsymbol{C}^{-1}$. Thus, by replacing the matrix $\boldsymbol{C}$ with the *lumped* diagonal matrix $\widetilde{\boldsymbol{C}}$, sparsity is restored and $\boldsymbol{Q}_\alpha$ can be efficiently manipulated and decomposed through fast routines for sparse linear algebra available on a wide range of architectures. In addition, the introduction of $\widetilde{\boldsymbol{C}}$ leads to non zero terms on each row only for the *immediate neighbors* $\mathcal{I}(s_k)$ of a given node k on the triangular surface mesh, since the basis function $\{\psi_k\}$ is supported only on the elements connected to node k. This reduces the Gaussian random field to a Gaussian Markov random field, for which

$$\begin{aligned} &\rho\left(x(s_i), x(s_k) \,|\, x(s_j),\ s_j \in \mathcal{I}(s_i)\right) = \\ &= \rho\left(x(s_i) \,|\, x(s_j),\ s_j \in \mathcal{I}(s_i)\right), \end{aligned} \tag{6}$$

or, in other words, *given the value of x on its neighbors*, at any node k the random field $x(s_k)$ is statistically independent from any other location. The approximation error introduced in (6) is, however, small and often negligible in applications [22]. The interested reader is referred to [1] for additional detail on the derivation of $\boldsymbol{Q}_\alpha$.

Numerically generated realizations for various correlation lengths are shown in Figure 2 on an ideal cylindrical representation of the descending thoracic aorta, while Figure 3 shows the agreement of the generated field with the Matérn model.



(a) $\rho = .95\,\text{cm}$ **(b)** $\rho = 3.7\,\text{cm}$ **(c)** $\rho = 7.2\,\text{cm}$

Figure 2: Random field generated on a cylindrical mesh for various correlation lengths.

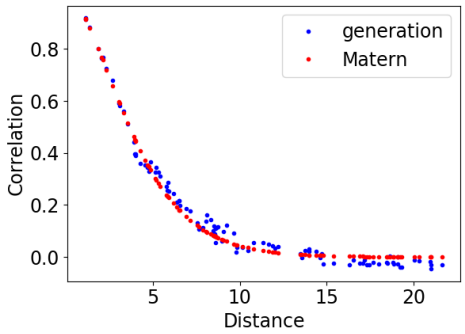


Figure 3: Comparison between spatial correlations from a numerically generated field $x(\boldsymbol{s})$ with precision matrix $\boldsymbol{Q}_\alpha$ and the exact Matérn model.

2.2 A segregated solver for fluid-structure interaction phenomena

2.2.1 Finite element model for the vessel wall

We use a small strain, linear, 3 d.o.f. elastic thin shell which allows for a full compatibility between the fluid mesh discretized with tetrahedral elements and the solid walls. The in-plane stiffness of the shell is complemented with a transverse shear stiffness which provides stability under transverse loading [2]. Using a superscript l and lower case x, y, z to indicate quantities expressed in the local shell reference frame, we introduce a constitutive relation in Voigt notation expressed as

$$\boldsymbol{\sigma}^l = \boldsymbol{C} \cdot \boldsymbol{\varepsilon}^l, \text{ with } \boldsymbol{\sigma}^l = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix}, \boldsymbol{\varepsilon}^l = \begin{bmatrix} \partial u_x/\partial x \\ \partial u_y/\partial y \\ (\partial u_x/\partial y + \partial u_y/\partial x) \\ \partial u_z/\partial x \\ \partial u_z/\partial y \end{bmatrix}. \tag{7}$$

We assume $\varepsilon^l_{zz} = 0$, i.e., zero deformation through the thickness, disregarding the effect of both the normal pressure acting at the lumen surface and the Poisson effect due to the membrane deformations. Strains $\boldsymbol{\varepsilon}^l$ and nodal displacements

$\boldsymbol{u}$ are related through the matrix $\boldsymbol{B}$ of shape function derivatives for a linear triangular element, i.e.

$$\boldsymbol{\varepsilon}^l = \boldsymbol{B}\,\boldsymbol{u} = \frac{1}{2\,A_e}\begin{bmatrix} y_{23} & 0 & 0 & y_{31} & 0 & 0 & y_{12} & 0 & 0 \\ 0 & x_{32} & 0 & 0 & x_{13} & 0 & 0 & x_{21} & 0 \\ x_{32} & y_{23} & 0 & x_{13} & y_{31} & 0 & x_{21} & y_{12} & 0 \\ 0 & 0 & y_{23} & 0 & 0 & y_{31} & 0 & 0 & y_{12} \\ 0 & 0 & x_{32} & 0 & 0 & x_{13} & 0 & 0 & x_{21} \end{bmatrix} \begin{bmatrix} u_{x,1} \\ u_{y,1} \\ u_{z,1} \\ u_{x,2} \\ u_{y,2} \\ u_{z,2} \\ u_{x,3} \\ u_{y,3} \\ u_{z,3} \end{bmatrix} \tag{8}$$

where $x_j, y_j,\ j \in \{1,2,3\}$ are the local coordinates of the j-th element node, $x_{ij} = x_i - x_j$ (similarly for y_{ij}), $u_{x,j}, u_{y,j}, u_{z,j}$ are the local nodal displacements and A_e is the triangular element area. The constitutive matrix is expressed as

$$\boldsymbol{C} = \frac{E}{(1-\nu^2)}\begin{bmatrix} 1 & \nu & 0 & 0 & 0 \\ \nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5\,(1-\nu) & 0 & 0 \\ 0 & 0 & 0 & 0.5\,k\,(1-\nu) & 0 \\ 0 & 0 & 0 & 0 & 0.5\,k\,(1-\nu) \end{bmatrix} \tag{9}$$

where E and ν are the Young's modulus and Poisson's ratio coefficient, respectively, and the shear factor k accounts for a parabolic variation of transverse shear stress through the shell thickness (assumed as $5/6$ for a shell with rectangular cross section). Finally, the local element stiffness matrix $\boldsymbol{k}_e \in \mathbb{R}^{9\times 9}$ can be expressed as

$$\boldsymbol{k}_e = \int_{\Omega_e^s} \boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\;\mathrm{d}\Omega_e^s = \sum_{i=1}^{n_{\text{gp}}} \boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\,A_e\,\zeta_i\,w_i, \tag{10}$$

where n_{gp} is the total number of integration points and $\zeta_i, w_i, i \in \{1,2,\ldots,n_{\text{gp}}\}$, are the element thickness and integration rule weights, respectively. In this study, we adopt a three-point Gauss integration rule to capture a linear variation for E and ζ through each element, generated from a Gauss Markov random field with Matérn covariance $\boldsymbol{Q}_\alpha$, i.e., $E = E(x,y,\omega)$ and $\zeta = \zeta(x,y,\omega)$. Nodal vectors $\boldsymbol{E} \sim \mathcal{N}(\overline{\boldsymbol{E}}, \boldsymbol{Q}_\alpha^{-1})$ and $\boldsymbol{\zeta} \sim \mathcal{N}(\overline{\boldsymbol{\zeta}}, \boldsymbol{Q}_\alpha^{-1})$ are generated as

$$\boldsymbol{E} = \overline{\boldsymbol{E}} + (\boldsymbol{L}^T)^{-1}\boldsymbol{z},\ \text{and}\ \boldsymbol{\zeta} = \overline{\boldsymbol{\zeta}} + (\boldsymbol{L}^T)^{-1}\boldsymbol{z}, \tag{11}$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ is a vector with standard Gaussian components and $\boldsymbol{L}$ is the sparse Cholesky factor of $\boldsymbol{Q}_\alpha$, i.e., $\boldsymbol{Q}_\alpha = \boldsymbol{L}\,\boldsymbol{L}^T$. The covariance matrix $\boldsymbol{Q}_\alpha$ is assembled in compressed sparse column (CSC) format and the Cholesky decomposition is computed using the *cholmod* routine provided by the *scikit-sparse* Python library. Finally, the product $(\boldsymbol{L}^T)^{-1}\boldsymbol{z}$ is performed using the *solve_Lt* routine, as the solution of a triangular system.

2.2.2 Variational multiscale finite element fluid solver

The evolution of blood flow and pressure in the human cardiovascular system can be modeled using the Navier-Stokes equations. Even though many simplifying assumptions can be made to the equations to reduce the computational complexity, here we focus on high-fidelity models, i.e., models associated with large discretizations of a three-dimensional ($n_{\text{sd}} = 3$) fluid domain $\Omega^f \subseteq \mathbb{R}^{n_{\text{sd}}}$. The boundary Γ^f of Ω^f coincides with the mid-plane of the solid domain Ω^s, and is partitioned into $\Gamma^f = \Gamma_g^f \cup \Gamma_h^f \cup \Gamma_s^f$ which correspond to the application of Dirichlet, Neumann boundary conditions and interaction with the solid, respectively. Consider also the vector fields $\boldsymbol{h} : \Gamma_h \times (0,T) \to \mathbb{R}^{n_{sd}}$, $\boldsymbol{g} : \Gamma_g \times (0,T) \to \mathbb{R}^{n_{sd}}$, $\boldsymbol{f} : \Omega \times (0,T) \to \mathbb{R}^{n_{sd}}$ and $\boldsymbol{v}^0 : \Omega \to \mathbb{R}^{n_{sd}}$. We would like to solve the problem of finding $\boldsymbol{v}(\boldsymbol{X},t)$ and $p(\boldsymbol{X},t), \forall\,\boldsymbol{X} \in \Omega, \forall\, t \in [0,T]$ such that

$$\begin{cases} \rho\,\dot{\boldsymbol{v}} + \rho\,\boldsymbol{v}\cdot\nabla\boldsymbol{v} = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \boldsymbol{f} & (\boldsymbol{X},t) \in \Omega \times (0,T) \\ \nabla\cdot\boldsymbol{v} = 0 & (\boldsymbol{X},t) \in \Omega \times (0,T), \end{cases} \tag{12}$$

subject to the boundary conditions

$$\begin{cases} \boldsymbol{v} = \boldsymbol{g} & (\boldsymbol{X},t) \in \Gamma_g \times (0,T) \\ \boldsymbol{t_n} = \boldsymbol{\sigma}\cdot\boldsymbol{n} = [-p\,\boldsymbol{I} + \boldsymbol{\tau}]\cdot\boldsymbol{n} = \boldsymbol{h} & (\boldsymbol{X},t) \in \Gamma_h \times (0,T) \\ \boldsymbol{t_n} = \boldsymbol{t}^f & (\boldsymbol{X},t) \in \Gamma_s \times (0,T) \\ \boldsymbol{v}(\boldsymbol{X},0) = \boldsymbol{v}^0(\boldsymbol{X}) & \boldsymbol{X} \in \Omega, \end{cases} \tag{13}$$

where $\boldsymbol{\tau} = \mu(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)$ is the viscous stress tensor resulting from considering blood as a Newtonian fluid. Solution of (12) in weak form requires to define four approximation spaces, i.e., two trial spaces for the velocity $\boldsymbol{v}$ and pressure p

$$\begin{aligned}\mathscr{S}_k^h &= \Big\{\boldsymbol{v} \,|\, \boldsymbol{v}(\cdot,t) \in \boldsymbol{H}^1(\Omega), t \in [0,T], \\ &\qquad \boldsymbol{v}\,|_{\boldsymbol{x}\in\Omega_e} \in P_k(\Omega_e),\, \boldsymbol{v}(\cdot,t) = \boldsymbol{g} \text{ on } \Gamma_g\Big\}, \\ \mathscr{P}_k^h &= \left\{p \,|\, p(\cdot,t) \in L^2(\Omega), t \in [0,T],\, p\,|_{\boldsymbol{x}\in\bar{\Omega}_e} \in P_k(\Omega_e)\right\},\end{aligned}$$

and two test spaces for $\vec{w}$ and q

$$\begin{aligned}\mathscr{W}_k^h &= \Big\{\boldsymbol{w} \,|\, \boldsymbol{w}(\cdot,t) \in \boldsymbol{H}^1(\Omega),\, t \in [0,T], \\ &\qquad \boldsymbol{w}\,|_{\boldsymbol{x}\in\Omega_e} \in P_k(\Omega_e),\, \boldsymbol{w}(\cdot,t) = \boldsymbol{0} \text{ on } \Gamma_g\Big\},\;\; \mathscr{Q}_k^h = \mathscr{P}_k^h\end{aligned}$$

where $\boldsymbol{H}^1(\Omega)$ is the Sobolev space of function triplets in $L^2(\Omega)$ with derivatives in $L^2(\Omega)$ and $P_k(\Omega)$ is the space of polynomials of order k in Ω. The above spaces are separated into large and a small scale contributions $\mathscr{S}_k^h = \overline{\mathscr{S}_k^h} \oplus \widetilde{\mathscr{S}_k^h}$, $\mathscr{W}_k^h = \overline{\mathscr{W}_k^h} \oplus \widetilde{\mathscr{W}_k^h}$ and $\mathscr{P}_k^h = \overline{\mathscr{P}_k^h} \oplus \widetilde{\mathscr{P}_k^h}$ with a corresponding decomposition of velocity and pressures as $\boldsymbol{v} = \overline{\boldsymbol{v}} + \widetilde{\boldsymbol{v}}$ and $p = \overline{p} + \widetilde{p}$, respectively. This decomposition is introduced in a weak form of the Navier-Stokes equations (12) and a *closure* obtained by expressing the small scale variables $\widetilde{\boldsymbol{v}}$ and $\widetilde{p}$ in terms of their large scale counterparts $\overline{\boldsymbol{v}}$ and $\overline{p}$ using [23]

$$\begin{bmatrix}\widetilde{\boldsymbol{v}} \\ \widetilde{\boldsymbol{p}}\end{bmatrix} = \begin{bmatrix}\tau_M\, \boldsymbol{R}_M(\overline{\boldsymbol{v}},\overline{p}) \\ \tau_C\, R_C(\overline{\boldsymbol{v}},\overline{p})\end{bmatrix}, \tag{14}$$

where $\boldsymbol{R}_M$ and R_C represent the momentum and continuity residual expressed as

$$\begin{cases}\boldsymbol{R}_M(\overline{\boldsymbol{v}},\overline{p}) = \boldsymbol{R}_M = \rho\,\dot{\overline{\boldsymbol{v}}} + \rho\,\overline{\boldsymbol{v}}\cdot\nabla\overline{\boldsymbol{v}} + \nabla\overline{p} - \nabla\cdot\boldsymbol{\tau} - \boldsymbol{f}, \\ R_C(\overline{\boldsymbol{v}},\overline{p}) = R_C = \nabla\cdot\overline{\boldsymbol{v}},\end{cases} \tag{15}$$

and the *stabilization* coefficients are expressed as

$$\begin{aligned}\tau_M &= \left(\frac{4}{\Delta t^2} + \boldsymbol{u}\cdot\boldsymbol{G}\,\boldsymbol{u} + C_I\,\nu^2\,\boldsymbol{G}:\boldsymbol{G}\right)^{-1/2} \\ \tau_C &= (\tau_M\,\boldsymbol{g}\cdot\boldsymbol{g})^{-1},\;\; G_{i,j} = \sum_{k=1}^{3}\frac{\partial\xi_k}{\partial x_i}\frac{\partial\xi_k}{\partial x_j},\;\; g_i = \sum_{k=1}^{3}\frac{\partial\xi_k}{\partial x_i},\end{aligned} \tag{16}$$

where $\partial\boldsymbol{\xi}/\partial\boldsymbol{x}$ is the inverse Jacobian of the element mapping between the parametric and physical domains.

To simplify the notation, in what follows the large scale variables $\overline{\boldsymbol{v}}$ and $\overline{p}$ will be denoted simply by $\boldsymbol{v}$ and p and the spaces $\overline{\mathscr{S}_k^h}$, $\overline{\mathscr{W}_k^h}$ and $\overline{\mathscr{P}_k^h}$ by $\mathscr{S}_k^h$, $\mathscr{W}_k^h$ and $\mathscr{P}_k^h$. A weak solution of the Navier-Stokes equations can now be determined by finding $\boldsymbol{v} \in \mathscr{S}_h^k$ and $p \in \mathscr{P}_h^k$ such that

$$\begin{aligned}&B(\boldsymbol{w},q\,;\,\boldsymbol{v},p) = B_G(\boldsymbol{w},q\,;\,\boldsymbol{v},p) + \\ &+\sum_{e=1}^{n_{\text{el}}}\int_{\Omega_e}\left\{(\boldsymbol{v}\cdot\nabla)\,\boldsymbol{w}\cdot(\tau_M\,\boldsymbol{R}_M) + \nabla\cdot\boldsymbol{w}\,\tau_C\,R_C\right\}\,\mathrm{d}\Omega_e + \\ &+\sum_{e=1}^{n_{\text{el}}}\int_{\Omega_e}\left\{\boldsymbol{w}\cdot[-\tau_M\,\boldsymbol{R}_M\cdot\nabla\boldsymbol{v}] + [\boldsymbol{R}_M\cdot\nabla\boldsymbol{w}]\cdot[\overline{\tau}\,\boldsymbol{R}_M\cdot\boldsymbol{v}]\right\}\,\mathrm{d}\Omega_e + \\ &+\sum_{e=1}^{n_{\text{el}}}\int_{\Omega_e}\nabla q\cdot\frac{\tau_M}{\rho}\,\boldsymbol{R}_M\,\mathrm{d}\Omega_e = 0,\end{aligned} \tag{17}$$

for all $\vec{w} \in \vec{\mathscr{W}}_h^k$ and $q \in \mathscr{P}_h^k$, where the Galerkin functional B_G is expressed as

$$\begin{aligned}&B_G(\boldsymbol{w},q\,;\,\boldsymbol{v},p) = \int_\Omega \boldsymbol{w}\cdot(\rho\,\dot{\boldsymbol{v}} + \rho\,\boldsymbol{v}\cdot\nabla\boldsymbol{v} - \boldsymbol{f})\,\mathrm{d}\Omega + \\ &+\int_\Omega\{\nabla\boldsymbol{w}\,:\,(-p\boldsymbol{I}+\boldsymbol{\tau}) - \nabla q\cdot\boldsymbol{v}\}\,\mathrm{d}\Omega + \\ &+\int_{\Gamma_h}\{-\boldsymbol{w}\cdot\boldsymbol{h} + q\,v_n\}\,\mathrm{d}\Gamma + \int_{\Gamma_s}\left\{-\boldsymbol{w}\cdot\boldsymbol{t}^f + q\,v_n\right\}\,\mathrm{d}\Gamma + \\ &+\int_{\Gamma_g} q\,v_n\,\mathrm{d}\Gamma.\end{aligned} \tag{18}$$

The discrete variables $\boldsymbol{w}^h$, $\boldsymbol{v}^h$, q^h and p^h are introduced in (17), leading to a non linear system of equations of the form

$$\begin{cases} \boldsymbol{N}_M(\dot{\boldsymbol{v}}^h, \boldsymbol{v}^h, p^h) = 0 \\ N_C(\dot{\boldsymbol{v}}^h, \boldsymbol{v}^h, p^h) = 0, \end{cases} \tag{19}$$

A predictor-multicorrector scheme [24] is used for time integration and, at each time step, the resulting non linear system (19) is solved through successive Newton iterations. Additional details can be found in [23, 25].

2.2.3 Fluid-structure coupling

Since this study focuses more on the development of an ensemble solver, we provide a one-way coupled, simplified treatment of the interaction between fluid and structure, leaving a more rigorous treatment to future work. Here we assume a fluid model with rigid walls and a structural model of the lumen governed by the three d.o.f. shell discussed in Section 2.2.1. The lumen deformation does not, in turn, affect the geometry of the fluid region. The elastic forces in the lumen wall are in equilibrium with the shear and pressure exerted by the fluid, or, in other words $\boldsymbol{t}^s = -\boldsymbol{t}^f$, where the wall stress $\boldsymbol{t}^f$ computed by the fluid solver is

$$\boldsymbol{t}^f = \boldsymbol{\sigma}^f \cdot \boldsymbol{n}, \;\; \boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} - p\boldsymbol{I}, \tag{20}$$

p is the main *nodal* pressure unknown in the VMS fluid solver, and $\boldsymbol{\varepsilon} = \nabla^s \boldsymbol{u}$ is the symmetric part of the velocity gradient, which is constant on each P1 element in the fluid mesh. An example of shear forces computed by the VMS fluid solver for a coronary model is illustrated in Figure 4. At every node on the wall, the shear forces and normal vectors from adjacent elements are averaged and the nodal pressure added, leading to the three components of the nodal force that are passed to the structural solver.

Finally, stress components in the circumferential, radial and axial directions are obtained by transforming the solid stress tensor to a local cylindrical coordinate system (see Figure 4). For an arbitrary Gauss point or lumen shell node $\boldsymbol{s}$, we identify the closest location on the vessel centerline. The tangent vector to the centerline at $\boldsymbol{s}$ is defined as the local *axial* direction z, the normal at the Gauss point is the local *radial* direction r, and the local *circumferential* direction θ is obtained as the cross product between r and z.

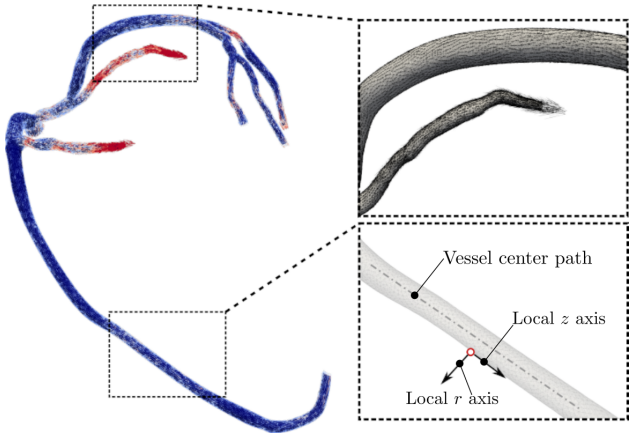


Figure 4: Visualization of interface shear forces exchanged between the fluid and the structural solver. The local axis system used for stress post-processing is also shown in the close up.

2.3 Distributed explicit FSI solver on multiple CPUs

The solution in time for the dynamics of the undamped non-linear structural system

$$\boldsymbol{M}\,\ddot{\boldsymbol{u}} + \boldsymbol{C}\,\dot{\boldsymbol{u}} + \boldsymbol{K}(\boldsymbol{u})\,\boldsymbol{u} = \boldsymbol{f}(t), \tag{21}$$

is computed using central differences in time, resulting in an update formula in the $[n\,\Delta t, (n+1)\,\Delta t]$ intervals expressed as

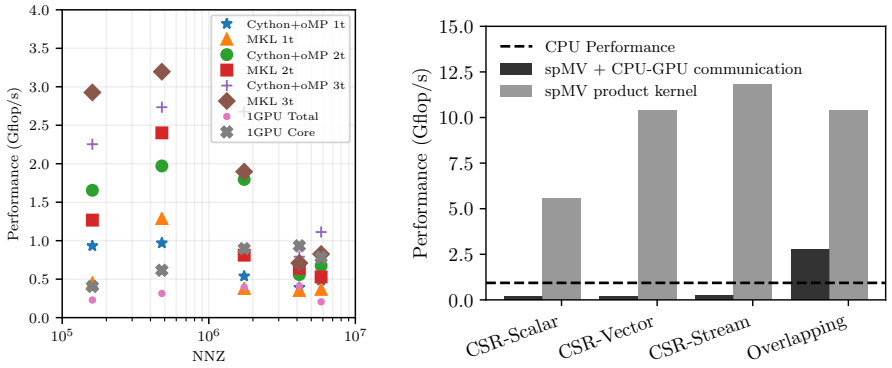
$$\left(\widetilde{\boldsymbol{M}} + \frac{\Delta t}{2}\widetilde{\boldsymbol{C}}\right)\,\boldsymbol{u}_{n+1} = \Delta t^2\,\boldsymbol{f}_n - \left(\Delta t^2\boldsymbol{K} - 2\,\boldsymbol{M}\right)\,\boldsymbol{u}_n - \left(\boldsymbol{M} - \frac{\Delta t}{2}\boldsymbol{C}\right)\,\boldsymbol{u}_{n-1} \tag{22}$$

which is performed independently by an arbitrary number of mesh partitions. To do so efficiently, $\widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{C}}$ in (22) are lumped mass matrix pre-assembled before the beginning of the time loop, containing the elemental contributions

from all finite elements, even those belonging to separate mesh partitions. Given the limited amount of deformation typically observed in cardiovascular applications over a single heart cycle, the assumption of a fixed nodal mass over time is considered realistic. This assumption removes the need of communicating nodal masses during finite element assembly, improving scalability. In some cases, a viscous force $\boldsymbol{f}_v$ is added to the right-hand-side in order to damp the high-frequency oscillations as $\boldsymbol{f}_v = -c_d\, \dot{\boldsymbol{u}}_n$, through an appropriate damping coefficient c_d.

Even though the geometry of the vessel lumen is updated at every step by adding $\boldsymbol{u}_{n+1}$, the displacement magnitudes over a typical heart cycle remain of the same order as the thickness of the vessel wall, hence in the linear regime. In this context, the stiffness do not significantly change and it is possible to save computational time by avoiding to assemble it at every iteration. In our code, we therefore provide the option to selectively update the stiffness matrix after a prescribed number of iterations. Synchronization of displacements for the nodes shared by multiple partitions is implemented using *Send*, *Recv* to the root CPU and broadcast back.

The performance of the proposed ensemble solver was first tested on multiple CPUs. We use distributed sparse matrices in the Yale compressed sparse row (CSR) format [26] with dense coefficient entries of size $9 \cdot n_s$, where n_s is the number of material property realizations and a local element matrix of size 3×3 results from selecting a three-d.o.f.s shell finite element. The code for the finite element assembly and sparse matrix-vector multiplication was developed in Cython+MPI+openMP and compared both with a C implementation and with the *mkl_cspblas_dcsrgemv* routine provided through the Intel MKL library, using single and multiple threads. We verified the satisfactory performance of our implementation under a wide range of mesh sizes, number of cores, and with/without multithreading. Encouraging speedups were obtained on multiple CPUs, as shown in Figure 5 and Figure 6.

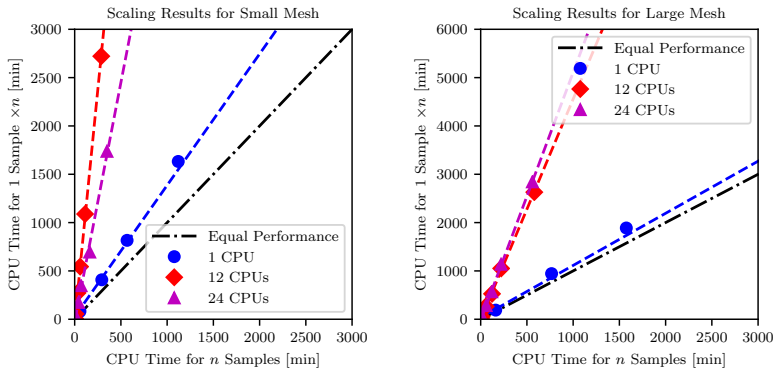


(a) **(b)**

Figure 5: Performance of matrix-vector product kernel on CPU by our Cython+openMP implementation, GPU implementation and MKL library on multiple threads (a). Optimization of GPU matrix-vector kernel and CPU-GPU communication performance (b). Tests were performed using 1 CPU and 1 GPU on a cylindrical model associated with a sparse matrix having nnz = 160,587 and 1,000 material property realizations.

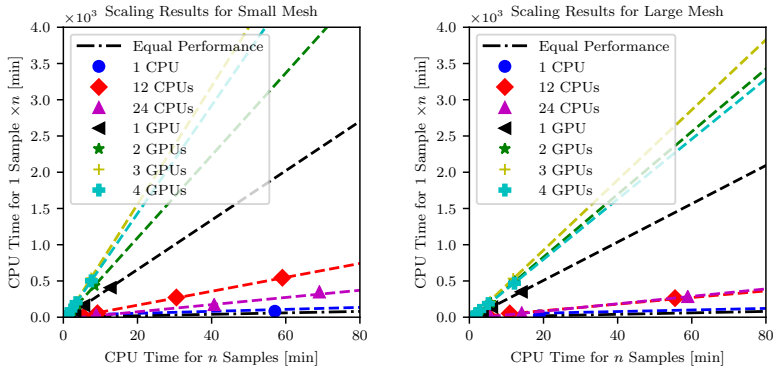
2.4 Distributed explicit FSI solver on multiple GPUs

We developed an hardware-independent openCL implementation of the solver running on multiple GPUs. We started with a naïve CSR-based parallel sparse matrix-vector product (SpMV), known as *CSR-Scalar*, where each row of the sparse matrix is assigned to a separate thread [27]. This works well on CPUs, but causes uncoalesced, slow memory accesses on GPUs, since elements in each row occupy consecutive addresses in memory, but consecutive threads access elements on different rows. In addition, long rows lead to an unequal amount of work among the threads and some of them need to wait for others to finish. We then transitioned to a *CSR-Vector* scheme [28], assigning a wavefront (or so-called *warp* on NVIDIA architectures) to work on a single row of the matrix. This allows for access to consecutive memory locations in parallel, resulting in fast coalesced loads. However, CSR-Vector can lead to poor GPU occupancy for short rows due to unused execution resources. Improved performance can be achieved using *CSR-Stream* [29] which statically fixes the number of nonzeros that will be processed by one wavefront and streams all of these values into the local scratchpad memory, effectively utilizing the GPU's DRAM bandwidth and improving over CSR-Scalar. CSR-Stream also dynamically determines the number of rows on which each wavefront will operate, thus improving over CSR-Vector. While CSR-Stream substantially improves the performance of the spMV product kernel, the CPU-to-GPU data transfer still dominates the time step update, as shown in Figure 5b. This problem can be mitigated by sending data to the GPU in smaller chunks, thus overlapping data transfer and kernel execution.



(a) **(b)**

Figure 6: Performance of explicit ensemble solver on multiple CPUs for a mesh with 5,074 elements, 2,565 nodes (a) and for a mesh with 15,136 elements and 7,628 nodes (b).



(a) **(b)**

Figure 7: Performance of explicit ensemble solver on multiple CPUs/GPUs for a mesh with 5,074 elements, 2,565 nodes (a) and for a mesh with 15,136 elements and 7,628 nodes (b).

In addition, data transfer can be minimized by an assembly-free approach. Even though this is a standard practice in explicit finite element codes, it is particularly effective on a GPU for three reasons. First, storage of a sparse global matrix would occupy a significant portion of the GPU memory, posing restrictions on the model size. Second, indexing operations to access entries in the global matrix would cause severe uncoalesced memory access, reducing significantly the degree of parallelism in the GPU. Third, for the most common sparse matrix storage schemes, indexing always include searching, which would be particularly slow on GPU. We compute local matrices directly in the GPU and take their product with a partition-based displacement vector, where only the displacements of shared nodes need to be synchronized through the root CPU.

To compute element matrices, we buffer data to the GPU before the beginning of the time integration loop, in order to avoid any CPU to GPU data transfer due to element assembly. Buffered quantities include mesh geometry and material properties, particularly the product between the Young's modulus and thickness at each Gauss point for all random field realizations. Note how the rest of the local stiffness matrix is constant for linear triangular elements. In addition, the left-hand-side lumped mass matrix resulting from the central difference scheme does not change throughout the time loop. We also leverage a mesh coloring algorithm, allowing working units to process different elements at the same time. During each time step, each computing group in the GPU works on one element, while the working units in the same group work on different realizations and therefore have access to coalesced GPU memory. Communication is only triggered by displacement synchronization. We use pinned host memory to speed up the data transfer between CPU and GPU. All displacements for each realization and the local matrices are stored in private memory since they do not need to be shared with other working units. Final GPU speed ups are illustrated in Figure 7.

2.5 A Python code-base

The CVFES solver is developed in Python 3 with optimization in Cython [30] and element assembly and matrix product implementation on openCL [31] (though the Python pyOpenCL library [32]). The code leverages the VTK library [33] to read the solid, fluid mesh and boundary conditions. In this context the solver is fully compatible with the input files generated by the SimVascular software platform [34] and can be easily integrated with the SimVascular modeling workflow. Partitioning on multiple CPUs and GPUs is obtained for both solvers using parMETIS [35]. The code used to generate the results discussed in Section 3 is available through a public GitHub repository at https://github.com/desResLab/CVFES.

3 Results

3.1 Ideal cylindrical benchmark

The first benchmark represents an ideal cylindrical lumen subject to aortic flow. The cylinder has a diameter equal to 4 cm and length of 30 cm, while two Matérn random fields for thickness and elastic modulus have been assigned as discussed in Section 2.1. Specifically, a mean $\mu = 7.0 \times 10^6$ Barye and a standard deviation $\sigma = 7.0 \times 10^5$ Barye have been assumed for the elastic modulus, whereas the thickness random field is characterized through a mean $\mu = 0.4$ cm and a standard deviation $\sigma = 0.04$ cm. Three values of the correlation length were considered, equal to 0.95 cm, 3.7 cm and 7.2 cm, respectively (see discussion in [36]). A uniform pressure of 13 mmHg was added to the pressure computed by the VMS fluid solver. We considered diastole as a natural (unstressed) state and applied the difference between a diastolic pressure of 80 mmHg and the mean brachial pressure in a healthy subject (i.e., with systolic pressure equal to 120 mmHg). This configuration is analyzed both under steady state and pulsatile flow conditions, under fully fixed structural boundary conditions at the two cylinder ends.

3.1.1 Steady state analysis

The fluid solution is computed with the VMS fluid solver discussed in Section 2.2.2 using a parabolic velocity profile at the inflow corresponding to a -66.59 mL/s volumetric flow rate, zero-traction boundary condition at the outlet and a no-slip condition at the lumen wall. As expected, the fluid solution show a perfectly linear relative pressure profile along the cylinder center path and a uniform parabolic velocity profile from inlet to outlet, typical of viscous-dominated Poiseuille flow, as shown in Figure 8.

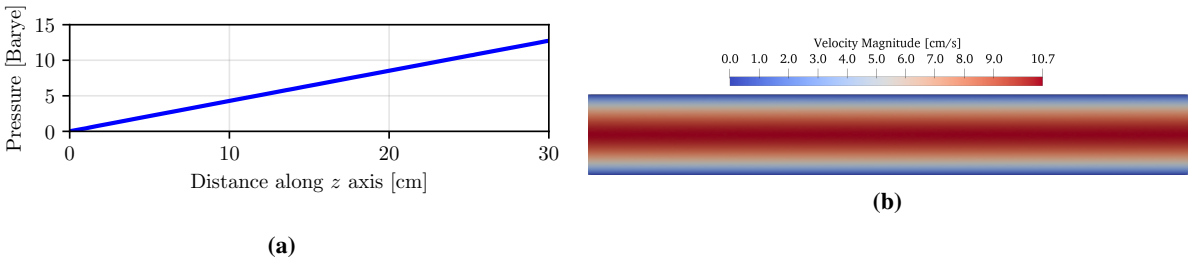
The steady state pressure resulting from the fluid solver is applied as discussed in Section 2.2.3 and 100 thickness and elastic modulus realizations are solved simultaneously. The explicit structural simulation is run for 0.5 seconds, until a steady state was observed. Three wall mesh densities (coarse, medium and fine) are finally considered, consisting of 5,074, 15,136 and 32,994 triangular shell elements, with explicit time steps set to $\Delta t = 4.0 \times 10^{-5}$ s, $\Delta t = 4.0 \times 10^{-5}$ s and $\Delta t = 1.0 \times 10^{-5}$ s, respectively and no viscous damping.

Displacement magnitudes along the longitudinal z axis (cylinder generator) are shown in the top row of Figure 9 for the finer mesh and various correlation lengths. The mean displacement one diameter away from the fully fixed ends (thick black line) is consistent with an homogeneous solution for a thick cylinder with average elastic modulus and thickness (blue dashed line). Displacements associated with single random field realizations are also shown, in Figure 9 (top row), using colors. As expected, the displacement wave length increases with the correlation length, and so does the displacement uncertainty quantified through the 5%-95% confidence interval (gray shaded area). Finally, the second row of Figure 9 shows how increasing the mesh density produces a limited difference in the 5%-95% confidence interval.

Circumferential stress was found to be the most significant component, as expected, showing uncertainty increasing with the correlation length, similarly to what observed for the displacement magnitude. The in-plane and out-of-plane shear components ($\sigma_{\theta z}$ and σ_{rz}), though much smaller than circumferential and axial stresses, are essentially related to the material property non-homogeneity and reduce for an increasing correlation length. These stress components are zero both on average and by solving the model with average material properties. Thus, they can only be captured by explicitly modeling the spatial variability of material properties as in the proposed approach.

3.1.2 Validation under pulsatile flow conditions

A parabolic pulsatile inflow (Figure 11a) was applied to the same cylindrical geometry discussed in the previous section, while all the other boundary conditions (walls and outlets) were kept the same as in the steady case. The time step is set to 1.0×10^{-5} and the simulation run for two heart cycles and 100 material property realizations. To avoid the application of impulsive loads which can excite a broad range of frequencies, significantly affecting the undamped dynamics, a ramp was applied to the wall loads resulting from the fluid solver in the last heart cycle. The ramp follows



(a) **(b)**

Figure 8: Steady state pressure (a) and velocity (b) distributions from variational multi-scale fluid solver.

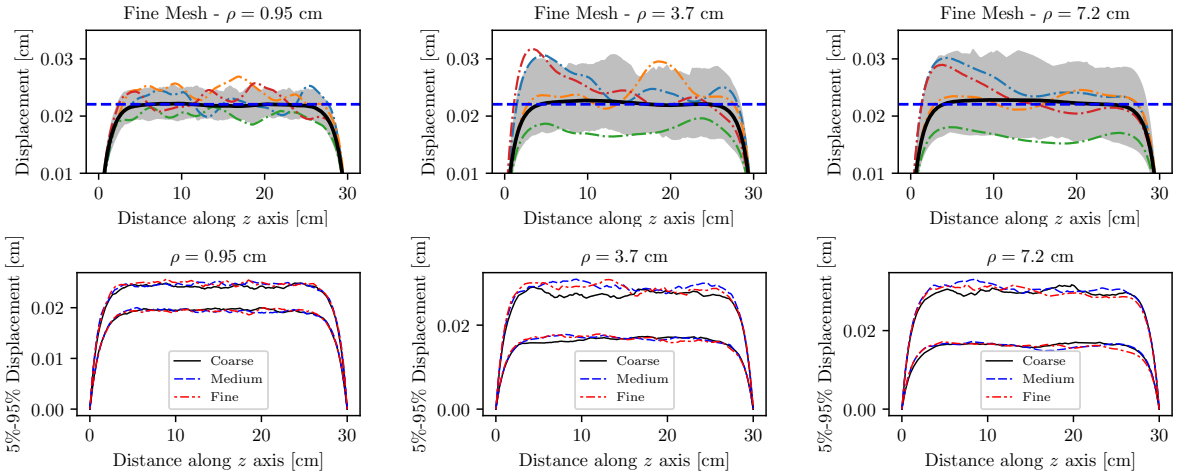


Figure 9: Displacement magnitude along cylinder generator (longitudinal z axis) for various correlation lengths (first row). 5%-95% confidence intervals for displacement magnitudes for three increasing mesh densities (i.e., coarse, medium and fine, second row).

a sine wave and is kept active for the first 0.2 seconds of the simulation. No damping was applied to the simulation, i.e., $\boldsymbol{f}_v = 0$.

As expected, the resulting displacements follow a time profile similar to the inflow, and the 5%-95%confidence intervals increase with the correlation length of the underlying random field. Circumferential and axial stress components are the dominant stress components, which also increase with the correlation length, as observed for the steady state results.

3.2 Benchmark on patient-specific coronary model

We also demonstrate the results of the proposed ensemble solver on a patient-specific model of the left anterior descending coronary branch. An ensemble of 100 model solutions were obtained using random field parameters for the elastic modulus equal to $\mu = 1.15 \times 10^7$ Barye and $\sigma = 1.7 \times 10^6$ Barye. For the thickness, we considered a mean equal to $\mu = 0.075$ cm and a standard deviation of $\sigma = 0.017$ cm. A slip-free boundary condition was applied at the outlets of the fluid domain, whereas fully fixed mechanical restraints (i.e., all three nodal translations) where applied along the edges at both the inlets and outlets. A uniform pressure has been superimposed to the lumen stress obtained from the fluid solver as discussed for the ideal aortic model in Section 3.1.

3.2.1 Steady state analysis

A constant flow rate equal to -0.28 mL/s is applied at the model inlet with a parabolic profile, while a zero-traction boundary condition is applied at the outlets and a no-slip condition at the walls. The explicit time step is set to 2.5×10^{-6} and the model was run for 0.13 seconds to reach the steady state. No additional viscous force $\boldsymbol{f}_v$ was considered. Figure 12 shows the displacements and stress for all three analyzed correlation lengths, averaged through a cross sectional slice of the branch of interest, and plotted for successive slices along the longitudinal z axis.

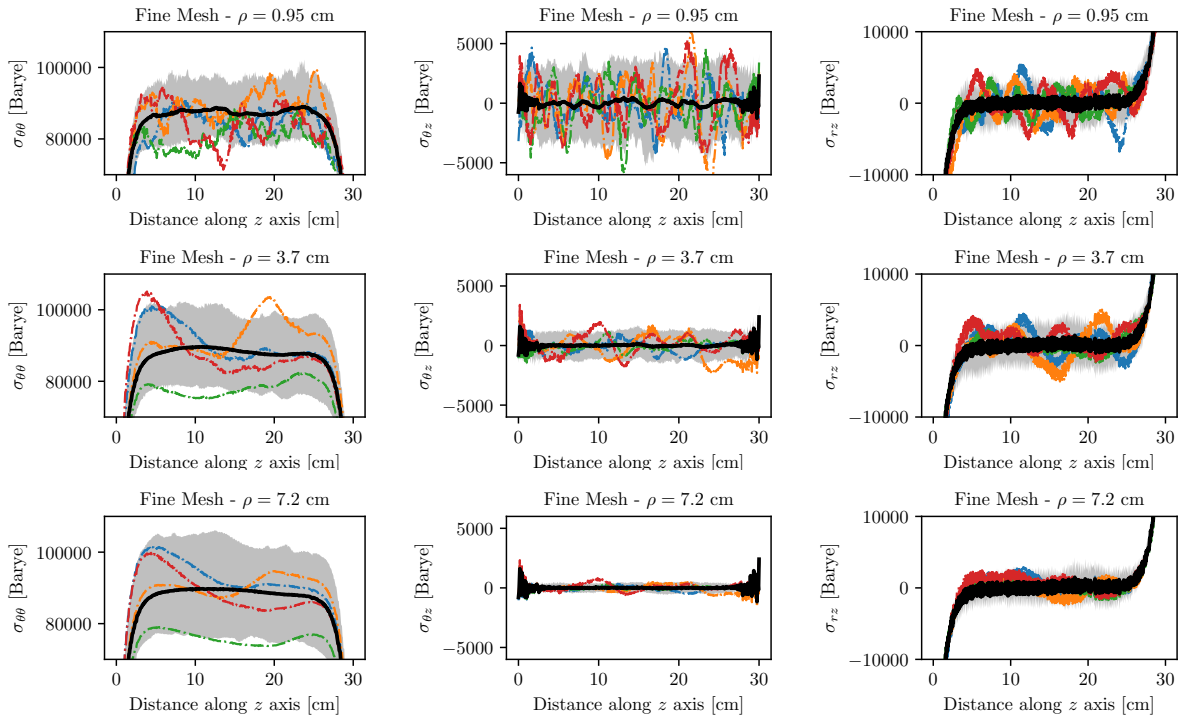
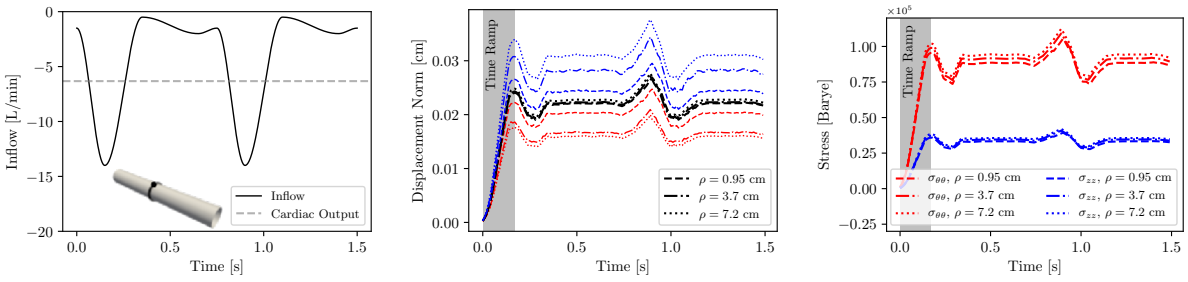


Figure 10: Ensemble means, 5%-95% confidence intervals and single realizations of longitudinal stress profiles for various mesh densities and random field correlation lengths.



(a) **(b)** **(c)**

Figure 11: Results for pulsatile flow on ideal cylindrical vessel. Inflow time history and spatial location for acquisition of displacement and stress outputs (a). Displacement time history at selected spatial location with thick lines representing ensemble averages, and thin lines used for 5%/95% percentiles (b). Circumferential and axial stress time history at selected spatial location (c).

Displacement results confirm the absence of torsion, and a prevalent radial deformation mode, with rigid body motion evident from the axial displacements d_z, affected by the centerline path geometry. The stress results confirm the importance of the circumferential followed by the axial component. The circumferential stress reduces with the coronary branch radius as intuitively suggested by the Barlow (or Mariotte) formula for thin-walled cylinders. Similar cylindrical displacements, circumferential and axial stress are observed for all three correlation lengths. For the selected coronary branch, the smaller correlation length (0.95 cm) is approximately equal to twice the largest diameter, inducing minimal changes in local deformability.

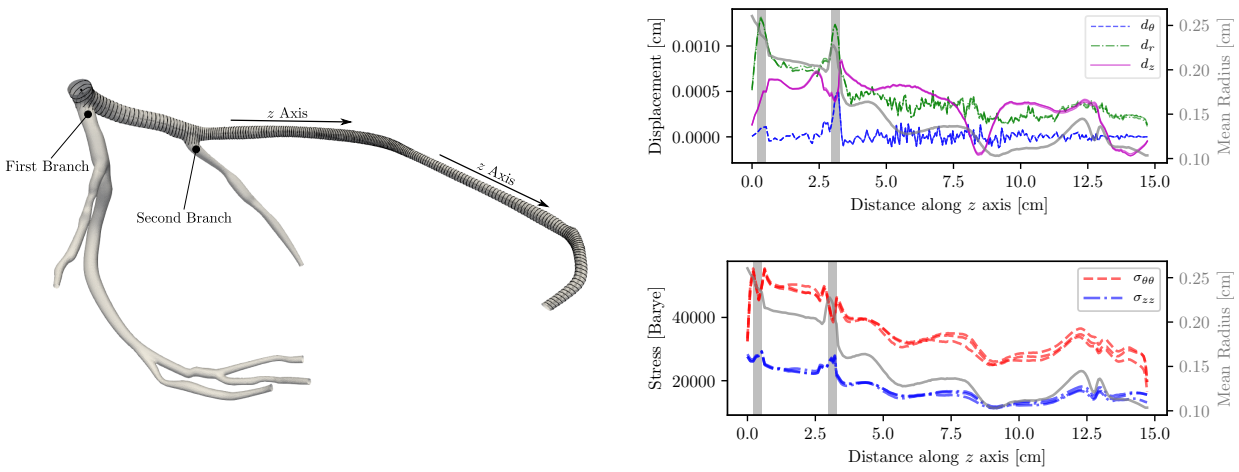


Figure 12: Displacement and stress profiles in left coronary artery LAD branch under steady flow, averaged over all material property realizations and cross-sectional slice.

3.2.2 Validation under pulsatile flow conditions

The same geometry analyzed in the previous section is subject to a parabolic pulsatile inflow shown in Figure 13. A time step equal to 2.0×10^{-6} is selected, and the model is run for two complete heart cycles (1.6 seconds) and 100 material property realizations. Similar to the pulsatile cylindrical test case, a time ramp is applied during the first 0.2 s, to avoid impulsive loads produced by a non-zero wall stress at $t = 0$, and a pressure of 13 mmHg is superimposed to the fluid wall stress. A viscous force $\boldsymbol{f}_v$ was applied, using a damping coefficient equal to $c_d = 0.005$, which was found from various tests to remove the high-frequency oscillations without affecting the system dynamics. Results for the average displacements and stress are shown in Figure 13 at four successive locations over the center path of the LAD branch. Even for this case the circumferential and axial stress are the most relevant components and their time history is similar to the inflow and exhibit a maximum at diastole.

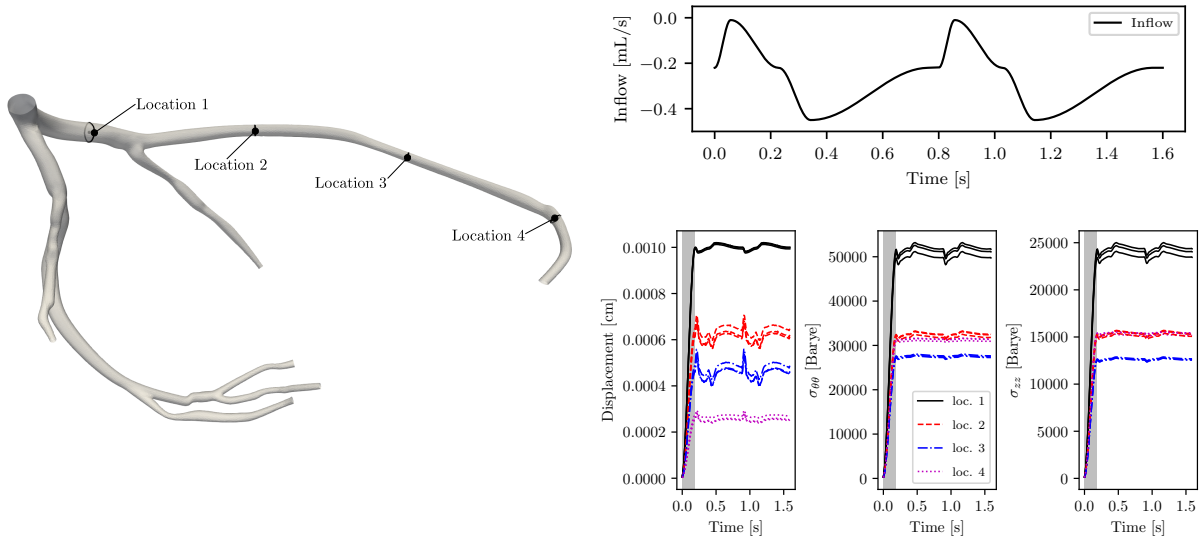


Figure 13: Displacement and stress profiles in left coronary artery LAD branch under pulsatile flow, averaged over all material property realizations and cross-sectional slice.

Table 1: Speedup - Model with 5074 elements, 2565 nodes - 1000 time steps with $\Delta t = 1.0 \times 10^{-5}$

Hardware	1 Smp (Spd)	10 Smp (Spd)	50 Smp (Spd)	100 Smp (Spd)	200 Smp (Spd)	500 Smp (Spd)
1 CPU	0:00:16 (1.0,1.0)	0:01:54 (1.43,1.0)	0:09:46 (1.39,1.0)	0:18:53 (1.44,1.0)	0:37:20 (1.46,1.0)	1:40:13 (1.36,1.0)
12 CPU	0:00:10 (1.0,1.50)	0:00:18 (5.92,6.20)	0:01:00 (8.92,9.62)	0:01:58 (9.21,9.59)	0:03:45 (9.65,9.94)	0:09:35 (9.46,10.45)
24 CPU	0:00:06 (1.0,2.35)	0:00:17 (3.93,6.45)	0:01:21 (4.27,7.22)	0:02:18 (5.03,8.21)	0:05:25 (4.26,6.87)	0:11:38 (4.98,8.61)
1 GPU	0:00:01 (1.0,9.42)	0:00:01 (10.20,67.19)	0:00:02 (29.82,201.96)	0:00:05 (32.75,214.30)	0:00:09 (37.84,244.67)	0:00:20 (42.31,293.77)
2 GPU	0:00:01 (1.0,10.56)	0:00:01 (9.17,67.69)	0:00:02 (36.27,275.28)	0:00:03 (44.01,322.76)	0:00:05 (55.86,404.83)	0:00:11 (66.74,519.27)
3 GPU	0:00:01 (1.0,9.49)	0:00:01 (9.53,63.25)	0:00:01 (44.86,305.92)	0:00:02 (59.66,393.12)	0:00:04 (75.29,490.21)	0:00:09 (91.75,641.43)
4 GPU	0:00:01 (1.0,9.34)	0:00:01 (9.70,63.35)	0:00:01 (46.12,309.52)	0:00:02 (58.67,380.35)	0:00:04 (75.09,481.08)	0:00:09 (90.21,620.49)

(*) All time entries are in a *days:hours:minutes:seconds* format. The speedup is indicated as (x, y), where x is the speedup with respect to the number of samples and y the speedup by distributing the computation on multiple CPUs or GPUs.

Table 2: Speedup - Model with 15136 elements, 7628 nodes - 1000 time steps with $\Delta t = 1.0 \times 10^{-5}$

Hardware	1 Smp (Spd)	10 Smp (Spd)	50 Smp (Spd)	100 Smp (Spd)	200 Smp (Spd)	500 Smp (Spd)
1 CPU	0:00:37 (1.0,1.0)	0:05:24 (1.16,1.0)	0:25:33 (1.23,1.0)	0:52:29 (1.20,1.0)	2:03:22 (1.02,1.0)	4:48:25 (1.09,1.0)
12 CPU	0:00:10 (1.0,3.59)	0:00:21 (4.83,14.89)	0:01:50 (4.74,13.82)	0:04:05 (4.29,12.83)	0:07:23 (4.74,16.69)	0:19:19 (4.54,14.92)
24 CPU	0:00:11 (1.0,3.32)	0:00:28 (4.02,11.48)	0:01:57 (4.83,13.01)	0:03:58 (4.77,13.21)	0:07:20 (5.16,16.79)	0:18:38 (5.08,15.47)
1 GPU	0:00:03 (1.0,9.72)	0:00:03 (10.16,84.84)	0:00:07 (26.56,209.58)	0:00:13 (27.89,225.99)	0:00:24 (31.22,297.26)	0:00:56 (34.28,305.28)
1 GPU	0:00:02 (1.0,15.01)	0:00:02 (10.11,130.45)	0:00:04 (29.17,355.60)	0:00:08 (30.94,387.40)	0:00:13 (37.20,547.27)	0:00:30 (40.80,561.29)
1 GPU	0:00:02 (1.0,17.02)	0:00:02 (9.47,138.46)	0:00:03 (29.37,405.87)	0:00:06 (33.57,476.46)	0:00:10 (40.38,673.50)	0:00:24 (44.55,694.70)
1 GPU	0:00:01 (1.0,20.01)	0:00:02 (8.21,141.15)	0:00:03 (28.94,470.08)	0:00:05 (35.15,586.47)	0:00:08 (42.02,823.93)	0:00:19 (48.60,891.06)

(*) All time entries are in a *days:hours:minutes:seconds* format. The speedup is indicated as (x, y), where x is the speedup with respect to the number of samples and y the speedup by distributing the computation on multiple CPUs or GPUs.

3.3 Performance assessment

In this section, we compare the performance obtained by running the proposed ensemble solver on three cylindrical models with an increasing number of elements, as shown in Table 1, Table 2 and Table 3. The explicit time step was set to 1.0×10^{-5} s for all models, and each run consisted of 1,000 time steps. The GPUs used for these tests are four *GeForce RTX 2080 Ti* with 11GB of RAM equipped with 4352 NVIDIA CUDA Cores and connected to the server main board through PCI express ports.

Two types of speedup are investigated, the first relates to solving the same number of material property realizations on an increasing number of processors, either CPUs or GPUs, and provides an idea of the effectiveness of the various optimizations presented in Section 2.3 and Section 2.4. This speedup (first number in parenthesis) is observed to increase with the model size and the number of random field realizations. The computation/communication tradeoff and the small mesh sizes selected for these tests also justify the negligible benefits of using 24 CPU cores instead of 12. The speedup achieved by our GPU implementation is instead very relevant, i.e., approximately three orders of magnitude with respect to a single CPU implementation.

The second type of speedup quantifies the efficiency in the proposed ensemble solver, i.e., how much faster one can obtain the solution of multiple realizations by solving them at the same time, with respect to the solution of a single realization on the same hardware, multiplied by the total number of realizations. The computational savings of an ensemble solution are quantified between one and two orders of magnitude, confirming our initial claim.

4 Conclusions and future work

Our study investigates the efficiency achievable by a ensemble cardiovascular solver on modern GPU architectures. The basic methodological approaches are discussed together with details on our efforts to optimize execution on such architectures, both algorithmically and in terms of host-to-device communication. In particular, computation of local element matrices is performed directly on the GPU, and working units are used to determine displacement increments for independent material property realizations.

The main result from our analysis is the observation that explicit-in-time ensemble solvers based on matrix-vector product naturally achieve high scalability, due to their ability to generate large amount of computations and re-usable data patterns provided by independent realizations. This also suggests how ensemble cardiovascular solvers are *ideal* to efficiently generate campaigns of high-fidelity model solutions that form a pre-requisite for uncertainty quantification studies, the state-of-the-art paradigm for the analysis of cardiovascular systems, due to their ability to quantify the effects of multiple sources of uncertainty on the simulation outputs.

Table 3: Speedup - Model with 131552 elements, 65896 nodes - 1000 time steps with $\Delta t = 1.0 \times 10^{-5}$

Hardware	1 Smp (Spd)	10 Smp (Spd)	50 Smp (Spd)	100 Smp (Spd)	200 Smp (Spd)	500 Smp (Spd)
1 CPU	0:06:47 (1.0,1.0)	0:56:58 (1.19,1.0)	4:06:55 (1.38,1.0)	7:55:06 (1.43,1.0)	-	-
12 CPU	0:00:22 (1.0,18.13)	0:02:03 (1.83,27.74)	0:10:59 (1.71,22.48)	0:21:32 (1.74,22.06)	0:50:50 (1.46,1.0)	-
24 CPU	0:00:23 (1.0,17.50)	0:02:57 (1.32,19.29)	0:13:34 (1.43,18.20)	0:26:51 (1.45,17.69)	0:54:27 (1.43,0.93)	2:47:47 (1.16,1.0)
1 GPU	0:00:29 (1.0,13.93)	0:00:30 (9.74,113.79)	0:01:00 (24.17,244.68)	0:02:00 (24.33,236.96)	0:03:33 (27.48,14.32)	-
2 GPU	0:00:16 (1.0,25.45)	0:00:16 (9.95,212.32)	0:00:31 (25.24,466.69)	0:01:02 (25.54,454.35)	0:01:53 (28.31,26.95)	0:04:18 (30.99,38.94)
3 GPU	0:00:11 (1.0,37.06)	0:00:11 (9.68,300.82)	0:00:21 (25.24,679.48)	0:00:42 (25.69,665.52)	0:01:17 (28.57,39.60)	0:02:54 (31.48,57.60)
4 GPU	0:00:08 (1.0,46.39)	0:00:09 (9.52,370.13)	0:00:17 (25.33,853.57)	0:00:33 (26.47,858.07)	0:00:59 (29.47,51.13)	0:02:16 (32.24,73.82)

(*) All time entries are in a *days:hours:minutes:seconds* format. The speedup is indicated as (x, y), where x is the speedup with respect to the number of samples and y the speedup by distributing the computation on multiple CPUs or GPUs.

We have also validated our solver for the steady state and pulsatile solution of an idealized aortic flow and a patient-specific model of the left coronary artery, under vessel wall mechanical property uncertainty, modeled through a Gaussian random field approximation.

Future work will be devoted to further improve the computational efficiency of the proposed approach and to transition from a segregated to a fully coupled Arbitrary Lagrangian-Eulerian fluid-structure interaction paradigm. In addition, the proposed approach only considered uncertainties due to vessel wall thickness and elastic modulus. We plan to support uncertainty also in the boundary conditions and model geometry.

Acknowledgements

This work was supported by a National Science Foundation award #1942662 *CAREER: Bayesian Inference Networks for Model Ensembles* (PI Daniele E. Schiavazzi). This research used computational resources provided through the Center for Research Computing at the University of Notre Dame. We also acknowledge support from the open source SimVascular project at www.simvascular.org.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] F. Lindgren, H. Rue, and J. Lindström. An explicit link between gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[2] C.A. Figueroa, I.E. Vignon-Clementel, K.E. Jansen, T.J.R. Hughes, and C.A. Taylor. A coupled momentum method for modeling blood flow in three-dimensional deformable arteries. *Computer methods in applied mechanics and engineering*, 195(41-43):5685–5706, 2006.

[3] K. Bäumler, V. Vedula, A.M. Sailer, J. Seo, P. Chiu, G. Mistelbauer, F.P. Chan, M.P. Fischbein, A.L. Marsden, and D. Fleischmann. Fluid–structure interaction simulations of patient-specific aortic dissection. *Biomechanics and Modeling in Mechanobiology*, pages 1–22, 2020.

[4] H. Askes, D.C.D. Nguyen, and A. Tyas. Increasing the critical time step: micro-inertia, inertia penalties and mass scaling. *Computational Mechanics*, 47(6):657–667, 2011.

[5] S. Morlacchi, C. Chiastra, D. Gastaldi, G. Pennati, G. Dubini, and F. Migliavacca. Sequential structural and fluid dynamic numerical simulations of a stented bifurcated coronary artery. *Journal of biomechanical engineering*, 133(12), 2011.

[6] C. Chiastra, G. Dubini, and F. Migliavacca. Modeling the stent deployment in coronary arteries and coronary bifurcations. In *Biomechanics of Coronary Atherosclerotic Plaque*, pages 579–597. Elsevier, 2020.

[7] C. Chiastra, S. Morlacchi, D. Gallo, U. Morbiducci, R. Cárdenes, I. Larrabide, and F. Migliavacca. Computational fluid dynamic simulations of image-based stented coronary bifurcation models. *Journal of The Royal Society Interface*, 10(84):20130193, 2013.

[8] A. Bartezzaghi, M. Cremonesi, N. Parolini, and U. Perego. An explicit dynamics gpu structural solver for thin shell finite elements. *Computers & Structures*, 154:29–40, 2015.

[9] V. Strbac, D.M. Pierce, J. Vander Sloten, and N. Famaey. GPGPU-based explicit finite element computations for applications in biomechanics: the performance of material models, element technologies, and hardware generations. *Computer Methods in Biomechanics and Biomedical Engineering*, 20(16):1643–1657, 2017.

[10] N. Jiang and W. Layton. An algorithm for fast calculation of flow ensembles. *International Journal of Uncertainty Quantification*, 4(4):273–301, 2014.

[11] N. Jiang. A higher order ensemble simulation algorithm for fluid flows. *Journal of Scientific Computing*, 64(1):264–288, 2015.

[12] N. Jiang and W. Layton. Numerical analysis of two ensemble eddy viscosity numerical regularizations of fluid motion. *Numerical Methods for Partial Differential Equations*, 31(3):630–651, 2015.

[13] A. Takhirov, M. Neda, and J. Waters. Time relaxation algorithm for flow ensembles. *Numerical Methods for Partial Differential Equations*, 32(3):757–777, 2016.

[14] N. Jiang. A second-order ensemble method based on a blended backward differentiation formula timestepping scheme for time-dependent Navier–Stokes equations. *Numerical Methods for Partial Differential Equations*, 33(1):34–61, 2017.

[15] M. Mohebujjaman and L.G. Rebholz. An efficient algorithm for computation of MHD flow ensembles. *Computational Methods in Applied Mathematics*, 17(1):121–137, 2017.

[16] J.A. Fiordilino. Ensemble time-stepping algorithms for the heat equation with uncertain conductivity. *Numerical Methods for Partial Differential Equations*, 34(6):1901–1916, 2018.

[17] Y. Luo and Z. Wang. An ensemble algorithm for numerical solutions to deterministic and random parabolic PDEs. *SIAM Journal on Numerical Analysis*, 56(2):859–876, 2018.

[18] M. Gunzburger, N. Jiang, and Z. Wang. An efficient algorithm for simulating ensembles of parameterized flow problems. *IMA Journal of Numerical Analysis*, 39(3):1180–1205, 2019.

[19] M.L. Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[20] P. Whittle. On stationary processes in the plane. *Biometrika*, pages 434–449, 1954.

[21] P. Whittle. Stochastic-processes in several dimensions. *Bulletin of the International Statistical Institute*, 40(2):974–994, 1963.

[22] D. Bolin and F. Lindgren. A comparison between Markov approximations and other methods for large spatial data sets. *Computational Statistics & Data Analysis*, 61:7–21, 2013.

[23] Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1):173–201, 2007.

[24] K.E. Jansen, C.H. Whiting, and G.M. Hulbert. A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. *Computer methods in applied mechanics and engineering*, 190(3-4):305–319, 2000.

[25] J. Seo, D.E. Schiavazzi, and A.L. Marsden. Performance of preconditioned iterative linear solvers for cardiovascular simulations in rigid and deformable vessels. *Computational Mechanics*, pages 1–23, 2019.

[26] A. Buluç, J.T. Fineman, M. Frigo, J.R. Gilbert, and C.E. Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 233–244, 2009.

[27] M. Garland. Sparse matrix computations on manycore GPUs. In *Proceedings of the 45th annual Design Automation Conference*, pages 2–6, 2008.

[28] N. Bell and M. Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the conference on high performance computing networking, storage and analysis*, pages 1–11, 2009.

[29] J.L. Greathouse and M. Daga. Efficient sparse matrix-vector multiplication on GPUs using the CSR storage format. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 769–780. IEEE, 2014.

[30] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31–39, march-april 2011.

[31] J.E. Stone, D. Gohara, and G. Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66–73, 2010.

[32] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih. Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

[33] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit (4th ed.)*. Kitware, 2006.

[34] H. Lan, A. Updegrove, N.M. Wilson, G.D. Maher, S.C. Shadden, and A.L. Marsden. A re-engineered software interface and workflow for the open-source simvascular cardiovascular modeling package. *Journal of biomechanical engineering*, 140(2), 2018.

[35] George Karypis and Vipin Kumar. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. http://www.cs.umn.edu/~metis, 2009.

[36] J.S. Tran, D.E. Schiavazzi, A.M. Kahn, and A.L. Marsden. Uncertainty quantification of simulated biomechanical stimuli in coronary artery bypass grafts. *Computer Methods in Applied Mechanics and Engineering*, 345:402–428, 2019.