An ensemble solver for segregated cardiovascular FSI

Xue Li · Daniele E. Schiavazzi

Received: 21 January 2021 / Accepted: 3 August 2021 / Published online: 18 August 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

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.

Keywords 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 Fig. 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. 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 [25]. Additionally, we follow a one-way coupled, segregated approach to fluid-structure interaction, where the wall stress is computed by an implicit variational multiscale (VMS) fluid solver.
and passed to a structural, three-d.o.f.s shell model of the vascular walls [12]. 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 reducing 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 [4]).

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 $\Delta t$ is the amount of time required by an elastic wave to travel across the smallest element in the mesh. A reasonable value of $\Delta t$ 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, $c = \sqrt{E/\rho}$ (we disregard the Poisson effect for the sake of the argument) is the elastic wave speed, $E$ and $\rho$ the elastic modulus and density of the vascular tissue (assumed homogeneous and isotropic in the current argument). Typical values of $l_{e, \text{min}} = 1.0 \cdot 10^{-3}$ m, $E = 0.7$ MPa and $\rho = 1.06$ kg/m³ lead to a stable time step equal to approximately $0.9 \cdot 1.2 \cdot 10^{-6} \approx 1.0 \cdot 10^{-6}$. 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., [1]), 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 [11,28], and coupling with an implicit fluid solver is discussed in [10]. Implementation of structural explicit solvers on GPU are discussed in various studies in the literature. In [3] the authors describe in detail an application involving thin shells, while an overview on applications in biomechanics is discussed in [33]. Additionally, ensemble methods for fluid problems have been recently proposed by [18–21,34] in the context of the Navier–Stokes equations with distinct initial conditions and forcing terms. This is based on the observation that the solution of linear systems is responsible for a significant fraction of the overall running time for 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 [27], natural convection problems [13] and parametrized flow problems [16,26]. 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 Sect. 2, including the generation of random field material properties, the structural finite element formulation, the variational multiscale fluid solver, their weak coupling and CPU/GPU implementation.
Validation of the proposed approach is discussed in Sect. 3 with reference to an idealized model of the thoracic aorta and a patient-specific coronary model. The performance and scalability of the approach are discussed in Sect. 3.3, followed by conclusions and future work in Sect. 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 [31]. Let \( \| \cdot \| \) denote the Euclidean distance in \( \mathbb{R}^d \). The Matérn covariance between two points at distance \( \| h \| \) is

\[
r(\| h \|) = \frac{\sigma^2}{2^{v-1} \Gamma(v)} \left( \frac{\rho}{\kappa} \| h \| \right)^v K_v \left( \frac{\rho}{\kappa} \| h \| \right), \quad h \in \mathbb{R}^d
\]

where \( \Gamma(\cdot) \) is the gamma function, \( K_v \) is the modified Bessel function of the second kind, \( \sigma^2 \) is the marginal variance, \( v \) is a scaling parameter which determines the mean spatial differentiability of the underlying process, and \( \rho \) is related to the correlation length \( \rho = \sqrt{8v/\kappa} \), i.e., the distance which corresponds to a correlation of approximately 0.1, for all \( v \).

It is known from the literature [36, 37] how Gaussian random fields with Matérn covariance can be obtained as solutions of a linear fractional stochastic partial differential equation (SPDE) of the form

\[
(\kappa^2 - \Delta)^{\alpha/2} x(s) = \mathcal{W}(s), \quad s \in \mathbb{R}^d,
\]

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

\[
\sigma^2 = \frac{\Gamma(v)}{\Gamma(v + d/2)(4\pi)^{d/2} \kappa^{2v}}.
\]

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 [25]. We construct a discrete approximation of the solution \( x(s) \) using a linear combination of basis functions, \( \{ \psi_k \}, k = 1, \ldots, n \), and appropriate weights, \( \{ w_k \}, k = 1, \ldots, n \), i.e., \( x(s) = \sum_{k=1}^n \psi_k(s) w_k \). We then introduce an appropriate Sobolev space with inner product \( \langle \cdot, \cdot \rangle \), a family of test functions \( \{ \psi_k \}, k = 1, \ldots, n \), and derive a Galerkin functional for (2) of the form

\[
\langle \psi_i, (\kappa^2 - \Delta)^{\alpha/2} \psi_j \rangle w_j = \langle \psi_i, \mathcal{W} \rangle
\]

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

\[
Q_\alpha = \kappa^2 C + G \quad \text{for } \alpha = 1
\]

\[
Q_\alpha = (\kappa^2 C + G)^T C^{-1} (\kappa^2 C + G) \quad \text{for } \alpha > 2,
\]

where, for \( \alpha \geq 3 \) a recursive Galerkin formulation is used, letting \( \alpha = 2 \) on the left-hand side of Eq. (2) and replacing the right-hand side with a field generated by \( \alpha - 2 \), assigning \( \psi_k = \psi_k \), \( k = 1, \ldots, n \). Note how the use of piecewise linear basis functions \( \{ \psi_k \}, k = 1, \ldots, n \), lead to matrices

\[
G_{ij} = \langle \nabla \psi_i, \nabla \psi_j \rangle \quad \text{and} \quad C_{ij} = \langle \psi_i, \psi_j \rangle,
\]

that are sparse, and often found in the finite element discretization of second order elliptic PDEs. However, the precision matrices \( Q_\alpha \) are, in general, not sparse as they contain the inverse \( C^{-1} \). Thus, by replacing the matrix \( C \) with the lumped diagonal matrix \( C \), sparsity is restored and \( Q_\alpha \) 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 \( C \) leads to non zero terms on each row only for the immediate neighbors \( \mathcal{F}(s_k) \) 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

\[
\rho \left( x(s_i), x(s_k) \mid x(s_j), s_j \in \mathcal{F}(s_i) \right)
\]

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 [8]. The interested reader is referred to [25] for additional detail on the derivation of \( Q_\alpha \).

Numerically generated realizations for various correlation lengths are shown in Fig. 2 on an ideal cylindrical representation of the descending thoracic aorta, while Fig. 3 shows the agreement of the generated field with the 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 [12]. 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

$$\sigma^l = C \cdot \varepsilon^l, \quad \text{with} \quad \sigma^l = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix},$$

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

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

(9)

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

$$k_e = \int_{\Omega_e} B^T C B \, d\Omega_e = \sum_{i=1}^{n_{gp}} B^T C B A_e \xi_i w_i,$$

(10)

where $n_{gp}$ is the total number of integration points and $\xi_i, w_i, i \in \{1, 2, \ldots, n_{gp}\}$ are the element thickness and
of (12) in weak form requires to define four approximation spaces, i.e., two trial spaces for the velocity $v$ and pressure $p$

$$ \mathcal{X}_h^p = \left\{ v \mid v(\cdot, t) \in H^1(\Omega), t \in [0, T], \right. $$

$$ \left. v|_{x \in \Omega_e} \in P_k(\Omega_e), \, v(\cdot, t) = g \text{ on } \Gamma_f \right\}, $$

and two test spaces for $w$ and $q$

$$ \mathcal{Y}_h^w = \left\{ w \mid w(\cdot, t) \in H^1(\Omega), t \in [0, T], \right. $$

$$ \left. w|_{x \in \Omega_e} \in P_k(\Omega_e), \, w(\cdot, t) = 0 \text{ on } \Gamma_f \right\}, \ \mathcal{Z}_h^w = \mathcal{Y}_h^w, $$

where $H^1(\Omega)$ 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 $\Omega$. The above spaces are separated into large and a small scale contributions $\mathcal{X}_h^p = \mathcal{X}_h^p \oplus \mathcal{Z}_h^p$, $\mathcal{Y}_h^w = \mathcal{Y}_h^w \oplus \mathcal{Z}_h^w$ and $\mathcal{Z}_h^w = \mathcal{Z}_h^w \oplus \mathcal{Z}_h^w$ with a corresponding decomposition of velocity and pressures as $v = \tilde{v} + \bar{v}$ and $p = \tilde{p} + \bar{p}$, 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 $\tilde{v}$ and $\tilde{p}$ in terms of their large scale counterparts $\bar{v}$ and $\bar{p}$ using [5]

\[ \begin{bmatrix} \bar{v} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \tau_M R_M(\bar{v}, \bar{p}) \\ \tau_C R_C(\bar{v}, \bar{p}) \end{bmatrix}, \]

where $R_M$ and $R_C$ represent the momentum and continuity residual expressed as

\[ \begin{bmatrix} \bar{v} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \bar{v} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \bar{v} + \bar{v} \cdot \nabla \bar{v} + \nabla \bar{p} - \nabla \cdot \tau - f, \\ \bar{v} \cdot \nabla \bar{v} = 0 \end{bmatrix}, \]

and the stabilization coefficients are expressed as

\[ \tau_M = \left( \frac{4}{\Delta t^2} + u \cdot G + C_1 \nu^2 : G \right)^{-1/2}, \]
\[ \tau_C = (\tau_M g : g)^{-1}, G_{i,j} = \sum_{k=1}^{3} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j}, \]

where $\partial \xi / \partial x$ 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 $\bar{v}$ and $\bar{p}$ will be denoted simply by $v$ and $p$ and the spaces $\mathcal{X}_h^p$, $\mathcal{Y}_h^w$ and $\mathcal{Z}_h^w$ by $\mathcal{X}_h^p$, $\mathcal{Y}_h^w$ and $\mathcal{Z}_h^w$. A weak solu-
tion of the Navier–Stokes equations can now be determined by finding $v \in \mathcal{V}^k_h$ and $p \in \mathcal{P}^k_h$ such that

$$B(w, q; v, p) = B_G(w, q; v, p)$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega_e} (v \cdot \nabla) w \cdot (\tau_M R_M) + \nabla \cdot w \tau_C R_C \, d\Omega_e$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega_e} [w \cdot [-\tau_M R_M \cdot \nabla] + [R_M \cdot \nabla w] \cdot [\tau R_M \cdot v]] \, d\Omega_e$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega_e} \nabla q \cdot \frac{\tau_M}{\rho} R_M \, d\Omega_e = 0,$$

(17)

for all $w \in \mathcal{W}^k_h$ and $q \in \mathcal{P}^k_h$, where the Galerkin functional $B_G$ is expressed as

$$B_G(w, q; v, p) = \int_{\Omega} w \cdot (\rho \dot{v} + \rho v \cdot \nabla v - f) \, d\Omega$$

$$+ \int_{\Omega} (\nabla w : (\rho I + \tau) - \nabla q \cdot v) \, d\Omega$$

$$+ \int_{\Gamma_h} [-w \cdot h + q v_n] \, d\Gamma + \int_{\Gamma_s} [-w \cdot t' + q v_n] \, d\Gamma$$

$$+ \int_{\Gamma_\infty} q v_n \, d\Gamma.$$  

(18)

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

$$\begin{align*}
N_M(v^h, q^h, p^h) &= 0 \\
N_C(v^h, q^h, p^h) &= 0.
\end{align*}$$  

(19)

A predictor-multicorrector scheme [17] 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 [5,30].

### 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 Sect. 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 $t^f = -t^f$, where the wall stress $t^f$ computed by the fluid solver is

$$t^f = \sigma^f \cdot n, \quad \sigma = 2\mu \varepsilon - \rho I,$$

(20)

$p$ is the main nodal pressure unknown in the VMS fluid solver, and $\varepsilon = \nabla \cdot u$ 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 Fig. 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.

Stress components in the circumferential, radial and axial directions are obtained by transforming the solid stress tensor to a local, vessel-based, cylindrical coordinate system (see Fig. 4). For an arbitrary Gauss point or lumen shell node $s$, we identify the closest location on the vessel centerline. The tangent vector to the centerline at $s$ 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 $\theta$ is obtained as the cross product between $r$ and $z$.

Finally, we would like to point out how the storage of fluid stress at the lumen and its transfer to the solid mesh is the only operation whose implementation is specific to the one-way coupled FSI nature of the solver discussed in the paper. All other features of our solver are implemented with the goal of developing a general, strongly-coupled, FSI solver.

### 2.3 Distributed explicit FSI solver on multiple CPUs

The solution in time for the dynamics of the undamped nonlinear structural system

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

(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
\[
\left( \tilde{M} + \frac{\Delta t}{2} \tilde{C} \right) u_{n+1} = \Delta t^2 f_n - \left( \Delta t^2 K - 2 \tilde{M} \right) u_n - \left( M - \frac{\Delta t}{2} \tilde{C} \right) u_{n-1}
\]  

(22)

which is performed independently by an arbitrary number of mesh partitions. To do so efficiently, \( \tilde{M}, \tilde{C} \) in (22) are 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 the nodal masses at each time step, improving scalability. In some cases, a viscous force \( f_v \) is added to the right-hand-side in order to damp the high-frequency oscillations as \( f_v = -c_d \tilde{u}_n \), through an appropriate damping coefficient \( c_d \).

Even though the geometry of the vessel lumen is updated at every step by adding \( u_{n+1} \), 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 [9] with dense coefficient entries of size \( 9 \cdot n_s \), where \( n_s \) is the number of material property realizations, and a local nodal matrix of size \( 3 \times 3 \) 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 Figs. 5 and 6.

### 2.4 Distributed explicit FSI solver on multiple GPUs

We also 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 [14]. 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 the others to finish. We then transitioned to a CSR-Vector scheme [7], 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 [15] 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 Fig. 5b. This problem can be mitigated by sending data to the GPU in smaller chunks, thus overlapping data transfer and kernel execution.

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. The local matrices for each realization are stored in private memory since they do not need to be shared with other working units, while displacements are stored in the global memory, being results that need to be copied back to the CPU for output. Final GPU speed ups are illustrated in Fig. 7.

Simulation results for selected time steps are saved in a single VTK file, assembled from the various partitions after synchronization in the root CPU. Each file stores the model results in terms of displacement and stress fields, either in the global Cartesian or a vessel-based cylindrical coordinate system. From this format, the spatial distribution of the statistical moments for any result quantity can be directly computed in Paraview [2] or through the VTK APIs [29]. No investigation has been made so far on techniques to compress the ensemble solver result files. This surely represents an interesting direction for future work.

2.5 A Python code-base

The CVFES solver is developed in Python 3 with optimization in Cython [6] and element assembly and matrix product implementation on openCL [32] (though the Python pyOpenCL library [23]). The code leverages the VTK library [29] 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 [24] and can be easily integrated with the SimVascular modeling workflow. Partitioning on multiple CPUs and GPUs is obtained for both solvers using parMETIS [22]. The code used to generate the results discussed in Sect. 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 Sect. 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 [35]). A uniform pressure of 13 mmHg was superimposed to the loads computed by the VMS fluid solver (with a traction-free outlet boundary condition) to account for the residual stresses. We assume the vessel wall to be at its natural (unstressed) state at diastole, and have applied an additional pressure equal to the difference between a diastolic value 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 Sect. 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 Fig. 8.

The steady state pressure resulting from the fluid solver is applied as discussed in Sect. 2.2.3 and 100 thickness and elastic modulus realizations are solved simultaneously. The explicit structural simulation is run for 0.5 s, until a steady state was observed. Three wall mesh densities (coarse, medium and fine) are finally considered, consisting of 5074, 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 Fig. 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 Fig. 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 Fig. 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 $\sigma_{rz}$), 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 aver-
were kept the same as in the steady case. The time step is set while all the other boundary conditions (walls and outlets) same cylindrical geometry discussed in the previous section, variational multi-scale fluid solver was applied to the simulation, i.e., \( f_v \) is kept active for the first 0.2 s, to avoid impulsive loads produced by a non-zero wall stress at \( t = 0 \). 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 loads from the fluid solver. The ramp follows an infinitely differentiable sinusoidal wave, and is kept active for the first 0.2 s of the simulation. No damping was applied to the simulation, i.e., \( f_v = 0 \).

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 dominant, and they also increase with the correlation length, as observed for the steady state results.

### 3.1.2 Validation under pulsatile flow conditions

A parabolic pulsatile inflow (Fig. 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 \( 1.0 \times 10^{-5} \) and the simulation run for two heart cycles and 100 material property realizations. As per the segregated FSI approach, our structural model is subject to either the steady state or the periodic pulsatile loads from the fluid, with a non-zero pressure and shear force applied at \( t = 0 \). 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 loads from the fluid solver. The ramp follows an infinitely differentiable sinusoidal wave, and is kept active for the first 0.2 s of the simulation. No damping was applied to the simulation, i.e., \( f_v = 0 \).

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 dominant, and they also increase with the correlation length, as observed for the steady state results.

### 3.2 Benchmark on a 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 \( \mu = 1.15 \times 10^3 \) 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 Sect. 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 \( 2.5 \times 10^{-6} \) and the model was run for 0.13 s to reach the steady state. No additional viscous force \( f_v \) 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.

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.

#### 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 Fig. 13. A time step equal to \( 2.0 \times 10^{-6} \) is selected, and the model is run for two complete heart cycles (1.6 s) and 100 material property realizations. 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.
Fig. 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)

Fig. 10  Ensemble means (thick black line), 5–95% confidence intervals (shaded gray area) and single realizations of longitudinal stress profiles for various mesh densities and random field correlation lengths
Fig. 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).

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

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 Tables 1, 2 and 3. The explicit time step was set to $1.0 \times 10^{-5}$ s for all models, and each run consisted of 1000 time steps.

The GPUs used for these tests are four GeForce RTX 2080 Ti with 11 GB 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 Sects. 2.3 and 2.4. This speedup (second 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 (first number in parenthesis) 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 an 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.

We discuss two different perspectives to improve the efficiency of numerical simulations for cardiovascular flow. We first discuss our development process in going from a CPU to a GPU implementation of an explicit structural solver for soft tissue biomechanics, and show that a GPU implementation is more efficient by a very significant margin. While efficient implementation of explicit structural solvers on GPUs have already been the subject of multiple contributions in the literature, we consider this problem in the unique context of an ensemble solution of multiple model instances. The availability of multiple material property realizations allow a single work-group to compute the stiffness matrix of one finite element for all material property realizations, enabling coalesced memory access. This further increase the efficiency gap between CPU and GPU implementations.

Second, we look at the most efficient way to upscale the efficient GPU solution for multiple model instances. It is evident that an embarrassingly parallel solution strategy is typically not viable for GPU solvers. For 500 (say) material property realizations, while the use of 500 CPU cores is not uncommon for large cardiovascular simulations using private or federal institutional resources, an embarrassingly
### Table 2: Speedup—model with 15,136 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.

### Table 3: Speedup—model with 131,552 elements, 65,896 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 (0.74, 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, 269.5) | 0:04:18 (30.99, 389.94) |
| 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.
parallel GPU solution would require 500 GPUs, i.e., a substantial amount. It is therefore evident that ensemble solvers are the way to go, and we investigate the scalability of such a paradigm by determining how much time the ensemble solution of 500 samples would require, with respect to the independent solution of 1 sample 500 times, using the same number of GPUs.

The main result from our analysis is the observation that explicit-in-time ensemble solvers based on matrix-vector product naturally achieve superior scalability, due to their ability to generate large amount of computations and reusable 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 (a 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.

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 fluid solver 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.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Askes H, Nguyen D, Tyas A (2011) Increasing the critical time step: micro-inertia, inertia penalties and mass scaling. Comput Mech 47(6):657–667
2. Ayachit U (2015) The paraview guide: a parallel visualization application. Kitware Inc, New York
3. Bartezzagh A, Cremonesi M, Parolini N, Perego U (2015) An explicit dynamics GPU structural solver for thin shell finite elements. Comput Struct 154:29–40
4. Bäumler K, Vedula V, Sailer A, Seo J, Chiu P, Mistelbauer G, Chan F, Fischbein M, Marsden A, Fleischmann D (2020) Fluid-structure interaction simulations of patient-specific aortic dissection. Biomech Model Mechanobiol 19:1602–1628
5. Bazilevs Y, Calv O, Cottrell J, Hughes T, Reali A, Scovazzi G (2007) Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Comput Methods Appl Mech Eng 197(1):173–201
6. Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn D, Smith K (2011) Cython: the best of both worlds. Comput Sci Eng 13(2):31–39. https://doi.org/10.1109/MCSE.2010.118
7. Bell N, Garland M (2009) Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: Proceedings of the conference on high performance computing networking, storage and analysis, pp 1–11
8. Bolin D, Lindgren F (2013) A comparison between Markov approximations and other methods for large spatial data sets. Comput Stat Data Anal 61:7–21
9. Buluç A, Fineman J, Frigo M, Gilbert J, Leiserson C (2009) 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, pp 233–244
10. Chiastra C, Morlacchi S, Gallo D, Morbiducci U, Cárdenes R, Larrabide I, Migliavacca F (2013) Computational fluid dynamic simulations of image-based stented coronary bifurcation models. J R Soc Interface 10(84):20130193
11. Chiastra C, Dubini G, Migliavacca F (2020) Modeling the stent deployment in coronary arteries and coronary bifurcations. In: Biomechanics of coronary atherosclerotic plaque. Elsevier, pp 579–597
12. Figueroa C, Vignon-Clementel I, Jansen K, Hughes T, Taylor C (2006) A coupled momentum method for modeling blood flow in three-dimensional deformable arteries. Comput Methods Appl Mech Eng 195(41–43):5685–5706
13. Fiordilino J (2018) Ensemble time-stepping algorithms for the heat equation with uncertain conductivity. Numer Methods Partial Differ Equ 34(6):1901–1916
14. Garland M (2008) Sparse matrix computations on manycore GPUs. In: Proceedings of the 45th annual design automation conference, pp 2–6
15. Greathouse J, Daga M (2014) 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. IEEE, pp 769–780
16. Gunzburger M, Jiang N, Wang Z (2019) An efficient algorithm for simulating ensembles of parameterized flow problems. IMA J Numer Anal 39(3):1180–1205
17. Jansen K, Whiting C, Hulbert G (2000) A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. Comput Methods Appl Mech Eng 190(3–4):305–319
18. Jiang N (2015) A higher order ensemble simulation algorithm for fluid flows. J Sci Comput 64(1):264–288
19. Jiang N (2017) A second-order ensemble method based on a blended backward differentiation formula time-stepping scheme for time-dependent Navier–Stokes equations. Numer Methods Partial Differ Equ 33(1):34–61
20. Jiang N, Layton W (2014) An algorithm for fast calculation of flow ensembles. Int J Uncertain Quantif 4(4):273–301
21. Jiang N, Layton W (2015) Numerical analysis of two ensemble eddy viscosity numerical regularizations of fluid motion. Numer Methods Partial Differ Equ 31(3):630–651
22. Karypis G, Kumar V (2009) MeTis: unstructured graph partitioning and sparse matrix ordering system, Version 4.0. http://www.cs.umn.edu/~metis
23. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A (2012) Pycuda and pyopencl: a scripting-based approach to GPU run-time code generation. Parallel Comput 38(3):157–174
24. Lan H, Updegrove A, Wilson N, Maher G, Shadden S, Marsden A (2018) A re-engineered software interface and workflow for the open-source simvascular cardiovascular modeling package. J Biomech Eng 140(2):0245011
25. Lindgren F, Rue H, Lindström J (2011) An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. J R Stat Soc Ser B (Stat Methodol) 73(4):423–498. https://doi.org/10.1111/j.1467-9868.2011.00777.x
26. Luo Y, Wang Z (2018) An ensemble algorithm for numerical solutions to deterministic and random parabolic PDEs. SIAM J Numer Anal 56(2):859–876
27. Mohebujjaman M, Rebholz L (2017) An efficient algorithm for computation of MHD flow ensembles. Comput Methods Appl Math 17(1):121–137
28. Morlacchi S, Chiastra C, Gastaldi D, Pennati G, Dubini G, Migliavacca F (2011) Sequential structural and fluid dynamic numerical simulations of a stented bifurcated coronary artery. J Biomech Eng 133(12):121010
29. Schroeder W, Martin K, Lorensen B (2006) The visualization toolkit, 4th edn. Kitware, New York
30. Seo J, Schiavazzi D, Marsden A (2019) Performance of preconditioned iterative linear solvers for cardiovascular simulations in rigid and deformable vessels. Comput Mech. https://doi.org/10.1007/s00466-019-01678-3
31. Stein M (2012) Interpolation of spatial data: some theory for kriging. Springer, Berlin
32. Stone J, Gohara D, Shi G (2010) Opencl: a parallel programming standard for heterogeneous computing systems. Comput Sci Eng 12(3):66–73
33. Strbac V, Pierce D, Vander Sloten J, Famaey N (2017) GPGPU-based explicit finite element computations for applications in biomechanics: the performance of material models, element technologies, and hardware generations. Comput Methods Biomech Biomed Eng 20(16):1643–1657
34. Takhirov A, Neda M, Waters J (2016) Time relaxation algorithm for flow ensembles. Numer Methods Partial Differ Equ 32(3):757–777
35. Tran J, Schiavazzi D, Kahn A, Marsden A (2019) Uncertainty quantification of simulated biomechanical stimuli in coronary artery bypass grafts. Comput Methods Appl Mech Eng 345:402–428. https://doi.org/10.1016/j.cma.2018.10.024
36. Whittle P (1954) On stationary processes in the plane. Biometrika 41:434–449
37. Whittle P (1963) Stochastic-processes in several dimensions. Bull Int Stat Inst 40(2):974–994

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.