INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **54**:473–496
Published online 22 December 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1405

A versatile incompressible Navier–Stokes solver for blood flow application

M. Garbey[1,*,†] and F. Pacull[2]

[1]*Department of Computer Science, University of Houston, Houston, TX 77204, U.S.A.*
[2]*Department of Mathematics, University of Houston, Houston, TX 77204, U.S.A.*

SUMMARY

We present in this paper an integrated approach to compute quickly an incompressible Navier–Stokes (NS) flow in a section of a large blood vessel using medical imaging data. The goal is essentially to provide a first-order approximation of some main quantities of interest in cardiovascular disease: the shear stress and the pressure on the wall. The NS solver relies on the L_2 penalty approach pioneered by Caltagirone and co-workers and combines nicely with a level set method based on the Mumford–Shah energy model. Simulations on stenosis cases based on angiogram are run in parallel with MatlabMPI on a shared-memory machine. While MatlabMPI communications are based on the load and save functions of Matlab and have high latency indeed, we show that our Aitken–Schwarz domain decomposition algorithm provides a good parallel efficiency and scalability of the NS code. Copyright © 2006 John Wiley & Sons, Ltd.

Received 14 February 2006; Revised 3 October 2006; Accepted 5 October 2006

KEY WORDS: incompressible flow; blood flow; image analysis; parallel algorithm

1. INTRODUCTION AND MOTIVATION

We present in this paper an integrated approach to image base haemodynamic applications [1]. We design a fast and robust numerical solver for the incompressible Navier–Stokes (NS) problem to compute the flow in a section of a large blood vessel using medical imaging data. The shear stress and pressure on the wall are interesting indicators to the medical doctors in the treatment of vascular diseases such as aneurism, atherosclerotic plaque or vein graft, for example. Our goal is essentially to provide a first-order approximation of these indicators with close to real-time calculation.

*Correspondence to: M. Garbey, Department of Computer Science, University of Houston, Houston, TX 77204, U.S.A.
†E-mail: garbey@cs.uh.edu

From medical imaging one may expect to acquire the geometry of a large vessel, as well as the main flow components at the inlet and outlet of the region of interest of the artery. Eventually, one may obtain a video of the main motion in time of the artery if such motion is large enough.

We are interested in having an automated tool that provides a coarse approximation of the shear stress on an artery wall in the presence of a stenosis, for example.

We present a fast, versatile and robust NS solver that relies heavily on the L_2 penalty approach pioneered by Arquis and Caltagirone [2] and combines nicely with a level set method based on the Mumford–Shah energy model [3].

In this problem, the complexity may come from the geometry as well as the motion of the vessel. For example, the section of the coronary artery that lies on the surface of the heart displays some strong motion transverse to the main direction of the blood flow in the artery during the cardiac cycle.

Our flow solver is an immersed boundary-like method [4]. The wall boundary condition is immersed in the Cartesian mesh thanks to a penalty term added to the momentum equation. This technique is simple and easy to implement. We will investigate in this paper the accuracy, robustness and numerical efficiency of the method.

While all the results presented here are in two-space dimensions, the method generalizes easily to three-space dimensions. In particular, there is no issue on mesh generation thanks to the penalty method [2, 5] that imposes automatically the no-slip boundary condition on the wall. Further following the suggestion of Schneider and Farge [6], one can easily take into consideration the wall motion. The drawback of the method is that one gets a lower order approximation of the solution [5]. However, due to the uncertainty on the medical imaging data as well as a number of biological unknowns such as the tissue constitutive laws, especially in the presence of a cardiovascular disease, we expect that the limit on the accuracy of our NS solver will not be the limiting factor in real clinical conditions. Further, we present a numerical technique to recover the shear stress on the wall that complies with the presence of the singular source term introduced in the momentum equation to enforce the no-slip boundary condition. Finally, we take full advantage of the regular data structure of the problem to use a domain decomposition (DD) algorithm that has high numerical efficiency and scales well with the MatlabMPI implementation of Kepner and Ahalt [7].

The plan of the paper is as follows. In Section 2, we formulate the problem and recall the penalty method. In Section 3, we discuss the discretization of the NS equations and the construction of the penalty term. In Section 4, we present the numerical solver and our computation of the shear stress on the wall. In Section 5, we give some numerical results with benchmark problems that are related to stenosis. Section 6 discusses the parallel implementation with MatlabMPI. Section 7 is our conclusion and proposes future directions of work.

2. FORMULATION OF THE PROBLEM AND METHODS

Since we will be concentrating our study on large vessels, we use an incompressible NS fluid flow model [8, 9]. Improvement may be realized *via* some quasi-Newtonian flow model with little additional computing effort [10]. There is an abundant literature on blood flow simulation in complex geometry pipe but confined by fixed walls. The published studies generally rely on the finite element/volume/difference approaches with irregular grid or finite volume/difference techniques with multiblock grid. However, high-order method as spectral element for NS flow in complex geometry vessel is also feasible and can be numerically efficient—see for example [11] and its references. A main goal in such studies is to have a fast solver that provides a solution with desirable accuracy levels, which is achievable since so many factors related to moving walls are neglected. As discussed by Shyy [12] and Ye *et al.* [13], various techniques have been proposed in the literature to treat moving boundary problems. Categorically, we mention Lagrangian (moving grid), Eulerian (fixed grid) and mixed Lagrangian–Eulerian techniques. The Lagrangian approach has the advantage of being able to handle the boundary condition at the interface precisely without ambiguity. The remeshing strategy is employed to track the interface movement. However, if the shape variation is substantial, issues arise in regard to grid skewness, geometric conservation laws and the need for generating new grid at every time instant. The Eulerian approach is, in general, of lower accuracy because it needs to reconstruct the interface without direct physical guidelines. The mixed Lagrangian–Eulerian approach is attractive as the unified approach because it combines the flexibility of a fixed grid with the merit of explicitly tracking the interface evolution.

For fast prototyping of incompressible NS flow, we prefer to stay with the Eulerian approach and combine fast solvers for regular Cartesian grid solution with some form of fictitious DD or locally fitted stencil to implement the boundary conditions [14–18, 12, 13]. In this paper, we will use the penalty method introduced by Arquis and Caltagirone [2] that is simpler to implement than our previous boundary-fitted methods [15] and applies naturally to flow in a pipe with moving walls [6].

The flow of incompressible fluid in a rectangular domain $\Omega = (0, L_x) \times (0, L_y)$ with prescribed values of the velocity on $\partial\Omega$ obeys the NS equations:

$$\partial_t U + (U \cdot \nabla)U + \nabla p - v\nabla \cdot (\nabla U) = f \quad \text{in } \Omega$$

$$\operatorname{div}(U) = 0 \quad \text{in } \Omega$$

$$U = \mathbf{g} \quad \text{in } \partial\Omega$$

We denote by $U(x, y, t)$ the velocity with components (u_1, u_2) and by $p(x, y, t)$ the normalized pressure of the fluid. v is a kinematic viscosity.

With an immersed boundary approach the domain Ω is decomposed into a fluid subdomain Ω_{f} and a wall subdomain Ω_{w}. In the L_2 penalty method, the right-hand side f is a forcing term that contains a mask function $\Lambda_{\Omega_{\mathrm{w}}}$

$$\Lambda_{\Omega_{\mathrm{w}}}(x, y) = 1 \quad \text{if } (x, y) \in \Omega_{\mathrm{w}} \quad 0 \text{ elsewhere}$$

and is defined as follows

$$f = -\frac{1}{\eta}\Lambda_{\Omega_{\mathrm{w}}}\{U - U_{\mathrm{w}}(t)\} \tag{1}$$

U_{w} is the velocity of the moving wall and η is a small positive parameter that tends to zero.

A formal asymptotic analysis helps us to understand how the penalty method matches the no-slip boundary condition on the interface $S_{\mathrm{w}}^{\mathrm{f}} = \bar{\Omega}_{\mathrm{f}} \cap \bar{\Omega}_{\mathrm{w}}$ as $\eta \to 0$. Let us define the following expansion:

$$U = U_0 + \eta U_1, \quad p = p_0 + \eta p_1$$

Formally, we obtained at leading order

$$\frac{1}{\eta}\Lambda_{\Omega_{\mathrm{w}}}\{U_0 - U_{\mathrm{w}}(t)\} = 0$$

that is

$$U_0 = U_{\mathrm{w}} \quad \text{for } (x, y) \in \Omega_{\mathrm{w}}$$

The leading order terms U_0 and p_0 in the fluid domain Ω_{f} satisfy the standard set of NS equations:

$$\partial_t U_0 + (U_0 \cdot \nabla)U_0 + \nabla p_0 - \nu\nabla \cdot (\nabla U_0) = 0 \quad \text{in } \Omega_{\mathrm{f}}$$

$$\operatorname{div}(U_0) = 0 \quad \text{in } \Omega$$

At next order we have in Ω_{w}

$$\nabla p_0 + U_1 + Q_{\mathrm{w}} = 0 \tag{2}$$

where

$$Q_{\mathrm{w}} = \partial_t U_{\mathrm{w}} + (U_{\mathrm{w}} \cdot \nabla)U_{\mathrm{w}} - \nu\nabla \cdot (\nabla U_{\mathrm{w}})$$

Further, the wall motion U_{w} must be divergence free.
At next order we have in Ω_{f}

$$\partial_t U_1 + (U_0 \cdot \nabla)U_1 + (U_1 \cdot \nabla)U_0 + \nabla p_1 - \nu\nabla \cdot (\nabla U_1) = 0$$

with

$$\operatorname{div}(U_1) = 0$$

In the simplest situation where $U_{\mathrm{w}} \equiv 0$, we observe that the motion of the flow is driven by the pressure following a classical Darcy law. η stands for a small permeability. To summarize as $\eta \to 0$, the flow evolution is dominated by the NS equations in the artery, and by the Darcy law with very small permeability in the wall. This actually corresponds to a standard multiscale model of blood flow in the main arteries [19]. But we will use η as an artificial parameter rather than a measured permeability of the wall. As a matter of fact, the discretization used in this paper is not appropriate to compute accurately all the space scales. From the analytical point of view it was shown in [5] for fixed wall, i.e. $U_{\mathrm{w}} \equiv 0$, that the convergence order of the penalty method is of order $\eta^{3/4}$, in the fluid domain, and $\eta^{1/4}$ in the wall.

In this paper, we will restrict ourselves to the situation where the fluid domain traverses the rectangular domain $\Omega = (0, L_x) \times (0, L_y)$ in x direction, and stays at a finite distance from the horizontal boundary of the domain $y = 0$ and $y = L_y$. Ω decomposes into three narrow bands of length L_x that are, respectively, the lower wall $\Omega_{\mathrm{w}}^{\mathrm{low}}$, the fluid domain Ω_{f} and the upper wall $\Omega_{\mathrm{w}}^{\mathrm{upper}}$.

Figures 9 and 10 give examples of the fluid flow domain in the benchmark problems of Section 5.

We impose inlet and outlet boundary conditions on the flow speed along the vertical wall $x = 0$ and $x = L_x$. The flow field at the inlet satisfies the Dirichlet boundary condition

$$u_1(0, y, t) = g(y, t), \qquad y \in (0, L_y)$$

g is set to zero inside the wall, i.e for y such that $(0, y) \in \Omega_{\mathrm{w}}$.

For simplicity, we impose a homogeneous Neumann boundary condition on the flow field, i.e. $\partial u_1/\partial x = \partial u_2/\partial x = 0$ at the outlet $x = L_x$ and a constant profile of the pressure along the outlet wall $x = L_x$.

In the absence of accurate medical data on inflow and outflow boundary conditions for the section of interest of the vessel, we expect to have a very simple laminar flow structure close to a Poiseuille flow near both ends of the section.

To minimize the presence of vortices, we further modify the NS equation in the neighbourhood of the vertical wall $x=0$, and $x=L_x$ by multiplying the convective term with a smooth function $\mathscr{H}(x)$ that is one inside the interval (d, L_x-d) and zero in the interval $(0, d/2) \cup (L_x-d/2, L_x)$. The momentum equation writes

$$\partial_t U + \mathscr{H}(\mathbf{U}\cdot\nabla)U - \nu\nabla\cdot(\nabla U) = f \quad \text{in } \Omega$$

The NS equations simplify then into the Stokes equation in the neighbourhood of the inlet and outlet of the fluid domain Ω_{f}. To avoid reflection of acoustic waves at the outlet, one may then use transparent boundary conditions [20].

On the horizontal wall, we assume either the periodicity of all variables, or we impose the flow speed to be U_{w}.

Since the model has been completely defined, we are now going to present how the set of equations is discretized.

3. DISCRETIZATION

The salient feature of our method is that the mesh generation is trivial irrespective of the wall location. The eventual complexity of the geometry of the fluid domain is taken care of by the forcing term in the penalty method which can be obtained directly from an image segmentation procedure.

The discretization of the NS equations is done with finite differences on a Cartesian grid following the standard staggered grid method [21]. We define then the following grid function:

$$u_1\left(ih_x, \left(j+\frac{1}{2}\right)h_y\right), \quad i=0\dots N_x, \quad j=0\dots N_y-1$$

$$u_2\left(\left(i+\frac{1}{2}\right)h_x, jh_y\right), \quad i=0\dots N_x-1, \quad j=0\dots N_y$$

$$p\left(\left(i+\frac{1}{2}\right)h_x, \left(j+\frac{1}{2}\right)h_y\right), \quad i=0\dots N_x-1, \quad j=0\dots N_y-1$$

on the staggered mesh, of space step $h_x=L_x/N_x$, $h_y=L_y/N_y$.

The diffusion term in the momentum equation is discretized with second-order central finite differences. For the convective term, we are using a method of characteristic that is first order in time and second order in space. To be more specific, let us consider the transport equation

$$\frac{\partial C}{\partial t} = u_1(x,y,t)\frac{\partial C}{\partial x} + u_2(x,y,t)\frac{\partial C}{\partial y} \tag{3}$$

We use the first-order approximation in time:

$$C(x,y,t^{n+1}) = C(x-u_1(x,y,t^n)\,\mathrm{d}t,\, y-u_2(x,y,t^n)\,\mathrm{d}t,\, t^n) \tag{4}$$

at every grid points. Since the velocity components are defined at different grid points of the staggered grid, we use a second-order bilinear interpolation to project the velocity components at the $(x-u_1(x, y, t^n)\,\mathrm{d}t,\ y-u_2(x, y, t^n)\,\mathrm{d}t)$ location. This bilinear interpolation satisfies a maximum principle and the time integration scheme is unconditionally stable. However, to keep the domain of dependency similar to the stencil used for centred finite differences, we choose a space step dt that satisfies the Courant, Friedrichs and Lewy (CFL) condition.

Further improvement in the grid discretization implementation may use a high-order interpolation scheme for the convective term that introduces less dissipation and a mesh refinement in the neighbourhood of the interface $S_{\mathrm{w}}^{\mathrm{f}}$.

The mask function $\Lambda_{\Omega_{\mathrm{w}}}$ is obtained with an image segmentation technique that is a level set method. Since the contours of the image are not necessarily sharp, it is interesting to use the level set method presented in [3] and based on the Mumford–Shah model. For completeness, we are going to describe briefly this method. We refer to the review papers [22, 23] for a more comprehensive description of the level set method in the framework of image analysis. Let $u_0 : [0, L_x] \times [0, L_y] \to \mathbb{R}$ be a given medical image of a blood vessel. Let us denote $C(s)$ the unknown parameterized curve(s) that delineate the vessel. We assume that the unknown function(s) $C(s) : [0, 1] \to \mathbb{R}^2$ is a piecewise $C^1[0, 1]$ function. In the level set method, $C(s)$ is represented by the zero level set of a Lipschitz function: $\phi : \Omega \to \mathbb{R}$.

$C(s)$ should correspond to the minimum of the energy $F(C, c_1, c_2)$:

$$F(C, c_1, c_2) = \mu_1 \cdot (\mathrm{length}(C)) + \mu_2 \cdot (\mathrm{area}(\mathrm{inside}(C))) \tag{5}$$

$$+ \lambda_1 \int_{\mathrm{inside}(C)} |u_0 - c_1|^2 \,\mathrm{d}x\,\mathrm{d}y + \lambda_2 \int_{\mathrm{outside}(C)} |u_0 - c_2|^2 \,\mathrm{d}x\,\mathrm{d}y \tag{6}$$

where

$$c_1(\phi) = \frac{\int_\Omega u_0(x, y) H(\phi(x, y))\,\mathrm{d}x\,\mathrm{d}y}{\int_\Omega H(\phi(x, y))\,\mathrm{d}x\,\mathrm{d}y}$$

$$c_2(\phi) = \frac{\int_\Omega u_0(x, y)(1 - H(\phi(x, y)))\,\mathrm{d}x\,\mathrm{d}y}{\int_\Omega (1 - H(\phi(x, y)))\,\mathrm{d}x\,\mathrm{d}y}$$

H is the Heaviside function $H(z) = 1$, if $z \geqslant 0$, 0 if $z < 0$. To understand the energy function used in this model, let us suppose that μ_1 and μ_2 are set to zero, and let us suppose that the image is a piecewise constant function with values 0 and 1. The angiogram of an artery with a 50% stenosis given in Figure 9 is at first sight close to this description. Clearly, the functions $C(s)$ that realize the minimum of energy are the boundaries that delimit the two sets $u_0 = 0$ and 1. The first two terms in the energy model are common in many active contour methods and control the smoothness of the curve $C(s)$ as well as the detection of the edges.

The numerical process to compute ϕ uses the following evolution problem:

$$\frac{\partial I}{\partial t} = N[I] \tag{7}$$

where $t \geqslant 0$ is an artificial time and N is the associated Euler–Lagrange equation

$$N[I] = \frac{\mathrm{d}}{\mathrm{d}z} H(z) \left[\mu \operatorname{div} \left(\frac{\nabla I}{|\nabla I|} \right) - v - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2 \right], \quad (x, y) \in \Omega$$

Process (7) is based on the fact that ϕ should satisfy $N[\phi] = 0$.

In the numerical implementation, one uses a regularization of the Heaviside function $H_\varepsilon \in C^1(0, 1)$ such that $H_\varepsilon \to H$, as $\varepsilon \to 0$, as well as a reinitialization procedure every few time steps that sharpens the level set function in the neighbourhood of the zero level set.

To successfully apply the Chan–Vese method, one must choose carefully the parameters of the method, i.e. λ_1, λ_2, μ_1, μ_2. From our experience, a successful combination of parameters to detect the edge of a thin long object is common such that with $\lambda_1 \gg 1$, and $\lambda_2 = 1$. This follows somehow the fact that the area inside $C(s)$ is much smaller than the area outside $C(s)$. We will discuss further the robustness of this segmentation technique in Section 5.

The mask function $\Lambda_{\Omega_\mathrm{w}}$ in the momentum equation is consequently

$$\Lambda(x, y) = H(\phi(x, y)), \quad (x, y) \in \Omega$$

There is a natural advantage to combining the level set method with the penalty technique for NS, because the level set function provides directly the source term in the momentum equation. We do not need an explicit geometric representation of the wall boundary, as a differentiable deformable model will do [24]. Further, any artefact possibly produced by the level set method such as an artificial closed subset of fluid in the wall, see Figure 9, should have little impact on the computation of the flow in the artery. As a matter of fact, such cavity has $U \approx U_\mathrm{w}$ on all boundaries.

We have now described the complete discretization of the set of NS equations. For time stepping, we have used in this paper a projection scheme [25]. The time integration writes then

- *Step 1*: prediction of the velocity $\hat{\mathbf{u}}^{k+1}$ by solving either

$$\frac{\hat{\mathbf{u}}^{k+1} - \mathbf{u}^{k,*}}{\Delta t} - v \Delta \mathbf{u}^k = \mathbf{f}^{k+1} - \nabla p^k \quad \text{in } \Omega = (0, L_x) \times (0, L_y) \tag{8}$$

or

$$\frac{\hat{\mathbf{u}}^{k+1} - \mathbf{u}^{k,*}}{\Delta t} - v \Delta \mathbf{u}^{k+1} = \mathbf{f}^{k+1} - \nabla p^k \quad \text{in } \Omega = (0, L_x) \times (0, L_y) \tag{9}$$

with boundary condition

$$\hat{\mathbf{u}}^{k+1} = \mathbf{g} \quad \text{on } \partial\Omega$$

and

$$\mathbf{f}^k = -\frac{1}{\eta} \Lambda_{\Omega_s} \{u^k - U_\mathrm{w}(t^k)\}$$

$u^{k,*}$ is obtained with the method of characteristics (3) and (4).

- *Step 2*: projection of the predicted velocity to the space of divergence-free functions

$$-\mathrm{div}\,\nabla\delta p = -\frac{1}{\Delta t}\,\mathrm{div}\,\hat{\mathbf{u}}^{k+1} \tag{10}$$

$$\mathbf{u}^{k+1} = \hat{\mathbf{u}}^{k+1} - \Delta t\nabla\delta p, \quad p^{k+1} = p^k + \delta p \tag{11}$$

We choose to use the semi-implicit scheme (9) instead of (8) if the mesh is fine enough to make the stability condition on the explicit treatment of the diffusion term too restrictive compared to the CFL condition. We have chosen *a priori* to stay with a first-order scheme in time because of the uncertainty on the (pulsating) inflow velocity condition. The emphasis in this work is more on the robustness of the algorithm than on the accuracy.

We are now going to describe the solvers used to solve the set of discrete equations.

4. SOLVER

The NS calculation decomposes into three steps that are the prediction of the flow speed components, the solution of a Poisson problem for the pressure and eventually the computation of the shear stress along the wall. We will review successively the numerical algorithm used at each step.

4.1. Solver for the momentum equation

The explicit time stepping of (8) does not require any solver. The semi-implicit scheme of (9) requires the solution of a linear system of equations corresponding to the discretization of the operator:

$$L = -\mathrm{d}t\, v\Delta + c\,\mathrm{Id}$$

where c is a coefficient that depends on the penalty term and Id is the identity operator. In practice, $\mathrm{d}t \approx \min(h_x, h_y)$ and $v \preccurlyeq \min(h_x, h_y)$. The discrete operator L^h is then close to the identity. The successive over-relaxation scheme is numerically very efficient on such operator. For parallel computing, we may also adopt the modified Schwarz method of [14, 15] that has been specially designed for small viscosity flow with a main dominant direction of convection as we have here. Further, we know *a priori* that the velocity inside the wall is asymptotically close to U_{w}. Consequently, the velocity does not need to be updated at grid points inside the wall Ω_{w} that are few meshes away from the wall.

We will discuss now the solution process for the Poisson problem that requires *a priori* a larger number of floating points operations (flops) than for the momentum equation.

4.2. Solver for the pressure equation

The pressure equation can be integrated with a number of existing fast Poisson solvers since the discretization grid is regular. It is convenient, for example, to use a (full) multigrid solver here. The arithmetic complexity of this solver is optimum. Further, the iterative solver converges extremely fast for those grid points that are in the solid wall.

A parallel version of our solver uses the Aitken–Schwarz DD technique presented in [16, 26], which is in this situation a direct solver. The implementation will be detailed in Section 6.

We recall that for a moderate number of subdomains, the overall arithmetic complexity of the Aitken–Schwarz algorithm is of the order of the arithmetic complexity of the fast Poisson solver applied to the subdomain factor, the number of subdomains. In practice, the subdomains can be narrow bands, and it is not clear what should be the fastest subdomain solver on a given computer architecture. We choose the subdomain solver for the Poisson problems that provides the smallest elapse time. We refer to [16] for an extensive study of the performance comparing several libraries invoking either an LU decomposition or a Krylov method or a multigrid scheme. We will see later on that our NS solver can perform very quickly the numerical simulation. However, the main difficulty consists of retrieving accurately a quantity of interest in blood flow calculations that is the shear stress on the wall boundary. We are going to address this problem in the next section.

4.3. Computation of the shear stress

While the computation of the hydrodynamic forces exerted by the fluid on the wall is easy to compute using the integral on Ω_{w} of the penalty term [6], the computation of the shear stress is more problematic in the penalty method.

The shear stress in the boundary layer can be obtained from the formula

$$\tau = \nu \left(\frac{\partial w_1}{\partial \xi} + \frac{\partial w_2}{\partial \eta} \right)$$

where (ξ, η) is the normal/tangential coordinate system along the wall, and (w_1, w_2) are the components of the flow field along, respectively, the tangential and normal direction to the wall. Because the flow field (u_1, u_2) is computed on the Cartesian staggered grid we rewrite the shear stress formula in the (x, y) coordinate system. If α denotes the angle of the tangent to the wall at point $M(x, y) \in S_{\mathrm{w}}^{\mathrm{f}}$, with the horizontal axis x, we get

$$\tau = \nu \left(\cos(2\alpha) \left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x} \right) + \sin(2\alpha) \left(\frac{\partial u_2}{\partial y} - \frac{\partial u_1}{\partial x} \right) \right)$$

The flow field is continuous but not differentiable on $S_{\mathrm{w}}^{\mathrm{f}}$. We cannot therefore approximate the shear stress with some central finite differences formula that will require points on both sides of the wall. Further, the computed velocity exhibits small oscillations in the vicinity of $S_{\mathrm{w}}^{\mathrm{f}}$ inside Ω_{f} because of the stiffness of the forcing term in the momentum equation at that location. We have tested two different methods to tackle this problem.

Method A follows a two-step procedure. First, the velocity variables are filtered using a high-order filter. This regularization reduces the Gibbs phenomenon by improving the approximation away from the zone of discontinuities [27]. We recall that a real and even function $\sigma(\eta)$ is called a filter of order p if [27]

- $\sigma(0) = 1$, $\sigma^{(l)}(0) = 0$, $1 \leqslant l \leqslant p-1$,
- $\sigma(\eta) = 0$, for $|\eta| \geqslant 1$,
- $\sigma(\eta) \in C^{p-1}$, for $\eta \in (-\infty, \infty)$.

Let us denote $\tilde{x} = (2\pi/L_x)x$, $\tilde{y} = (2\pi/L_y)y$. For commodity, we set $n_x = N_x/2$, $n_y = N_y/2$. We compute the discrete Fourier expansion

$$u(x, y, t) = \sum_{k_1=-n_x}^{n_x} \sum_{k_2=-n_y}^{n_y} \hat{u}_{k_1,k_2}(t) \mathrm{e}^{\mathrm{i}(k_1 \tilde{x} + k_2 \tilde{y})}$$

The filtered function u^σ is

$$u^\sigma(x,y,t)=\sum_{k_1=-n_x}^{n_x}\sum_{k_2=-n_y}^{n_y}\sigma\left(\frac{|k_1|\kappa}{n_x}\right)\sigma\left(\frac{|k_2|\kappa}{n_y}\right)\hat{u}_{k_1,k_2}(t)\mathrm{e}^{\mathrm{i}(k_1\tilde{x}+k_2\tilde{y})} \tag{12}$$

The parameter $\kappa\geqslant 0$ sets the level of cut in frequency space. As shown in [27], this filtering procedure efficiently reduces the impact of the Gibbs phenomenon on the accuracy of the approximation of a non-smooth periodic function.

All pointwise derivative $\partial u_{1/2}/\partial x$ and $\partial u_{1/2}/\partial y$ are computed with centred finite differences inside the domain of the fluid on those stencils that do not intersect the wall subdomain. We end up with the computation of an approximation of the first-order derivatives of the flow speed at grid points of coordinate (x_i, y_j) that are one or two cells away from the wall boundary.

Second, we proceed with a linear extrapolation formula to approximate the derivatives of interest (DI) functions $\partial u_{1/2}/\partial x$ and $\partial u_{1/2}/\partial y$ at the wall location using exclusively the previous values of the derivatives inside Ω_f. For simplicity, we use an extrapolation formula along the vertical, horizontal or diagonal direction that is the closest to the local normal direction to the wall. For example, with $\alpha\in(0,\pi/8)$ we use a linear extrapolation with the closest two grid values of the DI functions aligned along the vertical axis y. For $\alpha\in(3\pi/8,\pi/2)$, we use the grid values aligned along the horizontal direction. For $\alpha\in(\pi/8,3\pi/8)$, we use the grid values at points (x_i, y_j) of the Cartesian grid aligned along the direction $y=-(h_y/h_x)x$.

Method B uses a gridless approach. Let R be the rectangle $(-d,d)\times(0,l)$ in the (ξ,η) coordinate system that is tangent to the wall at $M(x,y)\in S_\mathrm{w}^\mathrm{f}$ and lies inside the flow region. Let R_h be the set of grid points inside this rectangle, completed by the grid points obtained from the intersection of the Cartesian mesh with the wall in the ball of centre $M(x,y)$ and radius d. For each flow velocity component, one computes a second-order polynomial approximation $P_M(x,y)=a_0+a_1x+a_2y+a_3x^2+a_4y^2+a_5xy$ that fits in the least-square sense the flow field component on R_h. We note that the least-square approximation filters out the possible oscillation of the solution near the wall.

The derivatives of the velocity field are then approximated by the derivative of $P(x,y)$. The dimensions of the rectangle are as follows. The width of the rectangle R is chosen to include approximately three grid points, i.e. $d=\sqrt{h_x^2+h_y^2}$. The length l of the rectangle is chosen to be proportional to the boundary layer thickness, i.e. $l\approx\sqrt{v}$, for small v, in order to capture accurately the parabolic profile of the flow field in the layer. l is *a priori* independent of the mesh size.

We have now described the method to compute the NS set of equations in a pipe of arbitrary shape that traverses the square subdomain Ω, and get an approximation of the shear stress along the wall boundaries. In the next section, we will report on the numerical accuracy obtained with this method.

5. NUMERICAL RESULTS

We are going to experiment our flow solver on various steady and unsteady two-dimensional problems with moderate Reynolds number.

Let us first study the impact of the grid on the accuracy of the computation of the shear stress for a simple Poiseuille flow in a straight pipe.

5.1. Poiseuille flow

We look first at the sensitivity of the result with various positions of a straight pipe of width 0.6 L_y included in the rectangle $(0, L_x) \times (0, L_y) = (0, 3.5) \times (0, 1)$.

In all numerical simulations presented in this paper, we observe in accordance with the numerical experiments of [5] that the influence of the penalty term on the numerical accuracy stays virtually the same for $\eta \ll (\min(h_x, h_y)^2$. No matter how small is η, the scheme stays first order in space. Further how large is the penalty term has no impact on the stability of the time stepping (8) or (9) because this (linear) source term (1) is treated implicitly. We set then the penalty parameter in all simulations to be $\eta = (\min(h_x, h_y))^3$.

In this section, we use a square mesh, i.e. $h_x = h_y$ for all numerical experiments. The main issue is to experiment how robust the computation of the shear stress at the wall remains when the wall boundary location is arbitrarily set. There is no reason for which the wall of the pipe should coincide with the grid points of the Cartesian mesh.

First, we keep the pipe parallel to the horizontal axis. In Figure 1, we test *method A* with various positions of the pipe and three levels of grid refinement. We observe globally the convergence of the method and obtain the correct shear stress within 5% of relative error with $N_y = 160$. *Method A* is, however, fairly sensitive to the choice of the cut-off parameter κ of the filter (12).

In a second set of experiments, we rotate the pipe that now forms an angle of measure α with the horizontal axis. In this experiment, α is kept in the range $(0, \pi/4)$.

We achieve the same level of accuracy as before with *Method A*, with a slightly finer mesh. In all simulations, we have used a moderate level of cut-off in the filtering with $\kappa \approx 0.8$. With no filtering the computation of the shear stress seems to give some random results (see Figure 2). The flow solution exhibits small oscillations next to the wall, because of the singular source term in the momentum equation.

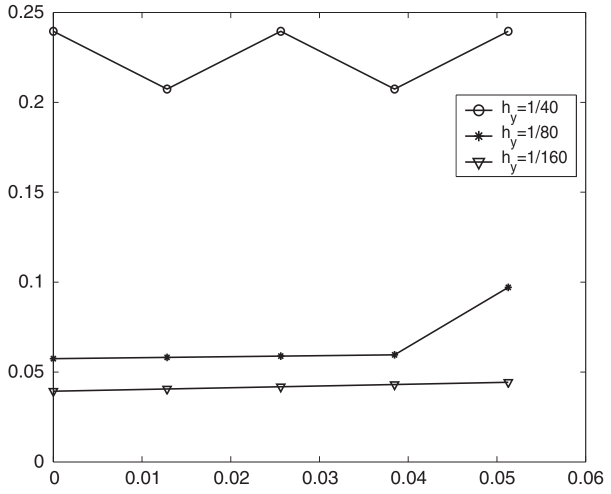

Figure 1. Sensitivity of the computation of the shear stress on the wall as a function of the horizontal position with *Method A*.

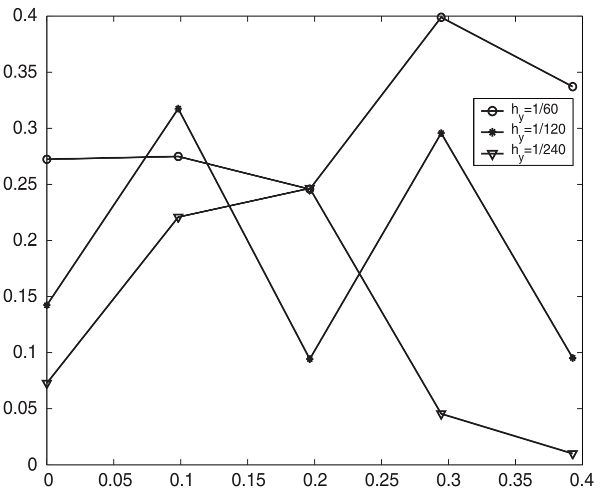

Figure 2. Sensitivity of the computation of the shear stress on the wall as a function of the angle α with *Method A* but no filtering case.

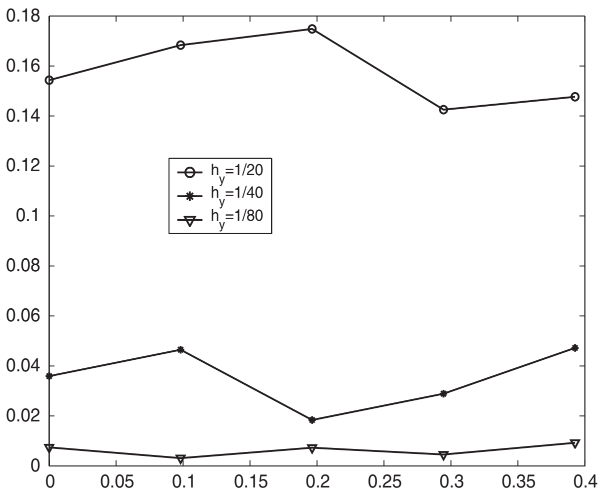

Figure 3. Sensitivity of the computation of the shear stress on the wall as a function of the angle α with *Method B*.

These numerical experiments with Poiseuille flow might be used to calibrate the filter (12) that can be used later for simulations with pipes that have complex geometry.

We have compared *Method B* with *Method A* for the same two sets of experiments.

Method B can provide a more robust and accurate result in all situations—see Figure 3—provided that the dimension of the rectangle R used in the gridless approximation is set properly. The dimension l in the direction orthogonal to the wall is chosen to be mesh independent. From our numerical experiments, to choose l proportional to the boundary layer thickness $\approx \sqrt{v}$ seems optimal. Let us note, however, that the number of grid points of R in the gridless approximation may become large when the space step is of the same order as v.

We now report on computation with a more complex flow domain.

5.2. *Benchmark problems*

We have first verified our code with a steady problem in a straight pipe of width $0.60\ L_y$ obstructed by an obstacle that is the upper half of a disc. The radius of the disc is $0.3\ L_y$, and the obstacle mimics a 50% stenose. The centre of the disc is at coordinates $(L_x/2, 0)$. We took $L_x = 3.5$, $L_y = 1$, and the viscosity is set to $v = 0.01$. The velocity profile at the left entry of the pipe corresponds to a Poiseuille flow with a velocity of maximum value one.

The steady solution is approximated by time marching until $t = 10$. Several finite-element solutions with up to 5×10^4 elements have been computed with the software package ADINA (http://www.adina.com/) for comparison purposes. We verified that our code converges to the same solution with a convergence order that is about one in the L_∞ norm for the velocity components and a slightly better order of convergence order in the L_2 norm. We get similar results for the pressure field in the L_2 norm, but note that the pressure may exhibit some sharp singular peaks at some of the grid points that are next to the wall of the disc. Because we know *a priori* that the pressure should stay continuous at the wall interface, it is straightforward to filter out numerically these singular grid point values.

We are going now to report on numerical simulation for steady and unsteady problems with Reynolds number in the range (300–1200). Those Reynolds numbers are not very large but are representative of real flow problem in the human carotids, or vein graft. Flow in the aorta close to the heart are characterized by several times larger Reynolds number and complex three-dimensional structure [28]. While the purpose of this paper is to provide a feasibility study with simple two-dimensional problems, we are currently developing and testing a three-dimensional version of our method for such clinical test.

Figures 4 and 5 report on a steady problem where $v = 0.0012$ and the stenosis reaches 67%. The domain of computation is the rectangle $([0, 12] \times [0, 1])$. The width of the pipe away from the obstacle is 0.6. We use for the discretization $N_x = 680$, and $N_y = 136$.

We see clearly the recirculation zone right after the obstacle. We observe in our numerical experiments a secondary recirculation zone near the upper side of the wall downstream. The local

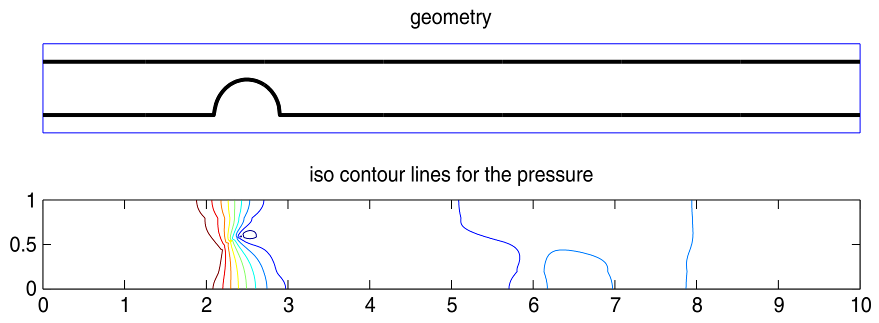

Figure 4. Geometry of the domain and pressure field for a steady flow with a 67% stenosis.

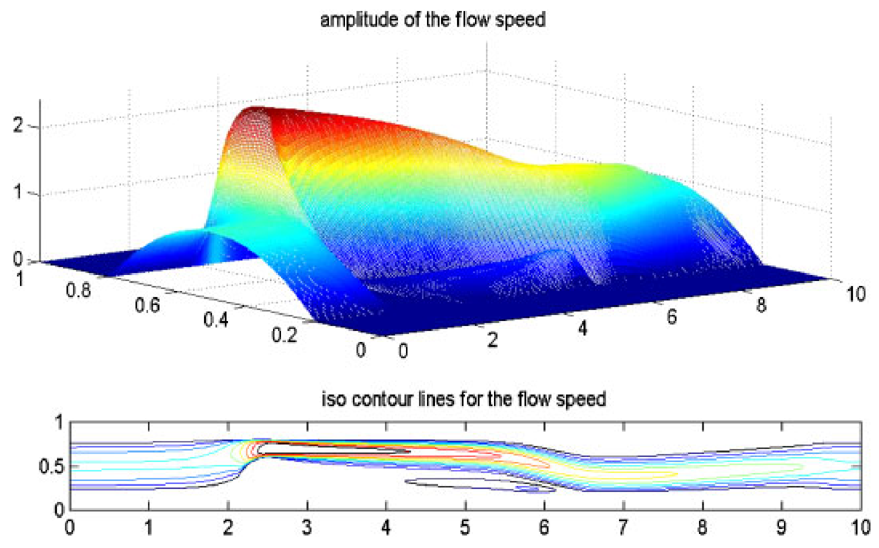

Figure 5. Amplitude of the velocity for a steady flow with a 67% stenosis.

Reynolds number reaches the value 1200 in this calculation. The flow speed has its maximum value, that is 2.42, roughly near the cross section where the pipe gets the more narrow.

We have tested unsteady flows corresponding to a velocity profile at the inlet that is a Womersley solution for a two-dimensional pulsating flow between plates [29]. This solution might be obtained with the method of separation of variable $u_1(0, y, t) = u_0(y)g(t)$ applied to the Stokes problem. In our simulation, $g(t)$ is provided by a measurement of the blood flow main velocity component with a healthy human subject (see Figure 6).

The geometry of the pipe is given in Figure 6—position A, and can be deformed eventually by a periodic smooth motion in time. To be more specific, the velocity field applied to the wall is parallel to the y-axis, and given by the following formula:

$$U_{\mathrm{w}}(x, t) = (0, u_{2\mathrm{w}})$$

with

$$u_{2\mathrm{w}}(x, t) = U^{\mathrm{o}} \left(\exp\left(-\frac{(x - L_x/2)^2}{\mu} \right) - (ax + b) \right) \cos(2\pi t) \tag{13}$$

The parameters a and b are chosen such that the wall at both ends of the pipe stays steady, i.e. $u_{2\mathrm{w}}(0, y, t) = u_{2\mathrm{w}}(L_x, y, t) = 0$. L_x is large and μ is small enough to have a and b close to zero. One can check that U_{w} is divergence free.

The sharpness of the wall curvature in its motion is set by μ and the amplitude of the motion is approximately U^{o}/π. Figure 6 shows the two extreme positions of the pipe in this oscillatory motion of the wall. In this simulation, $v = 0.005$, and the geometry is the same as above. Figure 7 shows the shear stress computed on the lower wall for the wall having oscillations between position A and B at speed (13). We observe a periodic solution in time that has a strong pique in space at the location of the stenose. This slow periodic motion of the wall adds a 10% increase on the shear

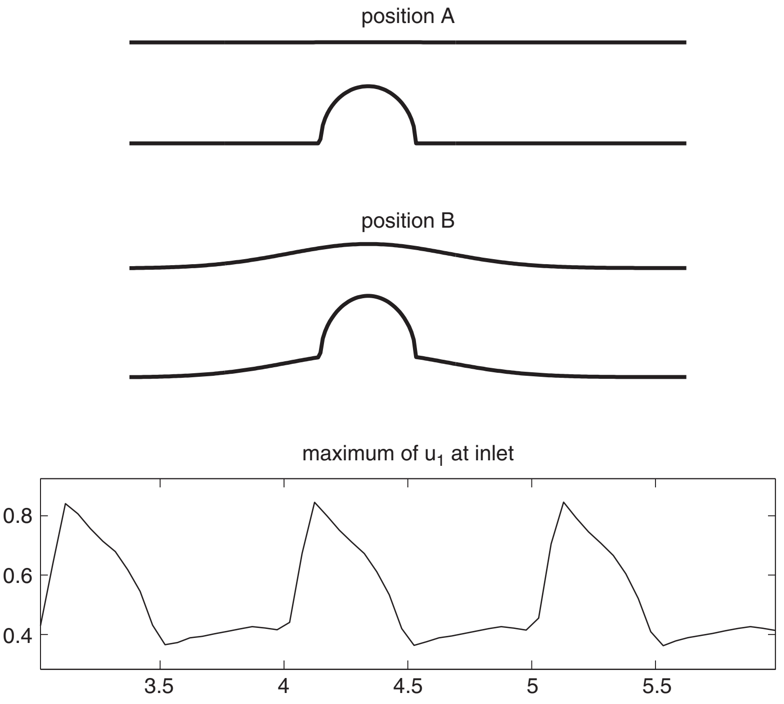

Figure 6. Pulsating flow problem.

stress *versus* the solution with static wall as in position A for an amplitude of the motion that is 0.127, see Figure 8. In this simulation, the maximum of the local Reynolds number oscillates in time in the range (100, 200).

One observes that the shear stress computation becomes noisier with the wall motion. In a real-life situation, the coronary section that is lying on the heart has motions with fast acceleration periods followed by slow relaxation synchronized with the cardiac cycle. We can check with our numerical experiment that the shear stress at the wall is strongly affected by this motion. A detailed analysis of this phenomenon requires that the simulation should be done in three-space dimensions and is currently the subject of further investigations.

Next, we have computed a steady flow in a pipe that is obtained from the image segmentation of the two-dimensional angiogram picture of a carotid. We recall that an angiogram uses X-rays to visualize blood vessels. To create the X-ray images, the physician injects a dye through a catheter that is of roughly one millimetre diameter. This dye, called contrast, makes the lumen of the vessel visible on an X-ray.

Figure 9 shows the result of the segmentation with the method of Chan and Vese [3]. One artefact in this image segmentation corresponds to an annotation of the angiogram with an arrow. A second artefact at the lower right corner of Figure 9 comes from the poor quality of the angiogram itself. We impose a Poiseuille-like boundary condition on the inflow at the inlet of the vessel and zero velocity boundary condition elsewhere along the vertical line $x = 0$. The viscosity is $v = 0.0015$ and the local Reynolds number reaches the value 440. Our flow server based on the L_2 penalty

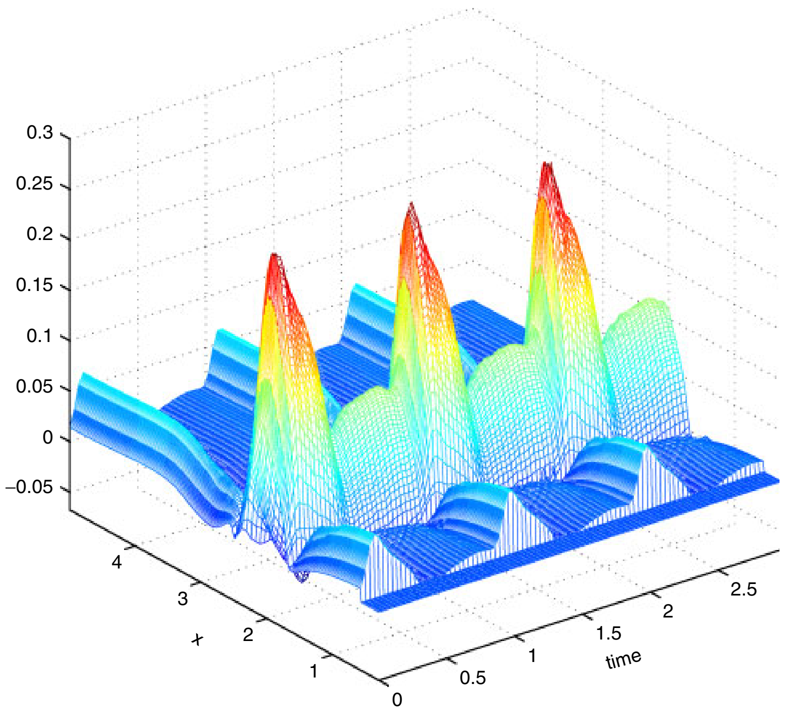

Figure 7. Shear stress on the lower wall that has periodic motion.

method does not suffer from the two artefacts as shown in Figure 10. The result of the image segmentation is also relatively sensitive to the choice of the parameters λ_1, λ_2, μ_1, μ_2 in (5), (6). The sensitivity of the flow simulation to the parameter of the image segmentation can be analysed experimentally or possibly with the automatic differentiation of our NS code that has a very simple implementation. The dynamic of the propagation of the dye from the catheter appears clearly during the beginning of the angiogram examination. It is easy to simulate this phenomenon that is essentially governed by the transport of mass with the method of characteristic already used in the resolution of the momentum equation. We have done numerous simulations of this type for our benchmark problems.

The main difficulty is the validation of the image segmentation that also depends strongly on the quality of the contrast in the X-ray picture.

To validate the medical image analysis, we propose to couple our NS solver with an optimization procedure that matches the flow solution with the measurement of the blood flow speed at a few locations in the vessel using ultrasound techniques, for example. This validation method, which will be the subject of future investigation in our work, requires the use of three-dimensional computations coupled to the image segmentation of three-dimensional angiogram. However, the efficiency of the optimization procedure depends very much on our ability to have very fast solution of the NS flow with complex geometry. To address this issue, we next present a parallel implementation of our NS code.

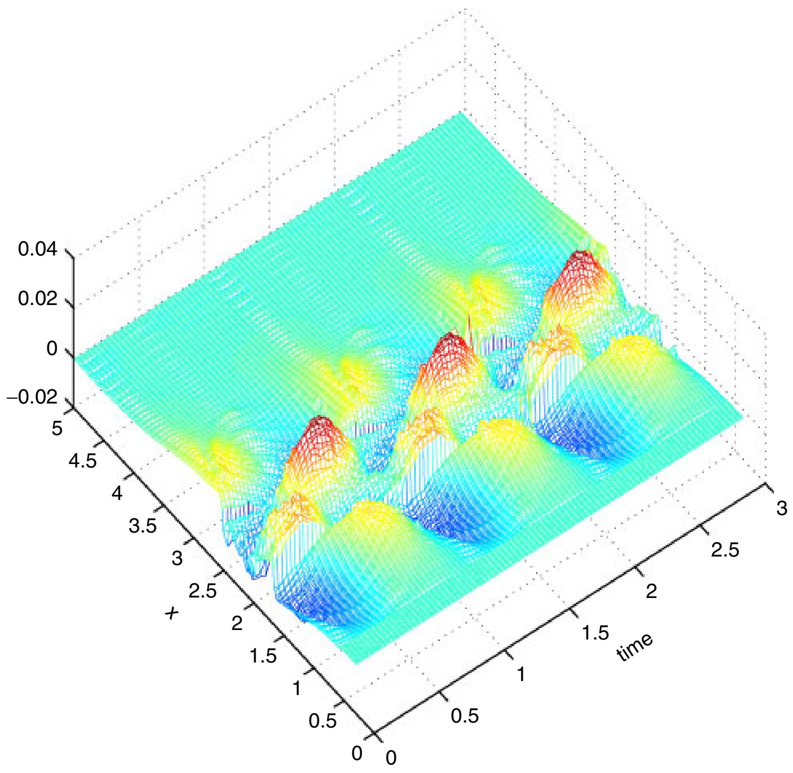

Figure 8. Effect of the motion of the wall on the shear stress.

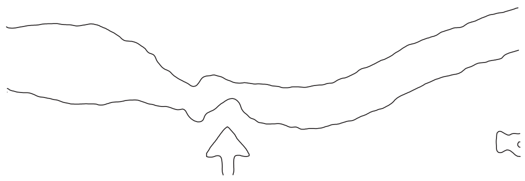
Figure 9. Image segmentation.

6. PARALLEL PERFORMANCE

Our goal is to compute efficiently large-size problems taking advantage of the interactivity of Matlab and its simplicity for coding.

Matlab is a high-level technical computing language and interactive environment for algorithm development, data visualization and numerical computation that is widely used by computational scientists and engineers. It is user friendly and offers a huge array of pre-defined functions. The drawback is that it is an interpreted language, not suited for iterative tasks: it must interpret every line of a loop and therefore cannot perform individual operations as fast as other languages such

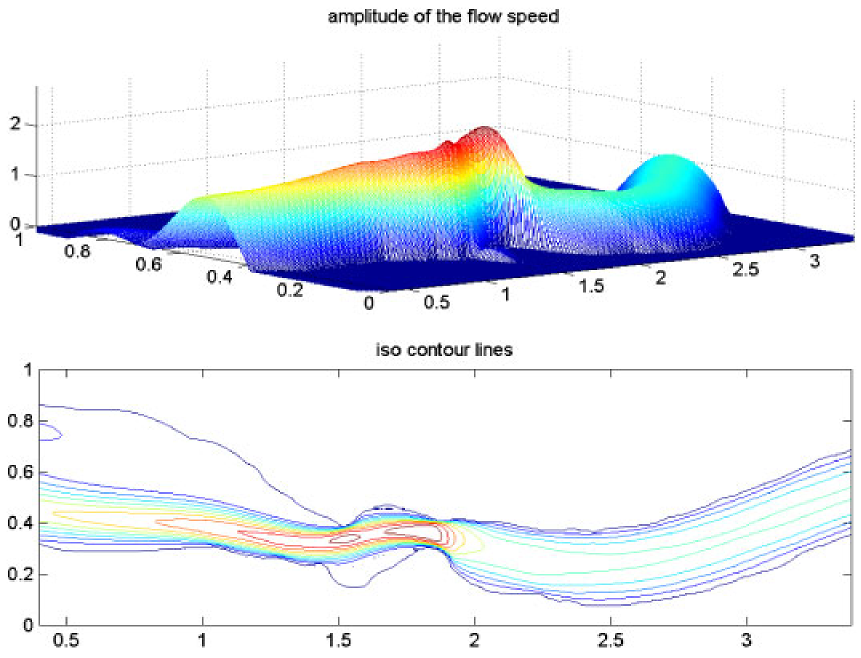

Figure 10. Flow amplitude for a stationary problem.

as C or Fortran. In our experience, the Matlab compiler does not seem to provide much faster codes. We choose here to use the MPI extension of Matlab described in [7]. This software library provides a wonderful framework to produce quickly a parallel code that takes advantage of all the memory available on a shared-memory system. MPI is the defacto standard for communication in parallel scientific applications [30]. MatlabMPI [7] is a set of Matlab scripts that implements a subset of MPI and allows any Matlab program to be run on a parallel computer. The MPI extension of Matlab is public domain and does not require anything other than a standard Matlab licence. This software is designated for users who want to do simple parallelism for codes that do not require a very small latency for the message passing. This approach gives access to larger-scale simulation with a minimum of time spent in the code development.

The message passing in MatlabMPI is done *via* the file system: when a processor sends a message to another processor, it writes the data in a file in a common communication directory. A processor that receives a message should read a file from this common directory. Communication on a cluster cannot be faster than the file server, which updates the directory in each node. The communication within a shared-memory multiprocessor (SMP) system is faster though, since the processors share a local directory. Still, there is a latency due to the detection and writing/reading time. The actual codes for the *MPI_Send* and *MPI_Recv* use file I/O: the *load* and *save* functions of Matlab. The sender creates two files in the common communication directory, one *lock* file and one *buffer* file. The receiver must detect the *lock* file and then load the data from the *buffer* file. This technique is efficient for very large messages since there is no buffering. It is recommended then to group small size messages into a larger package and send it all at once, whenever the algorithm allows to do so. Another way to decrease the latency with the MatlabMPI implementation of [7]

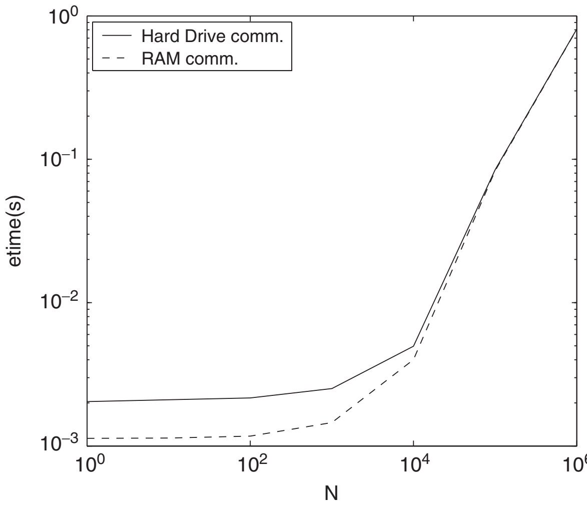

Figure 11. Clock time for one message passing in MatlabMPI using a communication directory either on the hard drive or in main memory.

is to create a virtual communication folder on the RAM. Access to the main memory should be faster than on the hard drive.

For reference, our computer plate-form is an 8-way Opteron running at 2.0 GHz with 32 GB of main memory. Figure 11 reports on the performance of the message passing with MatlabMPI on this system. N is the size of the one-dimensional array that is communicated. The clock time is measured using the Matlab function *etime* for a small Matlab program that sends the one-dimensional array to a given processor and receives it back from this second processor. This elapsed time is divided by two and given in Figure 11 as a function of N.

We observe a latency that is about 2×10^{-3} s using a communication directory in the hard drive and 1×10^{-3} s using the RAM. There is not as much improvement as one may expect, because the load and save procedures have inherently large overhead. Further, we found that the latency can be significantly larger for message passing between cluster nodes linked by a Gigabit ethernet switch.

We are now going to describe in more detail the parallel implementation of our NS solver. For simplicity, we restrict ourselves to explicit time stepping in the momentum equation. It should be noticed that the pressure correction step is always the most computationally expensive step of the projection scheme. Consequently, we will concentrate now on the parallelization of the pressure equation solver (10).

We use the analytic additive Aitken–Schwarz algorithm [26] that is designed to combine efficient distributed computing with high-latency networks and numerical efficiency. To clarify the parallel implementation, let us recall the algorithm that has been presented in more detail in [26].

The rectangular uniform mesh is decomposed into a unidirectional partition of overlapping strip domains. The method is a post-process of the standard additive Schwarz algorithm. An Aitken-like acceleration is applied to the sequences of interfaces produced with the blockwise Schwarz relaxation. Because the eigenvectors of the (linear) trace transfer operator are known, this acceleration provides the *exact* interface condition in one single step irrespective of the overlap between subdomains. For simplicity, we restrict ourselves in this brief description of the algorithm to a decomposition of Ω into two overlapping subdomains: $\Omega = \Omega_1 \cup \Omega_2$ where $\Omega_1 = [0, x_r] \times [0, 1]$ and $\Omega_2 = [x_l, 1] \times [0, 1]$, $x_l < x_r$. We suppose also that the problem to be solved has homogeneous Dirichlet boundary conditions along the vertical sides $[x_l] \times [0, 1]$ and $[x_r] \times [0, 1]$ and homogeneous Neumann boundary conditions on the horizontal sides of the rectangular domain. The additive Schwarz algorithm consists of repeating the following iteration:

$$\begin{gathered} \Delta p_1^{n+1} = \text{RHS} \quad \text{in } \Omega_1, \quad \Delta p_2^{n+1} = \text{RHS} \quad \text{in } \Omega_2 \\ p_{1|\Gamma_1}^{n+1} = p_{2|\Gamma_1}^{n}, \quad p_{2|\Gamma_2}^{n+1} = p_{1|\Gamma_2}^{n} \end{gathered} \tag{14}$$

until convergence. In practice, this algorithm is applied to compute the pressure correction δp rather than the pressure itself. A natural initial condition for the iterative solver is then $p_{1|\Gamma_1}^0 = p_{2|\Gamma_2}^0 = 0$.

We observe that a cosine expansion of the trace of the solution on the interface

$$p(y)_{|\Gamma_i} = \sum_{k=1\ldots N_y-1} \hat{p}_{|\Gamma_i}^k \cos(k\pi y), \quad i = 1 \text{ or } 2 \tag{15}$$

provides a diagonalization of the trace transfer operator:

$$(p_{1|\Gamma_1}^n, p_{2|\Gamma_2}^n) \xrightarrow{\text{T}} (p_{1|\Gamma_1}^{n+1}, p_{2|\Gamma_2}^{n+1}) \tag{16}$$

The Poisson problem satisfied by the pressure decomposes onto a set of independent two-point boundary value problems:

$$\frac{\partial^2 \hat{p}_k(x)}{\partial x^2} - \mu_k \hat{p}_k(x) = \widehat{\text{RHS}}_k \quad \forall k \tag{17}$$

Let us denote T_k the trace operator for each wave component of the interface:

$$(\hat{p}_{1|\Gamma_1}^{n,k} - \hat{p}_{\Gamma_1}^k, \hat{p}_{2|\Gamma_2}^{n,k} - \hat{p}_{\Gamma_2}^k) \xrightarrow{\text{T}_k} (\hat{p}_{1|\Gamma_1}^{n+1,k} - \hat{p}_{\Gamma_1}^k, \hat{p}_{2|\Gamma_2}^{n+1,k} - \hat{p}_{\Gamma_2}^k) \quad \forall k \tag{18}$$

The operators T_k are linear and the sequences $\{\hat{p}_{1|\Gamma_1}^n\}$ and $\{\hat{p}_{2|\Gamma_2}^n\}$ have linear convergence. This is expressed by the set of linear equations

$$\hat{p}_{1|\Gamma_2}^{n+1,k} - \hat{p}_{\Gamma_2}^k = \hat{\delta}_k^1 (\hat{p}_{2|\Gamma_1}^{n,k} - \hat{p}_{\Gamma_1}^k), \quad \hat{p}_{2|\Gamma_1}^{n+1,k} - \hat{p}_{\Gamma_1}^k = \hat{\delta}_k^2 (\hat{p}_{1|\Gamma_2}^{n,k} - \hat{p}_{\Gamma_2}^k) \tag{19}$$

where $\hat{\delta}_k^1$ and $\hat{\delta}_k^2$ are the so-called damping factors associated with each subdomain Ω_1 and Ω_2.

These damping factors are computed analytically from the eigenvalues of the operators. We apply a generalized Aitken acceleration separately to each wave coefficient in order to get the exact limit of the sequence on the interfaces based on the first Schwarz iterate. It consists of simply solving the 2×2 linear system (19) where $n = 0$, of unknown $(\hat{p}_{\Gamma_1}^k, \hat{p}_{\Gamma_1}^k)$. The same procedure applies with an arbitrary number of subdomains but then the matrix of the linear system corresponding to (19) has a pentadiagonal structure. Finally, the exact solution at the artificial interfaces Γ_i is reconstructed in physical space from its discrete trigonometric expansion.

For the nonhomogeneous mixed boundary conditions of our pressure problem (10), we must add a pre-processing step that consists of solving the zero-mode equation:

$$\frac{\partial^2 \hat{p}_0(x)}{\partial x^2} = \widehat{\mathrm{RHS}}_0 \tag{20}$$

with $\partial \hat{p}_0/\partial x = 0$ at $x = 0$ and $\hat{p}_0(L_x) = 0$.

To summarize, the algorithm writes:

- preliminary step: compute analytically each damping factor for each wave number;
- *Step 1*: solve the mode zero one-dimensional equation;
- *Step 2*: perform one additive Schwarz iterate in parallel;
- *Step 3*: apply the generalized Aitken acceleration on the interfaces;
  - 3.1: compute the cosine expansion of the traces of p on the artificial interfaces for the initial condition and the first Schwarz iterate;
  - 3.2: apply the generalized Aitken acceleration separately to each wave coefficients in order to get the limit expressed in the cosine functions vector basis;
  - 3.3: transfer back the interface limit values into the physical space;
- *Step 4*: compute the solution for each subdomain in parallel using the boundary values obtained from Step 3.

In our MatlabMPI implementation Steps 1 and 3.2 are not parallelized and computed redundantly by all the processors. An all-to-all Broadcast through cyclic reduction is used to gather the zero-mode component of the right-hand side and to build the artificial interface matrix.

Besides the pressure equation (10), the prediction and correction steps of the projection scheme are also parallelized: this implies one extra communication process with the neighbouring subdomains at the beginning of those two steps, in order to build the right-hand side of the equation. We used a fast LUPQ solver offered by Matlab that is based on the UMFPACK library [31] to solve each subdomain. The operator matrix M is decomposed into a unit lower triangular matrix L, an upper triangular matrix U, a permutation matrix P and a column reordering matrix Q so that $PMQ = LU$, using the fact that it is a very sparse matrix. This factorization, based on a tree and special ordering, optimizes the memory access patterns. We note that the decomposition is computed once and for all in the initialization phase of the NS code. The pressure solve reuses this decomposition at every time step. The UMFPACK decomposition provides particularly good performance in Matlab compared to the original Linpack LU solver that is typically two to three times slower for the size of the problem considered here.

Overall, the Aitken–Schwarz parallel solver is a direct solver and takes the exact same elapse time at each time step.

Figures 12 and 13 report, respectively, on the speedup and scalability of our parallel implementation of the complete NS code. In the scalability test, we have run successively a problem of size 141×567 with two processors, 200×801 on four processors, and 283×1129 on eight processors. The number of unknowns grows linearly with the number of processors, but the aspect ratio h_x/h_y of the grid stays the same. The scalability performance obtained in Figure 13 is particularly good because the arithmetic complexity of an NS solver, in general, grows faster than linearly with respect to the number of unknowns.

Let us note that one should synchronize the processors in order to measure accurate computational times, since the Matlab application starting time can vary. Performance speedup is based

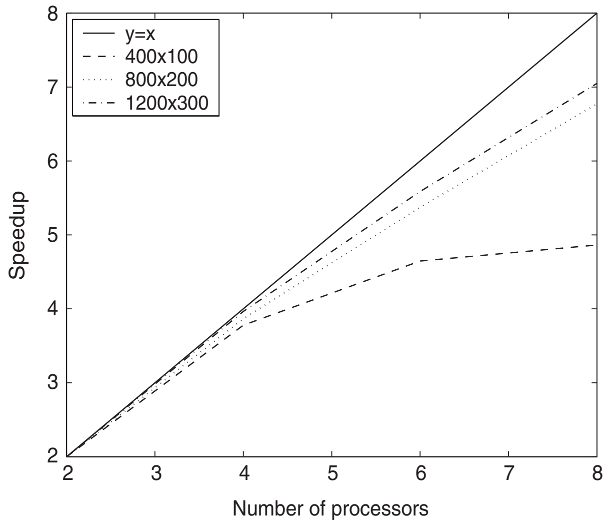

Figure 12. Parallel speedup for the parallel Navier–Stokes code.

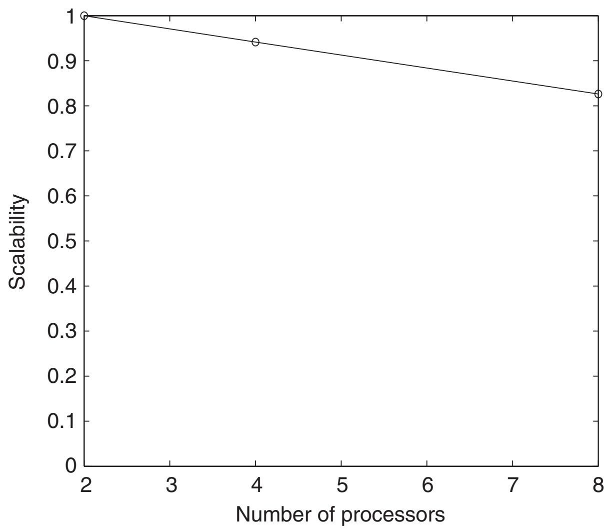

Figure 13. Parallel scalability for the Navier–Stokes code.

on the reference time provided by the code running with two subdomains on two processors. This speedup is, therefore, significantly better than what one obtains by comparing our parallel code with its sequential version. The speedup of the parallel code with two processors compared to the sequential code running on one processor is only 1.56. The overhead in the sequential code comes partially from the nature of the algorithm itself, which requires two subdomain solves, and partly from the fact that the parallel code has many more lines of code to be interpreted. Another aspect of the parallelization is that we gain computational time in the decomposition of the operator process: this time is divided by the square of the number of subdomains since the complexity of the decomposition is proportional to the bandwidth of the subdomains. While the speedup obtained in Figure 12 is not optimum, the MatlabMPI implementation still seems very attractive and allows for solving large problem size.

To give an overall idea of the MatlabMPI code performance, 100 time steps for a 100×400 problem size takes less than 10 s on eight processors. This is roughly the elapsed time needed to simulate one complete cardiac cycle with $\nu \approx 10^{-2}$.

A parallel implementation in Fortran or C can be further improved by taking advantage of the more efficient communication schemes offered by MPI for these programming languages. One can also adapt the communication scheme in such a way that high-frequency components of the interfaces are exchanged between neighbour subdomains only. We refer to the optimized implementation presented in [32] for grid computing that was done for three-space dimension elliptic problems. We will now summarize the conclusion of our investigation.

7. CONCLUSION

We have presented an integrated approach to compute quickly an incompressible NS flow in a section of a large blood vessel using medical imaging data. The key feature of the method is to use the L_2 penalty method pioneered by Arquis and Caltagirone [2]. The first two major advantages of the method are the easiness of the implementation and a discretization that allows the use of fast elliptic solvers optimized for cache memory access. Second, our numerous numerical experiments have shown that the numerical method is fairly robust. Third, the penalty method combines naturally with a level set method that directly provides the penalty term in the momentum equation. Further, while the numerical method is first order, we can surprisingly recover a reasonable estimate of the shear stress on the wall using a gridless approach. Finally, we have presented a straightforward parallelization of the NS code based on the Aitken–Schwarz algorithm and MatlabMPI. The Matlab language allows a fast prototyping of the code while MatlabMPI offers the possibility of easily accessing a large amount of memory.

The drawback of the approach followed in this paper is obviously the fact that we cannot solve accurately the boundary layers that may appear in the flow field. This type of simulation should then be limited to moderated values of the Reynolds number. We are currently developing a multiscale heterogeneous domain decomposition version of our code to address this problem [17].

Our next step is, however, to generalize the present work to simulation in three-space dimensions by extending the capability of our parallel three-dimensional NS solver presented in [14, 15] to blood flow simulation related to real clinical cases.

ACKNOWLEDGEMENTS

We would like to thank Chandler Wilkerson for his help on the MatlabMPI setting. Research reported here was supported by Award 0305405 from the National Science Foundation.

REFERENCES

1. Thomas JB, Milner JS, Rutt BK, Steinman DA. Reproductibility of image-based computational fluid dynamics models of the human Carotid bifurcation. *Annals of Biomedical Engineering* 2003; **31**:132–141.
2. Arquis E, Caltagirone JP. *Sur Les Conditions Hydrodynamiques au Voisinage d'une Interface Milieu Fluide-Milieux Poreux*: *Application a la Convection Naturelle*, CRAS, Paris II, vol. 299, 1984; 1–4.

3. Chan TF, Vese LA. Active contours without edges. *IEE Transactions on Image Processing* 2001; **10**(2):266–277.
4. Peskin CS. The immersed boundary method. *Acta Numerica* 2002; 1–39.
5. Angot P, Bruneau CH, Fabrie P. A penalisation method to take into account obstacles in viscous flows. *Numerische Mathematik* 1999; **81**:497–520.
6. Schneider K, Farge M. Numerical simulation of the transient flow behavior in tube bundles using a volume penalization method. *Journal of Fluids and Structures* 2005; **20**:555–566.
7. Kepner J, Ahalt S. MatlabMPI. *Journal of Parallel and Distributed Computing* 2004; **64**(8):997–1005.
8. McDonald DA. *Blood Flow in Arteries* (3rd edn). Edward Arnold: Paris, 1990.
9. Xu XY, Collins MW, Jones CJH. Flow studies in Canine Aortas. *Journal of Biomechanical Engineering* (ASME) 1992; **114**(11):505–511.
10. Perktold K, Resch M, Florian H. Pulsatile non-Newtonian flow characteristics in a three dimensional human Carotid bifurcation model. *Journal of Biomechanical Engineering* (ASME) 1991; **113**:464–475.
11. Deville MO, Fischer PF, Mund EH. *High Order Methods for Incompressible Fluid Flow*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2002.
12. Shyy W. Moving boundaries in micro-scale biofluid dynamics. *Applied Mechanics Review* 2001; **54**(5):405–453.
13. Ye T, Mittal R, Udaykumar HS, Shyy W. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics* 1999; **156**:209–240.
14. Garbey M, Kuznetsov YA, Vassilevski YV. A parallel Schwarz method for a convection–diffusion problem. *SIAM Journal on Scientific Computing* 2000; **22**(3):891–916.
15. Garbey M, Vassilevski YV. A parallel solver for unsteady incompressible 3D Navier–Stokes equations. *Journal of Parallel Computing* 2001; **27**:363–389.
16. Garbey M, Hadri B, Shyy W. Fast elliptic solver for incompressible Navier–Stokes flow and heat transfer problems on the grid. *43rd Aerospace Sciences Meeting and Exhibit Conference*, Reno, January 2005; *Paper Number: AIAA-2005-1386*.
17. Garbey M, Dia BO, Picard C, Tran Son Tay R. Heterogeneous domain decomposition for multi-scale problems. *43rd Aerospace Sciences Meeting and Exhibit Conference*, Reno, January 2005, *Paper Number*: *AIAA-24880*.
18. Glowinski R. A fictitious domain approach to the direct numerical simulation of incompressible flow past moving rigid bodies: application to particulate flow. *Journal of Computational Physics* 2001; **162**:363–426.
19. Quarteroni A, Veneziani A, Zunino P. Mathematical and numerical modelling of solute dynamics in blood flow and arterial walls. *SIAM Journal on Numerical Analysis* 2002; **39**(2):1488–1511.
20. Bruneau CH. Boundary conditions on artificial frontiers for incompressible and compressible Navier–Stokes equations. *M2AN* 2000; **34**(2):303–314.
21. Peyret R, Taylor TT. *Computational Methods for Fluid Flow*. Springer Series in Computational Physics. Springer: Berlin-Heidelberg-New York, 1985.
22. Osher S, Paragios N. *Geometric Level Set Methods in Imaging*, *Vision and Graphics*. Springer: Berlin, 2003 (ISBN 0387954880).
23. Sethian JA. Level set methods and fast marching methods: evolving interfaces in computational geometry. In *Fluid Mechanics*, *Computer Vision*, *and Materials Science*, Ciarlet PG, Iserles A, Kohn RV, Wright MH (eds). Cambridge Monographs on Computational Mathematics. Cambridge University Press, 1999.
24. Metaxas D. *Physics-Based Deformable Models*: *Application to Computer Vision*, *Graphics and Medical Imaging*. Kluwer Academic Publishers: Dordrecht, 1996.
25. Chorin AJ. The numerical solution of the Navier–Stokes equations for an incompressible fluid. *Bulletin of the American Mathematical Society* 1967; **73**:928.
26. Garbey M, Tromeur Dervout D. On some Aitken like acceleration of the Schwarz method. *International Journal for Numerical Methods in Fluids* 2002; **40**(12):1493–1513.
27. Gottlieb D, Shu C-W. On the Gibbs phenomenon and its resolution. *SIAM Review* 1997; **39**(4):644–668.
28. David NKu. Blood flow in arteries. *Annual Review of Fluid Mechanics* 1997; **29**:399–434.
29. Womersley JR. Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is known. *Journal of Physiology* 1955; **127**:553–563.
30. MPI Forum. *MPI*: *A Message-Passing Interface Standard*, Document for a Standard Message-Passing Interface, University of Tennessee, 1994.
31. Davis TA, Duff IS. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications* 1997; **18**(1):140–158.
32. Barberou N, Garbey M, Hess M, Resch M, Rossi T, Toivanen J, Tromeur Dervout D. On the efficient meta-computing of linear and nonlinear elliptic problems. *Journal of Parallel and Distributed Computing* 2003; **63**(5):564–577 (special issue on grid computing).