Computer Physics Communications 181 (2010) 462–472

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Hydrokinetic approach to large-scale cardiovascular blood flow

Simone Melchionna [a,b,*], Massimo Bernaschi [c], Sauro Succi [c,d], Efthimios Kaxiras [e], Frank J. Rybicki [f], Dimitris Mitsouras [f], Ahmet U. Coskun [g], Charles L. Feldman [h]

[a] SOFT-INFM, CNR, Rome, Italy
[b] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
[c] Istituto Applicazioni Calcolo, CNR, Rome, Italy
[d] Initiative in Innovative Computing, Harvard University, Cambridge, MA, USA
[e] Department of Physics and School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
[f] Applied Imaging Science Laboratory, Department of Radiology, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA
[g] Northeastern University, Department of Mechanical and Industrial Engineering, Boston, MA, USA
[h] Cardiovascular Division, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA

ARTICLE INFO

Article history:
Received 11 June 2009
Received in revised form 23 September 2009
Accepted 19 October 2009
Available online 24 October 2009

Keywords:
Lattice Boltzmann
Computational hemodynamics
Atherosclerosis
Accelerator-based supercomputing

ABSTRACT

We present a computational method for commodity hardware-based clinical cardiovascular diagnosis based on accurate simulation of cardiovascular blood flow. Our approach leverages the flexibility of the Lattice Boltzmann method to implementation on high-performance, commodity hardware, such as Graphical Processing Units. We developed the procedure for the analysis of real-life cardiovascular blood flow case studies, namely, anatomic data acquisition, geometry and mesh generation, flow simulation and data analysis and visualization. We demonstrate the usefulness of our computational tool through a set of large-scale simulations of the flow patterns associated with the arterial tree of a patient which involves two hundred million computational cells. The simulations show evidence of a very rich and heterogeneous endothelial shear stress pattern (ESS), a quantity of recognized key relevance to the localization and progression of major cardiovascular diseases, such as atherosclerosis, and set the stage for future studies involving pulsatile flows.

1. Introduction

The human cardiovascular system is a very complex physiological network: blood is pumped by the heart through the large arteries to the smaller diameter arterioles, then through capillaries and eventually to the venules, where the deoxygenated blood is passed through veins back to the heart, imposing a circular pattern through the whole body. Atherosclerosis is the most common cardiovascular disease, primarily affecting the arterial blood vessels; the resulting coronary heart disease is the most common cause of mortality and morbidity in developed countries, responsible for $\sim 35\%$ of annual deaths, about half of which occur suddenly and with no prior symptoms [1]. Although the development of atherosclerosis depends on the presence of systemic risk factors, such as high cholesterol, diabetes and high blood pressure, the clinical manifestations of the disease – heart attack, sudden coronary death and angina pectoris – are focal, resulting from the accumulation of lipid molecules and inflammatory cells at specific locations within the wall of the coronary arteries. Prior research, observational *in vitro* and *in vivo*, has found that the foci of atherosclerosis appear in regions of disturbed blood flow, where the local endothelial shear stress (ESS) is low (< 1.0 Pa) or of alternating direction [2]. Therefore, atherosclerotic lesions frequently form near arterial branches and bifurcations, where flow is always disturbed as compared to un-branched regions [3,4]. In those sites, the endothelial cell morphology is modified in a way to favor the adhesion of leukocytes on the surface wall and the uptake of lipoproteins.

The evidence for the key role of low average ESS in the localization and progression of atherosclerosis is compelling and widely accepted [2,3,5,6], although steady low ESS and oscillatory ESS with a mean close to zero seem to elicit somewhat different biological responses [7]. To date however, there is no direct method to predict the occurrence of atherosclerosis and no non-invasive method for measuring ESS in human coronary arteries *in vivo*. Therefore, predictions of where disease is likely to develop and what form it would take (that is, stable vs. unstable) in coronary arteries were previously limited. As previous work dating back about a decade has demonstrated [6,8,9], a viable alternative is offered by the combined use of 3D reconstruction techniques and fluid-dynamics simulation methods.

This work is part of a systematic effort aimed at simulating coronary endothelial shear stress using Multi-Detector Computed

* Corresponding author at: SOFT-INFM, CNR, Rome, Italy.
E-mail address: simone.melchionna@roma1.infn.it (S. Melchionna).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2009.10.017

Tomography (MDCT). Coronary CT angiography is an emerging, non-invasive imaging modality to assess the coronary artery lumen anatomy. This technique also gives information regarding coronary wall and plaque morphology; the spatial and temporal resolution of current systems have improved from earlier technologies. Newest MDCT systems with 320-detector rows [10,11] enable 3D acquisition of the entire coronary arterial tree in a single heart beat with radiation doses comparable to coronary catheterization.

The main objective of the computational approach to hemodynamics described below is to develop and deploy a general purpose, non-invasive, methodology to routinely study blood flow patterns *in vivo* in human coronary arteries. Another crucial goal is to design a non-invasive, practical tool capable of eventually determining which regions within the coronary arteries might be at risk of rapid progression to thin cap fibroatheromas and possible plaque rupture. A direct benefit of this approach would be the enhanced biomedical understanding of the causes and evolution of plaques in the body arteries, with important implications for predicting the course of atherosclerosis and possibly preventing or mediating its effects.

In a previous companion paper [12], we have discussed the main features of the code MUPHY, a general-purpose software for the simulation of complex flows. In this work, we present a major extension which allows MUPHY to incorporate the full sequence of steps entailed by a complete cardiovascular analysis of real-life patients, namely, medical data acquisition, geometry import and mesh generation, flow simulation, data analysis and visualization. The capabilities of the resulting computational tool are demonstrated through a set of large-scale simulations (performed on grids with ∼two hundred million computational cells) of the arterial tree of a real patient, involving a complex, multi-branched, geometry.

Our simulations of blood flow are based on the Lattice Boltzmann (LB) method, which is particularly efficient and flexible in handling complex arterial geometries. In the past, the LB method has been applied to a broad range of fluid-dynamic problems, including turbulence [13] and multiphase flows [14], as well as in blood-flow simulations with elastic boundaries [15], steady and pulsatile flows [16–21] and flows with complex boundaries [22]. The joint use of simulation and imaging techniques presented here could allow to non-invasively and inexpensively screen large numbers of patients for incipient coronary disease, and to intervene at clinical level prior to the occurrence of a catastrophic event.

The rest of the paper is organized as follows: Section 2 presents the LB method as implemented here for complex flows; Section 3 discusses some details of our MUPHY code; Section 4 presents results for a real arterial tree of a patient; the final Section 5 contains some concluding remarks.

2. Lattice Boltzmann for hemodynamic flows

In the last decade, the Lattice Boltzmann (LB) method has captured increasing attention from the fluid-dynamics community as a competitive computational alternative to the discretization of the Navier–Stokes equations of continuum mechanics. LB is a minimal form of the Boltzmann kinetic equation, based on the collective dynamics of fictitious particles on the nodes of a regular lattice. The dynamics of these particles is designed in such a way as to obey the basic conservation laws ensuring hydrodynamic behavior in the continuum limit, in which the molecular mean free path is much shorter than typical macroscopic scales [23]. This condition is clearly met in blood flow, together with the Newtonian rheological behavior of blood, in large coronary systems. Non-Newtonian rheological models appropriate for simulating blood flow in medium or small-sized arteries, such as the Casson or Carreau–Yasuda models, can be also incorporated within the LB approach [24].

In the LB method, the basic quantity is $f_i(\vec{x},t)$, representing the probability of finding a "fluid particle" at the spatial mesh location $\vec{x}$ and at time t with discrete speed $\vec{c}_i$. The mesh spacing is Δx. "Fluid particles" here do not correspond to individual fluid molecules, but represent instead the collective motion of a group of physical particles (often referred to as populations). For the present study, we use the common three-dimensional 19-speed cubic lattice where the discrete velocities $\vec{c}_i$ connect mesh points to first and second topological neighbors [13]. Once the discrete populations f_i are known, the kinetic moments are obtained by a direct summation upon all discrete populations at a given lattice site, with the local density obtained as

$$\rho(\vec{x},t) = \sum_i f_i(\vec{x},t) \tag{1}$$

the flow current as

$$\rho\vec{u}(\vec{x},t) = \sum_i f_i(\vec{x},t)\vec{c}_i \tag{2}$$

and the momentum-flux tensor as

$$\overleftrightarrow{P}(\vec{x},t) = \sum_i f_i(\vec{x},t)\vec{c}_i\vec{c}_i \tag{3}$$

The fluid populations are advanced in time through the following evolution equation

$$f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{x},t) + \omega\Delta t\left(f_i - f_i^{eq}\right)(\vec{x},t) + \Delta t F_i(\vec{x},t) \tag{4}$$

The right-hand side of Eq. (4) represents the effect of fluid–fluid molecular collisions, through a relaxation towards a local equilibrium, f_i^{eq}, typically a second-order expansion in the fluid velocity of a local Maxwellian with speed $\vec{u}$,

$$f_i^{eq} = w_i\rho\left[1 + \frac{\vec{u}\cdot\vec{c}_i}{c_s^2} + \frac{\vec{u}\vec{u}:(\vec{c}_i\vec{c}_i - c^2\overleftrightarrow{I})}{2c_s^4}\right] \tag{5}$$

where $c_s = 1/\sqrt{3}$ is the speed of sound, w_i is a set of weights normalized to unity, and $\overleftrightarrow{I}$ is the unit tensor in Cartesian space. The relaxation frequency ω controls the kinematic viscosity of the fluid, $\nu = c_s^2\Delta t(\frac{1}{\omega} - \frac{1}{2})$. The $F_i(\vec{x},t)$ term describes the coupling of the fluid with external/internal force fields. The LB solver is particularly well suited to the computational study of a broad class of complex flows, for several reasons. First, free-streaming proceeds along straight trajectories, which greatly facilitates the imposition of geometrically complex boundary conditions, such as those required for the description of realistic arteries. Second, the pressure and stress tensor can be computed locally, on-the-fly, as linear combinations of the discrete populations, with no need of solving a Poisson problem for pressure, nor of taking finite differences of the velocity field to compute the stress. Both aspects represent a major advantage for the practical implementation in complex geometries. Third, since particle collisions are completely local, and free-streaming involves only close neighbors, the LB scheme is well suited to parallel computing and implementations on accelerator-based hardware.

In hemodynamic simulations, the curved blood vessels are shaped on the LB Cartesian mesh scheme via a staircase representation, in contrast to body fitted grids that can be employed in direct Navier–Stokes simulations. This apparently crude representation of the vessel walls can be systematically improved with the mesh resolution. In addition, at the high mesh resolution required to sample low-noise ESS data, the LB method requires rather small time steps (of the order of 10^{-6} s for a resolution of 20 μm). In spite of these drawbacks, the extreme simplicity, scalability and

adaptability to GPU computing, renders the LB method a powerful tool for large-scale hemodynamic simulations.

We expand upon the calculation of pressure and wall shear stress, which are central to hemodynamic applications. In D spatial dimensions, the fluid pressure is given by

$$p(\vec{x},t) = \frac{1}{D}\sum_i f_i^{eq}(\vec{x},t)c_i^2 \tag{6}$$

The dependence on the velocity field configuration is entirely hidden within the local equilibrium, with no need of solving an elliptic and consequently costly Poisson problem. The reason is that LB describes a low-Mach, weakly-compressible fluid, in which the pressure field obeys its own dynamic equation. Since the pressure is an equilibrium property from a kinetic standpoint, its space–time dependence is fixed uniquely by the velocity dependence of the local equilibrium. The shear tensor can be computed as

$$\overleftrightarrow{\sigma}(\vec{x},t) = \frac{\nu\omega}{c^2}\sum_i \left(f_i - f_i^{eq}\right)(\vec{x},t)\vec{c}_i\vec{c}_i \tag{7}$$

where the shear tensor and the strain rate tensor

$$\overleftrightarrow{S} \equiv \frac{1}{2}\left(\vec{\partial}_x\vec{u} + \vec{\partial}_x\vec{u}^T\right) \tag{8}$$

are related by

$$\overleftrightarrow{\sigma} = 2\nu\rho\overleftrightarrow{S} \tag{9}$$

(the superscript T denotes the transpose). It is known [13] that for small departures from local equilibrium (the condition for continuum hydrodynamics to apply), this kinetic representation of the shear tensor reduces to the standard hydrodynamic expression involving the gradients of the velocity field. The kinetic representation is particularly advantageous near boundaries, where the computation of gradients is very sensitive to geometrical details and accuracy. The use of the kinetic expression (7) does not imply higher accuracy in the calculated shear tensor, but is simply more practical since it relieves from the need of taking derivatives of the flow field near the boundaries, which may itself introduce additional sources of errors, independently of the accuracy of the method. Of particular interest for hemodynamic applications is the tensor second invariant (Endothelial Shear Stress or ESS, denoted by $\mathcal{S}$ in the following)

$$\mathcal{S}(\vec{x}_w,t) = \sqrt{(\overleftrightarrow{\sigma}:\overleftrightarrow{\sigma})(\vec{x}_w,t)} \tag{10}$$

where $\vec{x}_w$ represents the position of the geometric wall, or sampling points in close proximity to the mesh wall nodes. $\mathcal{S}(\vec{x}_w,t)$ provides a direct measure of the strength of the shear stress near the wall [25].

A popular choice for imposing no-slip boundary conditions at the wall is through the bounce-back method; this consists of reversing at every time step the post-collisional populations pointing towards a wall node, corresponding to second-order accuracy for planar walls and first-order accuracy for irregular walls [23]. We have adopted this method due to its extreme simplicity in handling irregular vessel boundaries, although more sophisticated alternatives are available [22,26,27]. In the bounce back method the loci of points corresponding to null fluid velocity, that is, the exact no-slip hydrodynamic surface, is not easily identifiable, although it falls between the external fluid mesh nodes and the nearby wall mesh nodes.

2.1. Flow conditions at inlet and outlets

The multi-branched nature of the coronary artery system implies the presence of one inlet each for the left and right coronary arteries at the aortic root, and multiple outlets for each of the network terminals. In LB terms, the imposition of inlet and outlet boundary conditions is simple for regular geometries and is based on the knowledge of the local flow profile, such as a parabolic shape for straight infinite vessels [23,28]. For arterial systems, this information is of limited use due to missing information before the inlet and past the outlet and, more importantly, due to the strong irregularities of the geometry in the inlet region, which limit the accuracy of tomographic reconstruction close to the aorta. We employ a simple method of imposing as inlet and outlet boundary conditions the plug flow profiles obtained by substituting the fluid populations in those regions by the corresponding local equilibria, Eq. (5).

Inflow and outflow conditions affect the large scale features of the flow; the ideal procedure would be to impose clinically measured flow rates for each of the system outlets [29]. One possibility is to measure flow rates non-invasively by using techniques such as Doppler Ultrasound or Magnetic Resonance Angiography, but this is not a straightforward task.

A complete knowledge of the clinically measured values for several outflows is not available. Therefore, for our studies we have employed an empirical procedure based on the following *ansatz* for the flow partitioning at each bifurcation: in the coronary system we assume the physiological condition that the pressure drop in each vessel is driven by the oxygen request from the tissues nourished by the vessel. As a first guess and according to previous observational data, the flow splitting condition is taken as $\phi_1/\phi_2 = S_1/S_2$, where $\phi_{1,2}$ are the two outgoing flow rates and $S_{1,2}$ the corresponding sectional areas, that is, a plug flow profile is assumed for each outgoing vessel in the bifurcation region. This is quite different than what would be obtained by assuming a Poiseuille flow profile in the incoming and two outgoing vessels, with the flow splitting in two outgoing vessels being proportional to the pressure drop, which would lead to $\phi_1/\phi_2 = (S_1/S_2)^2$.

By knowing the incoming flow rate and flow splitting at each bifurcation of the network, and by applying the condition of constant flow rate along each unsplit vessel segment, it is a simple task to evaluate the set of outflows to be applied. However, *a posteriori* simulation results of the global network often exhibit unbalanced flows, especially in the main branches. We then correct our initial guess by altering the outlet conditions with the requirement that the outflows are well balanced, that is, by having for all vessels and in stationary flow conditions equal average flow rates, where the average is determined along the vessel terminal and the unbranched segment.

3. The MUPHY software

The simulation of real-life blood flows involves five basic steps:

(1) Acquisition of MDCT data;
(2) Data segmentation into a stack of slices;
(3) Mesh generation from the segmented slices;
(4) Flow simulation;
(5) Data analysis and visualization.

In this section, we illustrate the way that steps (1) and (2) are handled as the initial, pre-processing sequence, followed by how steps (3)–(5) are handled within our MUPHY code.

3.1. Code organization

The MUPHY simulation package is designed to handle generic geometries, such as those provided by the MDCT acquisitions, and to run large scale simulations on commodity or high-performance

hardware resources. The major advantage of MUPHY is the possibility of concurrently simulating fluid-dynamics together with suspended bodies at cellular and molecular scales. This multi-scale methodology arises from the combined use of Lattice Boltzmann and Molecular Dynamics techniques and has been discussed in a previous, companion paper [12].

In the design of MUPHY, we have followed some basic guidelines that allow us to use the software in a number of diverse applications. The cornerstone of such an approach is to use an indirect addressing scheme [30,31], similarly to other authors in the same field [32,33]. At variance with most Navier–Stokes solvers, the LB mesh is Cartesian, providing extreme simplicity in data management and algorithms. For simple compact regions shaped as parallelepipeds, data on the mesh can be accessed directly via the three-dimensional Cartesian components, and neighboring mesh points are accessed similarly. However, in order to sample high signal/noise ESS data, the LB mesh needs high spatial resolution, with mesh spacing as small as $\Delta x \simeq 20$ μm or below.

At this resolution, given the size of a reconstructed arterial tree (linear edge $\simeq 10$ cm), the resulting simulation box would have a size $\sim 10^{11}\Delta x^3$, clearly beyond the capabilities of most commodity and high-end computers. Therefore, for the LB simulation and all the ancillary stages of simulation (mesh construction and data analysis), only the active computational nodes, those residing inside the arterial vessel, should be taken into account, resulting in huge savings in memory (about three orders of magnitude) and CPU time. The scheme relies on representing sparse mesh regions as a compact one-dimensional primary array, complemented by a secondary array that contains the Cartesian location of each element. In addition, neighboring mesh points are accessed by constructing a connectivity matrix whose elements are pointers to the primary storage array. For the LB mesh topology, this matrix requires the storage of $18 \times N_{mesh}$ elements, where N_{mesh} is the number of active computational nodes.

This indirect addressing approach demands some extra programming effort and may result in a minor (and very reduced on modern computing platforms) computational penalty in simulating non-sparse geometries. This choice provides strategic advantages in handling sparse and generic systems, allowing us to handle a number of fluid nodes of the order 10^9, a size sufficient to study extended arterial systems with a high degree of ramification. We further mention the possibility of simulating the dynamical trajectories of active and passive tracers. Different ways to exchange hydrodynamic information locally between tracers and mesh nodes can be cast within the indirect addressing framework, without major efficiency penalties [34].

3.2. Geometry preparation and mesh generation

We next outline the procedure to build the LB mesh starting from the MDCT raw data for a typical coronary artery system, as illustrated in Fig. 1. Briefly, the procedure for the acquisition of the MDCT data is the following. Patients are imaged axially using a 320×0.5 mm detector CT scanner (Toshiba AquilionOne Dynamic Volume CT, Tochigi-ken, Japan) with 350 millisecond gantry rotation time. The geometry for a single vessel, reconstructed from the MDCT acquisition, is initially evaluated using a research-built post-processing workstation (Vitrea 4.0, Vital Images, Minnetonka, MN, USA), undergoes low-pass filtering, and is formatted as stacked bi-dimensional contours (DICOM slices), with a nominal resolution of 0.1 mm. The DICOM slices are over-sampled along the axial direction up to a slice–slice distance of 0.02 mm to allow further post-processing.

The contour path of each slice is irregular and shaped as a collection of 256 points. The slices are quasi-parallel and mostly transverse to the path connecting different centerlines. Moreover,

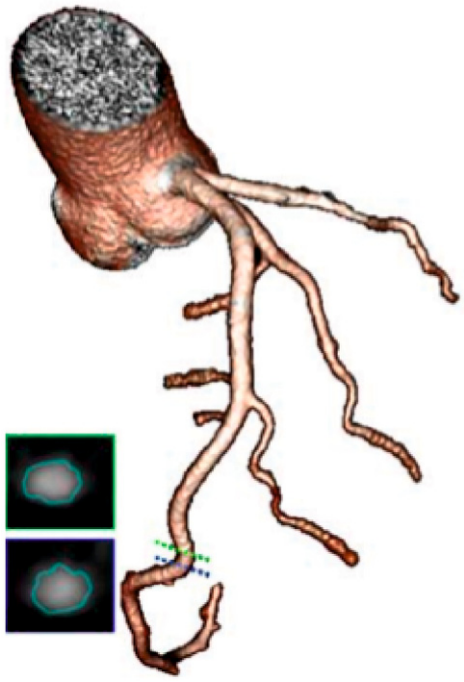

Fig. 1. Left coronary system imaged with 320-detector MDCT, where automatically segmented coronary arteries and branches with lumen diameter greater than 1 mm are illustrated. The smaller pictures illustrate two slices corresponding to the sections indicated by the green and blue dashed lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the sequence of contour points is mostly aligned along the stacking sequence, that is, the contour index does not present any major twisting when moving between contiguous centerlines. Data relative to each vessel in a multi-branched geometry have the same format. Finally, the geometrical data do not carry any topological information about branching, but bifurcations occur where the contours and centerlines of different vessels overlap in space.

The MDCT data formatting is rather general but presents some complications at the stage of the LB mesh creation for a ramified system of vessels. The complications arise from the limited resolution of the MDCT scans that often exhibits mismatches at the vessel attachment and bifurcation regions. Therefore, the preprocessing phase must be considered with care and aided by three-dimensional visualization tools. Starting from the raw MDCT data, the mesh generation proceeds through three distinct stages:

i) Raw MDCT data present a mild level of geometric irregularities that can affect the quality of the LB simulations. We regularize the initial geometry by slightly over-sampling the slices by a factor 2–3. Subsequently, the contour points are smoothed by using a linear filter along the axial direction (see Fig. 2).
ii) Each slice is scanned to locate the set of mesh points that fall in its proximity and enclosed within the vessel surface. For this purpose, for the s-th slice we consider the triangle having for vertices the slice centerline, the c and $(c+1)$ points of the contour. Given the corresponding triangle lying on the $(s+1)$-th slice and vertices coinciding with the slice centerline and c and $(c+1)$ contour points, we consider the prism enclosed between the two triangles and locate the mesh point enclosed within the prism (see Fig. 2). The enclosed mesh points are located by subdividing the prism into three adjacent tetrahedra and finding the mesh points falling inside each of the three tetrahedra. Each tetrahedron has vertices p_1, p_2, p_3 and p_4 and, for each putative mesh point m, we consider the determinants of the following 4×4 matrices

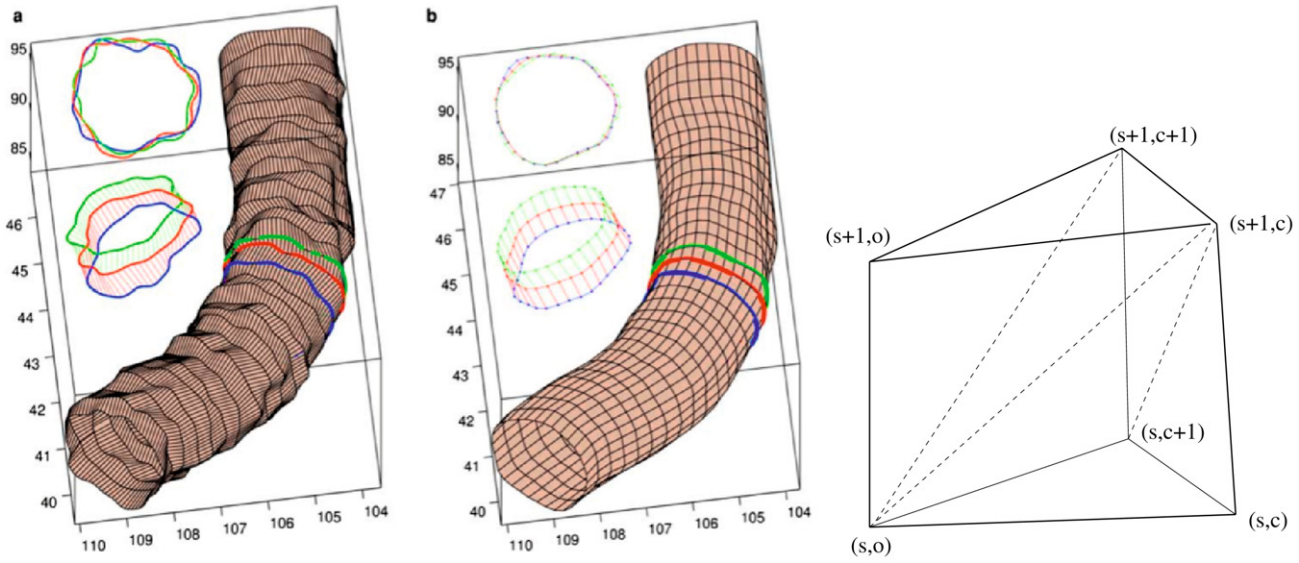

Fig. 2. Three-dimensional stack of slices over a segment of an artery from MDCT data (left panel) and after the regularization procedure (middle panel). The insets show the geometry of three consecutive slices. The right panel shows the prism enclosed by two contiguous slices (solid lines), with vertices (s, o) (centerline), (s, c) and $(s, c+1)$ (contour points) on slice s, and vertices $(s+1, o)$, $(s+1, c)$ and $(s+1, c+1)$ on slice $s+1$. The prism is further subdivided into three tetrahedra enclosed between solid and dashed lines and obtained by considering all possible combinations of four vertices, for the mesh point searching method.

$$d_0 = \det(\tilde{p}_1 \tilde{p}_2 \tilde{p}_3 \tilde{p}_4)$$

$$d_1 = \det(\tilde{m} \tilde{p}_2 \tilde{p}_3 \tilde{p}_4)$$

$$d_2 = \det(\tilde{p}_1 \tilde{m} \tilde{p}_3 \tilde{p}_4)$$

$$d_3 = \det(\tilde{p}_1 \tilde{p}_2 \tilde{m} \tilde{p}_4)$$

$$d_4 = \det(\tilde{p}_1 \tilde{p}_2 \tilde{p}_3 \tilde{m})$$

where $\tilde{p}_\alpha = (p_{\alpha,x}, p_{\alpha,y}, p_{\alpha,z}, 1)$ is a four-component vector constructed by the Cartesian component of a tetrahedron vertex complemented by unity, with $\alpha = 1, \ldots, 4$, and $\tilde{m} = (m_x, m_y, m_z, 1)$. The geometric criterion for m to belong to a tetrahedron is that d_0, d_1, d_2, d_3 and d_4 have the same sign. In this case, m is labeled as a fluid node if it falls between two generic slices, and is labeled as an inlet or outlet node if it falls between the initial or final slices of a vessel (and for slices not belonging to a bifurcation point).

iii) The 18 mesh points surrounding each fluid/inlet/outlet mesh node according to the LB connectivity are inspected in order to locate wall nodes. If any of the 18 surrounding points is yet unlabeled we tag it as a wall node. This phase needs to exclude multiple counting of wall nodes, since several fluid/inlet/outlet points can be connected to the same wall node. In our indirect addressing scheme, this is achieved by considering an ordered sequence of connectivity.

Each of these stages is designed in order to create a work-flow with minimal bottlenecks and high efficiency. The mesh search algorithm is the most time consuming part and we take advantage of the fact that the elementary regions to be scanned are tetrahedra. Thus, for each tetrahedron the search algorithm runs over the mesh points caged by the simplex, filtering out a large number of un-needed operations. Overall, the search algorithm scales linearly with the number of slices. Moreover, the algorithm is parallelized over the number of slices, by distributing the slices in a balanced manner across processors. As a result, this pre-processing stage can be completed in a few minutes on commodity parallel hardware.

An important remark regards the way we choose the sampling points that will be subsequently used to compute the ESS, that is, the points $\vec{x}_w$ in close proximity to the mesh wall nodes. The sampling points are chosen to coincide with the smoothed, off-lattice points forming the slice contours. Each contour point is surrounded by a small number of fluid nodes and a linear extrapolation scheme is used to evaluate the ESS. Clearly, the set of $\vec{x}_w$ points falls between the external fluid and wall mesh nodes and, as the mesh resolution increases, the set converges towards the exact no-slip hydrodynamic surface.

Finally, the volumetric quantities, such as the volumetric flow rates, are computed by using triangular quadratures over the slices. The triangles lie on a slice plane and have vertices $\vec{v}^1$, $\vec{v}^2$ and $\vec{v}^3$ corresponding to the slice centerline o and the c and $(c+1)$ contour points, respectively. We consider four-point quadratures, with the triangular integrals approximated as $\sum_{k=1,4} w_k F(\vec{r}_k)$, where $F(\vec{r}_k)$ are hydrodynamic quantities tri-linearly interpolated from the caging 8 mesh nodes, and the quadrature nodes are given by the combinations $\vec{r}_k = \sum_{i=1}^{3} t_k^i \vec{v}^i$. The quadrature coefficients and weights are $t_1^i = \frac{1}{3}(1,1,1)$ and $w_1 = 0.28125$, $t_2^i = \frac{1}{15}(11,2,2)$, $t_3^i = \frac{1}{15}(2,11,2)$, $t_4^i = \frac{1}{15}(2,2,11)$ and $w_2 = w_3 = w_4 = 0.26041666$ [35], with $i = 1, 2, 3$. Tests have shown that four-point quadratures are accurate enough as compared to more involved seven-point or higher-order quadratures.

3.3. Parallelism and Graphical Processing Units

Three-dimensional simulations of hemodynamics require very large amounts of memory and computing power. As previously mentioned in Section 3.1, MUPHY stores nodes according to a linearized indirect addressing scheme [30,31]. Although it requires, for each node, an additional data structure (the connectivity matrix) that contains the list of its neighboring (fluid, wall, inlet, outlet) nodes, for complex geometries like those found in the study of arterial blood flows, the indirect addressing provides savings in memory requirements and simulation times of up to three orders of magnitude [33].

To exploit the features of modern computing platforms, the MUPHY code has been highly tuned and parallelized. The code takes advantage of optimizations like a) removal of redundant operations; b) buffering of multiply-used operations [36]; c) fusion of the collision and streaming steps in a single loop. This last technique, already in use in other high-performance LB codes [36], sig-

nificantly reduces data traffic between main memory and processor, since there is only one read and one store of all LB populations at each time step. With these optimizations in place, we achieve $\sim 30\%$ of the peak performance of a single core of a modern CPU. Such a result is in line with other highly tuned LB kernels [36]. Indeed we wish to point out that: the algorithm for the update of the LB populations has an unfavorable ratio between number of floating point operations and number of memory accesses; no optimized libraries are available as for other computational kernels (*e.g.*, matrix operations or FFTs); and it is not possible to exploit the SIMD-like operations available on many modern processors since the LB method has a scattered data access pattern.

For the parallelization we chose MPI as the communication library since it offers high portability and good performance due to the availability of highly tuned implementations. To achieve maximum flexibility, we followed an approach that requires a run-time pre-simulation step. This initial stage can be summarized as follows: a subset of lattice nodes is assigned to each computational task, attempting to balance the number of nodes per task. MUPHY supports one-, two- or three-dimensional Cartesian decompositions. For hemodynamics we resort to a different strategy for decomposition in which the connectivity of the populations among the nodes is represented as a graph that is divided in a number of sub-graphs equal to the number of computational tasks by METIS, a set of programs for partitioning graphs based on multilevel recursive-bisection and multilevel k-way algorithms [37]. This is the same approach presented in [33], although for different computing platforms. After the assignment of the nodes to the tasks, the pre-processing phase begins. Basically, each task checks which tasks own the nodes to be accessed during the simulation, in particular for the streaming part of the LB update. Such an information is exchanged by using MPI collective communication primitives, so that each task knows the neighboring peers for send/receive operations. In principle, each node could determine by itself all the information required for the communication phase, but the availability of highly tuned collective communication primitives makes it more efficient to exchange part of these data among the tasks than computing everything locally. The reason why this processing is done at run-time is that the number and the identity of the tasks with which it is necessary to exchange data depends on the geometry of the domain, the total number of tasks, the decomposition method and the boundary conditions. As a consequence, all data structures required for the communication are dynamically allocated and each task allocates only a subset, which is equal to the number of tasks, of the total memory required by the simulation.

In a parallel LB scheme there are several sections requiring data exchange: streaming, handling of periodic boundary conditions and the presence of reflecting or absorbing walls within the computational domain. In MUPHY, both boundary conditions and walls are managed implicitly during the streaming phase by using indirect addressing. The communication scheme is based on the following pattern: the `RECEIVE` operations are always posted in advance by using the corresponding non-blocking MPI primitives, then the `SEND` operations are carried out using either blocking or non-blocking primitives, depending on the parallel platform in use (the choice can be done at run-time). Then, each task waits for the completion of its receive operations, by using the MPI `WAIT` primitives. The last operation, in case of non-blocking `SEND` operations, is to wait for their completion. All point-to-point send and receive operations involve only the LB populations strictly required by each task. For this purpose, a buffering mechanism is used to `PACK` (and `UNPACK`) data that need to be exchanged.

One of the most promising platforms for high performance computing are the general purpose Graphics Processing Units (GPU), high performance many-core processors originally developed as graphics accelerators. Among the GPU, those developed by NVIDIA appear particularly suitable to support general purpose processing thanks to their programming technology, named CUDA [38].

We had access to a NVIDIA Tesla C870 equipped with 16 multiprocessors with 8 processors each, for a total of 128 computational cores that can execute at a clock rate of 1.3 GHz. The processors handle integer types and 32-bit (IEEE 754 single-precision) floating point types. Each multiprocessor has a memory of 16 KByte size shared by the processors within the multiprocessor. Access to data stored in the shared memory has a latency of only 2 clock cycles allowing for fast non-local operations. Each multiprocessor is also equipped with 8192 32-bit registers. The total on-board global memory on the Tesla C870 amounts to 1.5 GByte with a 384-bit memory interface to the GPU that delivers 76.8 GByte/sec memory bandwidth. The latency for the access to this global memory is approximately 200 cycles (two-orders of magnitude slower than access to shared memory) with any location of the global memory visible by any thread, whereas shared memory variables are local to the threads running within a single multiprocessor. The CUDA software development toolkit offers an extended C compiler that supports a Single Instruction Multiple Data (SIMD) programming model (the interested reader may refer to [39] for additional information).

The porting of the kernel of the LB update to the GPU entailed some additions to the original MUPHY code, activated by simple pre-processor compiler directives. First of all, the routines in charge of the LB update were ported to CUDA. In the resulting code, an additional initialization routine starts the GPU and copies all necessary data from the main memory to the GPU global memory. After the initialization, the GPU carries out the collision and streaming phases of the LB update in a single primitive that splits the computational load in a number of threads defined via an external input file. Each thread loads all 19 fluid populations of a lattice site from the global memory of the GPU to local variables. Besides the populations, the entries corresponding to a given lattice site in the connectivity matrix are also loaded from the GPU global memory to local variables. All these load operations are aligned and coalesced, that is, they have an optimal layout according to the GPU memory model. All operations required by the collision phase of the LB update are performed using the local copy of the fluid populations. When the local copy of all fluid populations of a lattice site are updated, they are copied back to the GPU global memory using the double-buffer solution to maintain the current and the new configuration [36]. However, these store operations are not necessarily aligned and coalesced since their target location is defined by the connectivity matrix of the lattice site (these scattered write operations implement the streaming phase of the LB update). The problem is intrinsic to the LB scheme and not related to the indirect addressing scheme, as reported also in [39] where a simple direct addressing scheme based on the geometrical order is used. However, in case of direct addressing, it is possible to alleviate the problem by leveraging the GPU shared memory as a sort of temporary buffer for the store operations that would cause uncoalesced memory accesses [39]. This approach cannot be applied to our case since the access pattern depends on the specific geometry of the fluid domain, that is, a simple knowledge of the dimensions of the bounding box domain is not sufficient to determine where the memory locations of the neighbors of a lattice site are located. Additional details about the consequences of an indirect addressing scheme for an optimal GPU implementation of the LB update can be found in [40]. The function that computes the hydrodynamic variables (density, current, momentum flux tensor, ESS, etc.) is the last component of the GPU porting. As previously discussed, all memory operations are local meaning that only the fluid populations of a lattice site are required and that the result-

Table 1
Timing of 10,000 iterations on an irregular domain $1057 \times 692 \times 1446$ with $\simeq$ 4,000,000 fluid nodes in three different GPU architectures.

System	Total elapsed time (in seconds)	MFLUPS
1 C870 GPU	760	53
2 GT200 GPUs	159	252
8 GT200 GPUs	42	955

ing hydrodynamic variables belong to the same lattice site. As a consequence, the memory access pattern is optimal from the GPU viewpoint.

To highlight the computing capabilities provided by the GPU, we report in Table 1 the results of the GPU + MPI version of the MUPHY parallel code obtained on a cluster of GPUs composed as follows: 4 Quad core Xeon 2.8 GHz connected by Infiniband and each equipped with two pre-production S1070 GPU systems (for a total of 8 GT200 GPUs). For the test we used an irregular domain with a large bounding box ($1057 \times 692 \times 1446$) and a total number of fluid nodes $\simeq 4{,}000{,}000$. The performance is measured by using the *de facto* standard unit for Lattice Boltzmann code, that is, Million of FLuid node lattice Updates per Second (MFLUPS). A MFLUPS is approximately equal to 200 MFLOPS. It is interesting to note that going from one C870 GPU to eight GT200 GPUs, the performance improves by a factor 18. To obtain good results, special care is required in the communication among the GPUs since there is an intermediate passage through the host CPUs. In other words, when GPU X needs to send data to GPU Y, the following steps take place: GPU X copies data to CPU X memory; CPU X sends data, through MPI primitives, to CPU Y; CPU Y copies data from its memory to the memory of its GPU. To minimize the number of communications, a buffering mechanism has been implemented to transmit data that have the same source and target GPUs by means of a single MPI primitive.

4. Test application

We have investigated the left and right human coronary artery systems of a number of patients. In the following, we focus on a particular left coronary system composed of a left main/LAD vessel branching into six primary vessels, labeled according to standard nomenclature as LM/LAD, LCX, RAMUS, D1, D2, D3 and D4. LM/LAD represents the longest vessel, named LM in the proximal region and LAD after the bifurcation with LCX. The LCX vessel further branches into OM1, while RAMUS further branches into BRANCH. Overall, the test case is composed of nine vessels, that is, one inlet vessel at the LAD level and nine outlets for all terminals. For the current illustrative purpose, the geometry does not contain severe diseases and our numerical study is mainly devoted to analyzing the overall behavior of our hemodynamics simulator. In addition, we focus on the steady-state flow pattern and a detailed analysis of the pulsatile flow patterns is left for future work.

The capability of blood-flow simulations to predict regions prone to atherogenesis relies, in a crucial way, on the quality of the computed ESS data. The computed ESS turns out to be very sensitive to the mesh resolution, since ESS is proportional to the gradient of the velocity field and thus is significantly more sensitive to resolution than basic hydrodynamic fields. We consider a reliable calculation of ESS one in which the relative azimuthal dispersion is less than 10%. Preliminary tests of Poiseuille flows in straight cylinders have shown that a safe resolution is obtained for a number of mesh points larger than 50 mesh points per unit radius (data not shown). By taking a radius of about 1 mm, it is clear that the required mesh resolution can ramp up as high as $\frac{1}{20}$ μm^{-1}. For a realistic artery, the dispersion along the azimuthal (or contour) direction has little meaning, since the irregular geometry of the vessel does not guarantee a uniform ESS. We have thus considered the ESS sampled for each vessel slice and averaged over the contour, in order to verify that the data converge to a common value at increasing mesh resolution.

In Fig. 3, the variation of the volumetric flow rates and contour-averaged ESS are compared for different resolutions. The data clearly show convergence to a common profile and, as expected, the ESS profiles converge more slowly than the flow-rate profiles. Moreover, ESS appears to converge more slowly in regions of high ESS, for example, those in proximity of vessel restrictions, than in regions of low ESS, the pattern which is crucial to predict plaque formation. Overall, we deem the mesh spacing $\Delta x = 50$ μm to produce results too noisy for quantitative purposes whereas at $\Delta x = 20$ μm ESS results are quantitatively correct. However, the $\Delta x = 50$ μm data can still be used for a qualitative picture of the flow pattern and ESS distribution. For the size of the coronary system under study, the mesh spacing $\Delta x = 50$ μm results in $\sim 16{,}000{,}000$ mesh points, while for $\Delta x = 20$ μm to $\sim 250{,}000{,}000$ mesh points. The latter requires a global memory allocation of ~ 60 GByte in single precision representation, that is, a total of 5 Tesla S1070 each equipped with 16 GByte memory.

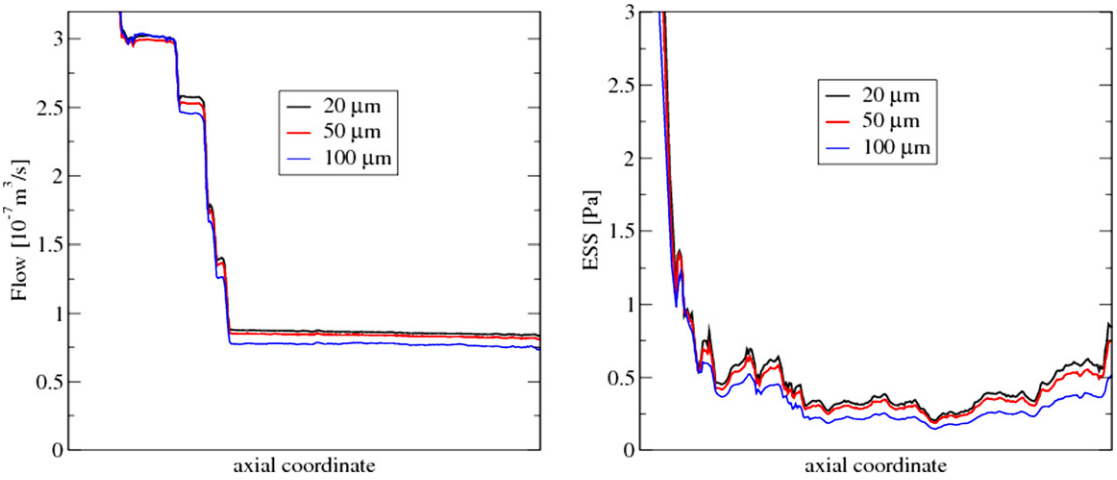

Fig. 3. Flow rate (left) and ESS (right) for the LM/LAD vessel at $\Delta x = 20$ μm (black), $\Delta x = 50$ μm (red) and $\Delta x = 100$ μm (blue) mesh spacing; in the flow rate, the jumps correspond to a bifurcation point along the vessel, the plateaux to the regions of conserved flow rate. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

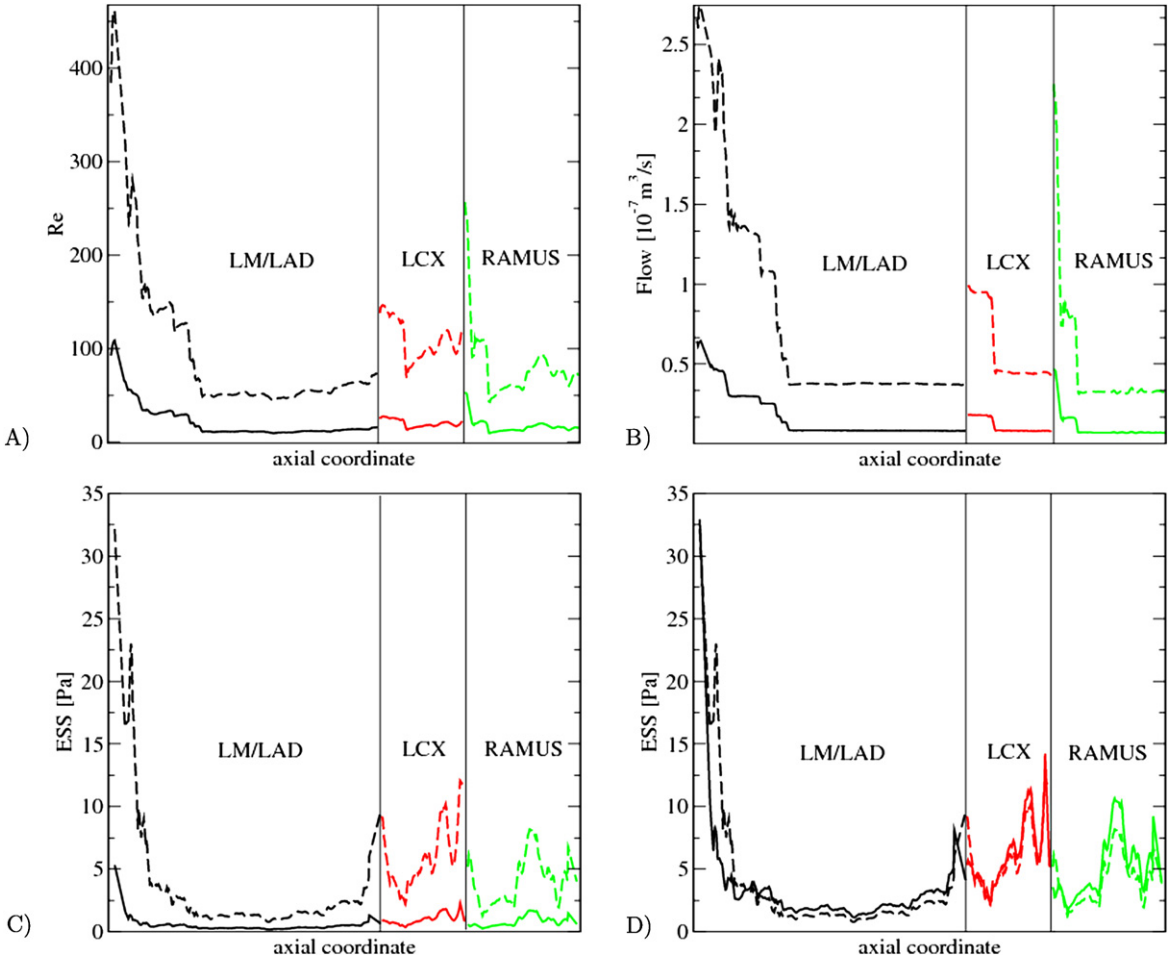

Fig. 4. Hydrodynamic profiles along the vessels axial coordinate for LM/LAD (black lines), LCX (red lines) and RAMUS (green lines) for $\phi^{\text{in}} = \phi_1^{\text{in}}$ (solid lines) and $\phi^{\text{in}} = \phi_2^{\text{in}}$ (dashed lines). A) Reynolds number profile, B) volumetric flow-rate profile, C) contour-averaged ESS, D) contour-averaged ESS with the low-Reynolds profile rescaled by $\phi_2^{\text{in}}/\phi_1^{\text{in}}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We have next analyzed the local Reynolds number ($Re \equiv ua/\nu$, where a is the local vessel contour-averaged radius) and flow rates along the main vessels (LM/LAD, LCX and RAMUS), together with the contour-averaged ESS, for two different values of the inlet flow rate, namely $\phi_1^{\text{in}} = 6.3 \times 10^{-7}\ \frac{\text{m}^3}{\text{s}}$ and $\phi_2^{\text{in}} = 27 \times 10^{-7}\ \frac{\text{m}^3}{\text{s}}$. As illustrated in Fig. 4A, the values of Re along the vessels are widely distributed, in the range 4–100 for the lower influx ϕ_1^{in}, and in the range 20–450 for the higher influx ϕ_2^{in}. The highest Re values correspond to the inlet region, and as the system bifurcates away from the aorta, the values rapidly drop. A similar pattern applies to the volumetric flow-rate profiles (Fig. 4B) and to the contour-averaged ESS profiles (Fig. 4C). In order to detect non-linear effects in the ESS profile, in Fig. 4D we report the ESS for the lower and higher influx values, where the latter has been rescaled by the ratio $\phi_1^{\text{in}}/\phi_2^{\text{in}}$. It is apparent that the overall ESS profile depends only weakly on the influx rate. However, when looking at the azimuthal distribution, the ESS depends strongly and non-linearly on the influx value, as further discussed below.

In Table 2 we report the vessel-averaged Reynolds numbers for the nine vessels, together with the outflows for each of the different outlets. The outflow rates exhibit a similar distribution of values. It should be noted that the outflows refer to the flow rates at each terminal and differ from the vessel-averaged flow rates that are imposed to obtain a balanced system of vessels.

The 3D geometry of the arterial system conveys visual insight into the ESS distribution and its azimuthal variations, as illustrated in Fig. 5. The data show rich patterns, in particular running along the main LM/LAD vessel, and with higher ESS values in the proximal portions of the vessel and lower ESS in the distal region [6, 41]. The simulations at the two influx values show distinct patterns, particularly in proximity to the bifurcations and along the main LM/LAD vessel. A great heterogeneity is found prior to bifurcations, where the vessel cross-section is quasi-elliptical. The ESS maps reveal a broad spectrum of ESS values near branches, with the highest values in the flow-dividing region of bifurcations.

Away from bifurcations, we observe generally lower ESS in the inner region of the vessels, that is, for the epicardial coronary regions (pointing towards the heart), compared to the pericardial regions (pointing away from the heart). The regions of lower ESS are visible close to regions of sudden bending of the vessels, an

Table 2
Computed vessel-averaged Reynolds number and outlet flow rates for two representative cases corresponding to $\phi_1^{\text{in}} = 6.3 \times 10^{-7}\ \frac{\text{m}^3}{\text{s}}$ and $\phi_2^{\text{in}} = 27 \times 10^{-7}\ \frac{\text{m}^3}{\text{s}}$; all outlet flow rates are in units of $10^{-7}\ \frac{\text{m}^3}{\text{s}}$.

Vessel	$\langle Re \rangle_1$	$\langle Re \rangle_2$	ϕ_1^{out}	ϕ_2^{out}
LM/LAD	20	90	0.81	3.7
LCX	21	110	0.79	4.3
RAMUS	17	79	0.69	3.2
D1	13	69	0.40	2.2
D2	18	82	0.75	3.4
D3	11	53	0.36	1.8
D4	12	41	0.48	1.6
OM1	23	126	0.91	4.9
BRANCH	22	123	0.85	4.8

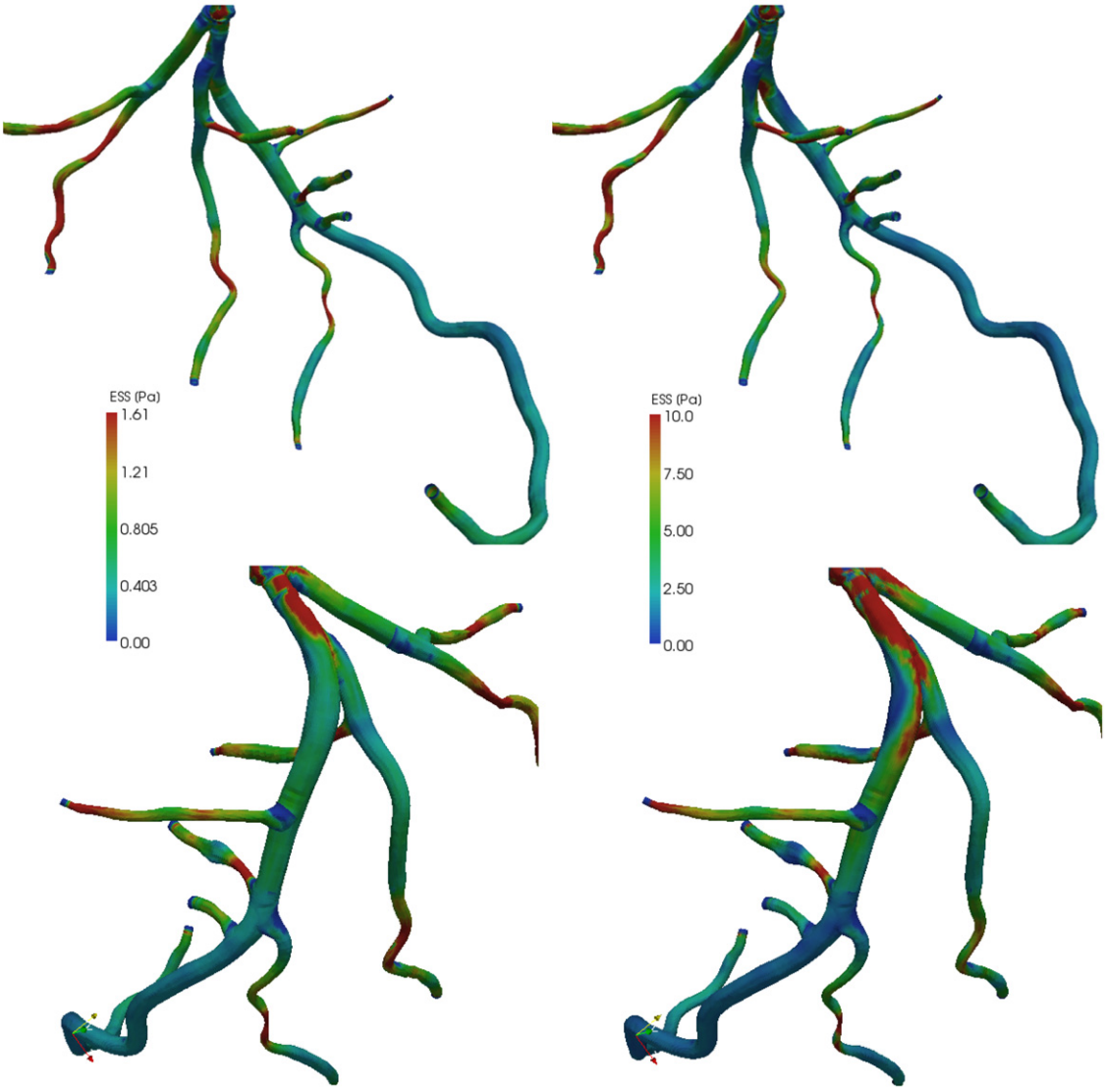

Fig. 5. Three-dimensional arterial walls colored locally according to the ESS. The left panels refer to the anterior (upper panel) and posterior (lower panel) views for ϕ_1^{in}. The right panels correspond to the anterior (upper panel) and posterior (lower panel) views for ϕ_2^{in}. Graphics generated with ParaView [42]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

effect related to the strong centrifugal forces experienced by the fluid in these critical regions. Centrifugal effects are clearly more marked at the highest influx rate. In addition, long patches at high ESS (red regions in Fig. 5) are visible in the high influx case, extending for more than ~ 10 mm.

A particularly useful representation of the ESS data is provided by a two-dimensional diagrammatic layout of the ESS maps, as displayed in Fig. 6. The plot effectively represents the data as an ensemble of contour maps, each having in the vertical axis the vessel curvilinear coordinate and as the horizontal axis the vessel contour coordinate, which is proportional to the mean vessel circumference in order to highlight correlations between vessel size and ESS. The diagram further exploits the vessel topological connectivity to link the two-dimensional stripes in the form of a branched tree, illustrating the regions of high ESS variability which correspond to the attachment regions. It is worth noticing that a faithful two-dimensional representation of any irregular vessel surface would require an involved topological reconstruction. We find the representation of the ESS map illustrated in Fig. 6 very useful. As an example, the diagram in Fig. 6 clearly shows the long-range extension of regions of high or low ESS, together with an extended region of low ESS at the LAD level past the last bifurcation (with D4), down to the LAD outlet level.

5. Conclusions

We have presented the main aspects of the lattice kinetic approach to the computational study of large-scale cardiovascular blood flow, through the detailed illustration of the specific features of the multi-scale/physics code MUPHY. We discussed the full sequence of steps involved by the analysis of blood flow in a real-life cardiovascular case, including medical data acquisition, geometry and mesh generation, flow simulation and data analysis and visualization. We have described the design, implementation and performance of our software on multiple Graphical Processing Units, that enable us to simulate extended coronary systems of unprecedented size on commodity hardware. To the authors' knowledge, this is the first time that grids with two-hundred million voxels make it feasible to address anatomically complete geometries of human coronary arteries and compute endothelial shear stress with high accuracy, revealing a rich structure of the endothelial shear stress, particularly in the vicinity of the main bifurcations.

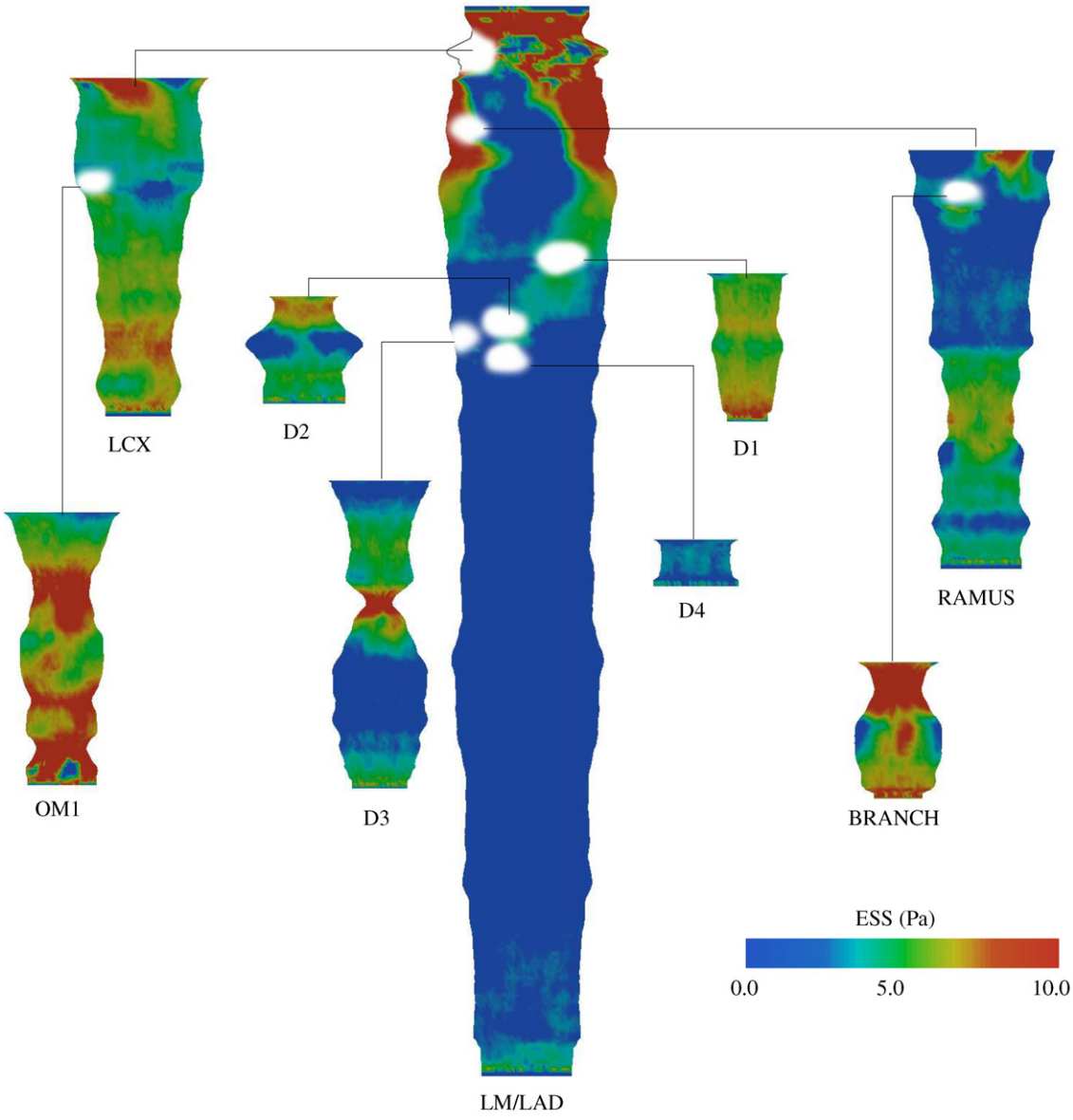

Fig. 6. Two-dimensional diagrammatic representation of the ESS map for the ϕ_2^{in} case, with vessels labeled according to the standard cardiovascular nomenclature. The white compact regions on the ESS map represent attachment regions on a mother vessel, where proper surface points are locally absent. The 2D ESS maps have a horizontal dimension proportional to the local vessel circumference in order to highlight the correlation between the local ESS values and the vessel geometry.

Acknowledgements

This work was supported by Harvard's Initiative in Innovative Computing and by the Cyber Infrastructure Laboratory of the Harvard School of Engineering and Applied Sciences. We wish to thank Michelle Borkin, Joy Sircar, Peter H. Stone, Michael Steigner for useful discussions.

References

[1] Heart and Stroke Encyclopedia, American Heart Association, 2009, http://www.americanheart.org.
[2] Y.S. Chatzizisis, M. Jonas, A.U. Coskun, R. Beigel, B.V. Stone, C. Maynard, R.G. Gerrity, W. Daley, C. Rogers, E.R. Edelman, et al., Circulation 117 (2008) 993.
[3] C. Caro, J. Fitzgerald, R. Schroter, Nature 223 (1969) 1159.
[4] A.M. Shaaban, A.J. Duerinckx, Am. J. Roentgenol. (AJR) 174 (2000) 1657.
[5] A.M. Malek, S.L. Alper, S. Izumo, JAMA 282 (1999) 2035, URL http://jama.ama-assn.org/cgi/content/abstract/282/21/2035.
[6] D.A. Vorp, D.A. Steinman, C.R. Ethier, Comput. Sci. Eng. (2001) 51–64.
[7] C. Cheng, D. Tempel, R. van Haperen, A. van der Baan, F. Grosveld, M.J.A.P. Daemen, R. Krams, R. de Crom, Circulation (ISSN 1524-4539) 113 (2006) 2744, PMID: 16754802, http://www.ncbi.nlm.nih.gov/pubmed/16754802.
[8] P.H. Stone, A.U. Coskun, S. Kinlay, M.E. Clark, M. Sonka, A. Wahle, O.J. Ilegbusi, Y. Yeghiazarians, J.J. Popma, J. Orav, et al., Circulation (ISSN 1524-4539) 108 (2003) 438, PMID: 12860915, http://www.ncbi.nlm.nih.gov/pubmed/12860915.
[9] J.J. Wentzel, R. Krams, J.C. Schuurbiers, J.A. Oomen, J. Kloet, W.J. van Der Giessen, P.W. Serruys, C.J. Slager, Circulation (ISSN 1524-4539) 103 (2001) 1740, PMID: 11282904, http://www.ncbi.nlm.nih.gov/pubmed/11282904.
[10] F.J. Rybicki, H.J. Otero, M.L. Steigner, G. Vorobiof, L. Nallamshetty, D. Mitsouras, H. Ersoy, R.T. Mather, P.F. Judy, T. Cai, et al., Intl. J. Cardiovasc. Imaging 24 (2008) 535.
[11] M.L. Steigner, H.J. Otero, T. Cai, D. Mitsouras, L. Nallamshetty, A.G. Whitmore, H. Ersoy, N.A. Levit, M.F.D. Carli, F.J. Rybicki, Intl. J. Cardiovasc. Imaging 25 (2009) 85.
[12] M. Bernaschi, S. Melchionna, S. Succi, M. Fyta, E. Kaxiras, J. Sircar, Comput. Phys. Comm. 180 (2009) 1495.
[13] R. Benzi, S. Succi, M. Vergassola, Phys. Rep. 222 (1992) 147.
[14] X. Shan, H. Chen, Phys. Rev. E 47 (1993) 1815.
[15] H. Fang, Z. Lin, Z. Wang, Phys. Rev. E 57 (1998) 25.
[16] H. Fang, Z. Wang, Z. Lin, M. Liu, Phys. Rev. E 65 (2002) 41.
[17] B. Chopard, R. Ouared, D.A. Rufenacht, Math. Comput. Simulation 72 (2006) 108.
[18] R. Ouared, B. Chopard, J. Stat. Phys. 121 (2005) 209.
[19] A.M. Artoli, A.G. Hoekstra, P.M.A. Sloot, Intl. J. Mod. Phys. B 17 (2003) 95.
[20] A.M. Artoli, A.G. Hoekstra, P.M.A. Sloot, J. Biomech. 39 (2006) 873.

[21] D. Evans, P. Lawford, J. Gunn, D. Walker, D. Hose, R. Smallwood, B. Chopard, M. Krafczyk, J. Bernsdorf, A. Hoekstra, Philos. Trans. R. Soc. A 366 (2008) 3343.
[22] Z. Guo, C. Zheng, B. Shi, Phys. Fluids 14 (2002) 2007.
[23] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, USA, 2001.
[24] J. Boyd, J.M. Buick, S. Green, Phys. Fluids 19 (2007) 093103.
[25] J. Boyd, J.M. Buick, Phys. Med. Biol. 53 (2008) 5781.
[26] A.J.C. Ladd, R. Verberg, J. Stat. Phys. 104 (2001) 1191.
[27] M. Mazzeo, P. Coveney, Comput. Phys. Comm. 178 (2008) 894.
[28] S.S. Chikatamarla, S. Ansumali, I.V. Karlin, Europhys. Lett. 74 (2006) 215.
[29] L. Grinberg, G.E. Karniadakis, Ann. Biomed. Eng. 36 (2008) 1496.
[30] A. Dupuis, B. Chopard, in: P. Sloot, M. Bubak, A. Hoekstra, B. Hertzberger (Eds.), HPCN Europe, Springer, 1999.
[31] M. Schulz, M. Krafczyk, J. Tolke, E. Rank, High performance scientific and engineering computing, in: M. Breuer, F. Durst, C. Zenger (Eds.), Proceedings of the 3rd International Fortwihr Conference on HPESC, Erlangen, March 12–14, 2001, in: Lecture Notes in Computational Science and Engineering, vol. 21, Springer, 2002.
[32] J. Bernsdorf, S.E. Harrison, S.M. Smith, P.V. Lawford, D.R. Hose, Comput. Math. Appl. 55 (2008) 1408.
[33] L. Axner, J. Bernsdorf, T. Zeiser, P. Lammers, J. Linxweiler, A. Hoekstra, J. Comput. Phys. 227 (2008) 4895.
[34] M. Fyta, E. Kaxiras, S. Melchionna, S. Succi, Comput. Sci. Eng. 10 (March/April 2008).
[35] M. Abramowitz, I. Stegun, Handbook of Mathematical Functions, 5th ed., Dover, New York, 1972.
[36] G. Wellein, T. Zeiser, G. Hager, S. Donath, Comput. Fluids 35 (2006) 910.
[37] G. Karypis, METIS, 2009, http://glaros.dtc.umn.edu/gkhome/views/metis.
[38] NVIDIA CUDA, Compute Unified Device Architecture. Programming guide, 2009, http://www.nvidia.com/cuda.
[39] J. Tolke, M. Krafczyk, in: Proceedings of International Conference for Mesoscopic Methods in Engineering and Science ICMMES07, Munich, 2007.
[40] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, E. Kaxiras, Concurrency and Computation: Practice and Experience, 10.1002/cpe.1466, 2009, in press.
[41] Y.S. Chatzizisis, A.U. Coskun, M. Jonas, E.R. Edelman, C.L. Feldman, P.H. Stone, J. Am. Coll. Cardiol. 49 (2007) 2379.
[42] J. Ahrens, B. Geveci, C. Law, in: C. Johnson, C. Hansen (Eds.), The Visualization Handbook, 1st ed., Academic Press, 2004.