A Versatile Incompressible Navier Stokes Solver for Blood Flow Application

Marc Garbey, Francois Pacull*

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-06-01

March 1, 2006

Keywords: Navier Stokes, Blood flows, Penalty method, Aitken-Schwarz, MatlabMPI

Abstract

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.

* Mathematics Department, University of Houston

A Versatile Incompressible Navier Stokes Solver for Blood Flow Application

M. Garbey[†], F. Pacull[††]

† Department of Computer Science, University of Houston, Houston, TX 77204, USA

†† Department of Mathematics, University of Houston, Houston, TX 77204, USA

Abstract

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.

1 Introduction and Motivation

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. 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 Caltagirone and co-workers [4] and combines nicely with a level set method based on the Mumford-Shah energy model [5].

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 [23]. 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, 4] that imposes automatically the no slip boundary condition on the wall. Further following the suggestion of [26] 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 [2]. 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 et al [35].

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 Matlab MPI. 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 [18, 32]. Improvement may be realized via some quasi-Newtonian flow model with little additional computing effort [22]. 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 multi-block 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 [6] 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 et al. [29, 33], 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 domain decomposition or locally fitted stencil to implement the boundary conditions [8, 10, 12, 13, 15, 29, 33]. In this paper we will use the penalty method introduced by Caltagirone and co-workers [4] that is simpler to implement than our previous boundary fitted methods [10] and applies naturally to flow in a pipe with moving walls [26].

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:

$$\partial_t U + (U.\nabla)U + \nabla p - \nu\nabla.(\nabla U) = f, \text{ in } \Omega$$

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

$$U = \mathbf{g} \ \text{ on } \ \partial\Omega,$$

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. ν 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 Λ_{Ω_w}

$$\Lambda_{\Omega_w}(x, y) = 1, \text{ if } (x, y) \in \Omega_w, \ 0 \text{ elsewhere},$$

and is defined as follows

$$f = -\frac{1}{\eta}\Lambda_{\Omega_w} \ \{U - U_w(t)\}.$$

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 $S_w^f = \bar{\Omega}_f \bigcap \bar{\Omega}_w$ as $\eta \to 0$. Let us define the following expansion:

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

Formally we obtained at leading order,

$$\frac{1}{\eta}\Lambda_{\Omega_w} \ \{U_0 - U_w(t)\} = 0,$$

that is

$$U_0 = U_w, \text{ for } (x, y) \in \Omega_w.$$

The leading order terms U_0 and p_0 in the fluid domain Ω_f satisfy the standard set of NS equations:

$$\partial_t U_0 + (U_0.\nabla)U_0 + \nabla p_0 - \nu\nabla.(\nabla U_0) = 0, \text{ in } \Omega_f$$

$$div(U_0) \ = \ 0, \text{ in } \Omega.$$

At next order we have in Ω_w,

$$\nabla p_0 \ + \ U_1 \ + Q_w = 0, \tag{1}$$

where

$$Q_w \ = \ \partial_t U_w + (U_w.\nabla)U_w - \nu\nabla.(\nabla U_w).$$

Further the wall motion U_w must be divergence free.

At next order we have in Ω_f,

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

with

$$div(U_1) = 0.$$

In the simplest situation where $U_w \equiv 0$, we observe that 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 [25]. 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 [2] for fixed wall, i.e. $U_s \equiv 0$, that the convergence order of the penalty method is of order $\eta^{\frac{3}{4}}$, in the fluid domain, and $\eta^{\frac{1}{4}}$ 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 Ω_w^{low}, the fluid domain Ω_f and the upper wall Ω_w^{upper}.

Figure 4 and Figure 9 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

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

g is set to zero inside the wall, i.e for y such that $(0, y) \in \Omega_w$.

For simplicity we impose a homogeneous Neumann boundary condition on the flow field, i.e $\frac{\partial u_1}{\partial x} = \frac{\partial u_2}{\partial x} = 0$ 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 neighborhood of the vertical wall $x = 0$, and $x = L_x$ by multiplying the convective term with a smooth function $\mathcal{H}(x)$ that is one inside the interval $(d, L_x - d)$ and zero in the interval $(0, \frac{d}{2}) \bigcup (L_x - \frac{d}{2}, L_x)$. The momentum equation writes:

$$\partial_t U + \mathcal{H}(\vec{U}.\nabla)U - \nu\nabla.(\nabla U) = f, \text{ in } \Omega$$

The NS equations simplify then into the Stokes equation in the neighborhood 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 [3].

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 no matter 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 discretisation of the NS equations is done with finite differences on a Cartesian grid following the standard staggered grid method [24]. We define then the following grid function:

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

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

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

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

$$\frac{\partial C}{\partial t} = u_1(x,y,t)\frac{\partial C}{\partial x} + u_2(x,y,t)\frac{\partial C}{\partial y}, \tag{2}$$

We use the first order approximation in time,

$$C(x,y,t^{n+1}) = C(x-u_1(x,y,t^n)\,dt, y-u_2(x,y,t^n)\,dt, t^n). \tag{3}$$

at every grid points. Since the velocity components are defined at different grid point of the staggered grid, we use a second order bilinear interpolation to project the velocity components at the $(x-u_1(x,y,t^n)\,dt, y-u_2(x,y,t^n)\,dt)$ 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 centered finite differences, we choose a space step dt that satisfies the 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 neighborhood of the interface S_w^f.

The mask function Λ_{Ω_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 [5] and based on the Mumford-Shah Model. For completeness we are going to describe briefly this method. We refer to the review papers [21, 28] for a more comprehensive description of the level set method in the framework of image analysis. Let us denote $C(s)$ the unknown parameterized curve(s) that delineate the vessel. We assume that the unknown function(s) $C(s) : [0,1] \rightarrow IR^2$ 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: $\phi : \Omega \rightarrow IR$.

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

$$F(C, c_1, c_2) = \mu_1.(length(C)) + \mu_2.(area(inside(C))) + \tag{4}$$

$$\lambda_1 \int_{inside(C)} |u_o - c_1|^2 \, dx + \lambda_2 \int_{outside(C)} |u_o - c_2|^2 \, dx, \tag{5}$$

where

$$c_1(\phi) = \frac{\int_\Omega u_0 \, H(\phi) \, dx}{\int_\Omega H(\phi) \, dx},$$

$$c_2(\phi) = \frac{\int_\Omega u_0 \, (1 - H(\phi)) \, dx}{\int_\Omega (1 - H(\phi)) \, dx}.$$

H is the Heaviside function $H(z) = 1$, if $z \geq 0$, 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 supposed that the image is a piecewise constant function with values 0 and 1. The angiogram of an artery with a 50 percent 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 $u_0 = 1$. 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

$$\frac{\partial I}{\partial t} = N[I], \tag{6}$$

where N is the associated Euler Lagrange equation

$$N[\phi] = \frac{d}{d\, z} H(z)[\mu \; div \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \nu - \lambda_1 (I - c_1)^2 + \lambda_2 (I - c_2)^2], \; x \in \Omega.$$

In the numerical implementation one uses a regularization of the Heaviside function $H_\epsilon \in C^1(0,1)$ such that $H_\epsilon \rightarrow H$, as $\epsilon \rightarrow 0$, as well as a reinitialization procedure every few time steps that sharpens the level set function in the neighborhood 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 in general such that with $\lambda_1 >> 1$, 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 Λ_{Ω_w} in the momentum equation is consequently

$$\Lambda(x,y) = H(\Phi(x,y)),\ (x,y) \in \Omega.$$

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 [17]. Further, any artifact 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 $U \approx U_s$ 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 [36]. The time integration writes then

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

$$\frac{\hat{\mathbf{u}}^{k+1} - \mathbf{u}^{k,*}}{\Delta t} - \nu\Delta\mathbf{u}^{k} \tag{7}$$
$$= \mathbf{f}^{k+1} - \nabla p^{k} \text{ in } \Omega = (0, L_x) \times (0, L_y),$$

  or

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

  with boundary condition

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

  and

$$\mathbf{f}^{k} = -\frac{1}{\eta}\Lambda_{\Omega_s}\ \{u^{k} - U_s(t^{k})\}.$$

  $u^{k,*}$ is obtained with the method of characteristics (2, 3).

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

$$-div\nabla\delta p = -\frac{1}{\Delta t} div\, \hat{\mathbf{u}}^{k+1}, \tag{9}$$
$$\mathbf{u}^{k+1} = \hat{\mathbf{u}}^{k+1} - \Delta t\nabla\delta p, \quad p^{k+1} = p^{k} + \delta p. \tag{10}$$

We choose to use the semi-implicit scheme (8) instead of (7) 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 (7) does not require any solver. The semi-implicit scheme of (8) requires the solution of a linear system of equations corresponding to the discretization of the operator:

$$L = -dt\ \nu\ \Delta\ +\ c\ Id$$

where u is given from the previous time step, c is a coefficient that depends on the penalty term and Id is the identity operator. In practice $dt \approx min(h_x, h_y)$, $\eta \prec\prec 1$ and $\nu \preceq min(h_x, h_y)$. 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 [8, 10] 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 Domain Decomposition (DD) technique presented in [9, 12] that 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 [12] 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 consist in 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 [26], 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 $M(x, y) \in S_w^f$, 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_w^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_w^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 [14]. We recall that a real and even function $\sigma(\eta)$ is called a filter of order p if [14],

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

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

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

The filtered function u^σ is

$$u^\sigma(x, y, t) = \sum_{k_1=-n_x}^{n_x} \sum_{k_2=-n_y}^{n_y} \sigma\left(\frac{|k_1|\kappa}{n_x}\right) \sigma\left(\frac{|k_2|\kappa}{n_y}\right) \hat{u}_{k_1,k_2}(t)\ e^{i(k_1\tilde{x}+k_2\tilde{y})}. \qquad (11)$$

The parameter $\kappa \geq 0$ sets the level of cut in frequency space. As shown in [14], this filtering procedure efficiently reduces the impact of the Gibbs phenomenon on the accuracy of the approximation of a non smooth periodic function.

All point wise derivative $\frac{\partial u_{1/2}}{\partial x}$ and $\frac{\partial u_{1/2}}{\partial y}$ are computed with centered 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 $\frac{\partial u_{1/2}}{\partial x}$ and $\frac{\partial u_{1/2}}{\partial y}$ 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 $\alpha \in (0, \frac{\pi}{8})$ we use a linear extrapolation with the closest two grid values of the DI functions aligned along the vertical axis y. For $\alpha \in (3\frac{\pi}{8}, \frac{\pi}{2})$ we use the grid values aligned along the horizontal direction. For $\alpha \in (\frac{\pi}{8}, 3\frac{\pi}{8})$ we use the grid values at points (x_i, y_j) of the Cartesian grid aligned along the direction $y = -\frac{h_y}{h_x}x$.

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 $M(x, y) \in S_w^f$ 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 center $M(x, y)$ and radius d. For each flow velocity components one computes a second order polynomial approximation $P_M(x, y) = a_0 + a_1 x + a_2\, y + a_3\, x^2 + a_4\, y^2 + a_5\, x\, y$ that fits in the least square sense the flow field component on R_h. We notice 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 approximatively 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., $l \approx \sqrt{\nu}$, for small ν, 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

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 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 five per cent 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 (11).

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 $(0, \frac{\pi}{4})$.

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 gives 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.

These numerical experiments with Poiseuille flow might be used to calibrate the filter (11) that can be used later for simulations with pipes that have complex geometry.

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

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 $\approx \sqrt{\nu}$ seems optimal. Let us notice, 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 ν.

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 $0.6\ L_y$ obstructed by an obstacle that is the upper half of a disk. The radius of the disk is $0.3\ L_y$, and the obstacle mimics a fifty percent stenose. The center of the disc is at coordinates $(\frac{L_x}{2}, 0)$. We took $L_x = 3.5$, $L_y = 1$, and the viscosity is set to $\nu = 0.01$. 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 notice that the pressure may exhibit some sharp singular peaks at some of the grid points that are next to the wall of the disk. 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.

Figure 4 reports on a stiffer steady problem where $\nu = 0.003$ and the stenosis reaches 67%.

We see clearly the recirculation zone right after the obstacle. For smaller viscosity, we observe in our numerical experiments a secondary recirculation zone near the upper side of the wall downstream, and the solution becomes eventually unsteady.

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 [31]. This solution might be obtained with the method of separation of variable $u_1(0, y, t) = u_o(y)\ g(t)$ 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 5).

The geometry of the pipe is given in Figure 5 - postion 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:

$$U_w(x,t) = (0, u_{2w}), \text{ with} \tag{12}$$

$$u_{2w}(x,t) = U^o \left(exp\left(-\frac{(x - \frac{L_x}{2})^2}{\mu} \right) - (a\ x\ +\ b) \right) \cos(2\pi t).$$

The parameters a and b are chosen such that the wall at both ends of the pipe stays steady, i.e., $u_{2w}(0, y, t) = u_{2w}(L_x, y, t) = 0$. 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 approximatively $\frac{U^o}{\pi}$. Figure 5 shows the two extreme positions of the pipe in this oscillatory motion of the wall. Figure 6, (respectively Figure 7) shows the shear stress computed on the lower wall for the wall position as in Figure 5 (respectively, the wall oscillations between position A and B at speed (12)). We observe a periodic solution in time that has a strong pique in space at the location of the stenose. The periodic motion of the wall adds a 12 percent increase on the shear stress versus the solution with static wall as in position A. As seen in the contour plot of Figure 8, 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.

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 has roughly a one millimeter 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 [5]. One artifact in this image segmentation corresponds to an annotation of the angiogram with an arrow. A second artifact 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 $x = 0$. Our flow server based on the L_2 penalty method does not suffer from the two artifacts as shown in Figure 10. The result of the image segmentation is also relatively sensitive to the choice of the parameters $\lambda_1,\ \lambda_2, \mu_1,\ \mu_2$ in (4,5). The sensitivity of the flow simulation to the parameter of the image segmentation can be analyzed 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 remains 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, that 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 solver of the NS flow with complex geometry. To address this issue, we next present a parallel implementation of our NS code.

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 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 [35]. 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 [19]. MatlabMPI [35] 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 send 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 Multi Processor (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 [35] 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 ways 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 send 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 (9).

We use the analytic additive Aitken-Schwarz algorithm [9] 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 [9].

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 block-wise Schwarz relaxation. Because the eigenvectors of the (linear) trace transfer operator are known, this acceleration provides the *exact* interface condition in one single step no matter 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 consist of repeating the following iteration:

$$\begin{gathered} \Delta p_1^{n+1} = RHS \ \text{ in } \ \Omega_1, \ \ \Delta p_2^{n+1} = RHS \ \text{ in } \ \Omega_2, \\ p_{1|\Gamma_1}^{n+1} = p_{2|\Gamma_1}^{n}, \ \ p_{2|\Gamma_2}^{n+1} = p_{1|\Gamma_2}^{n}, \end{gathered} \tag{13}$$

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:

$$p(y)_{|\Gamma_i} = \sum_{k=1..N_y-1} \hat{p}^k_{|\Gamma_i} cos(k\pi y)\ ,\ \ i = 1\ \text{ or }\ 2, \tag{14}$$

provides a diagonalization of the trace transfer operator:

$$\left(p^n_{1|\Gamma_1}, p^n_{2|\Gamma_2}\right) \stackrel{T}{\rightarrow} \left(p^{n+1}_{1|\Gamma_1}, p^{n+1}_{2|\Gamma_2}\right). \tag{15}$$

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

$$\frac{\partial^2 \hat{p}_k(x)}{\partial x^2} - \mu_k\ \hat{p}_k(x) = \widehat{RHS}_k,\ \ \forall k. \tag{16}$$

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

$$\left(\hat{p}^{n,k}_{1|\Gamma_1} - \hat{p}^k_{\Gamma_1}, \hat{p}^{n,k}_{2|\Gamma_2} - \hat{p}^k_{\Gamma_2}\right) \stackrel{T_k}{\rightarrow} \left(\hat{p}^{n+1,k}_{1|\Gamma_1} - \hat{p}^k_{\Gamma_1}, \hat{p}^{n+1,k}_{2|\Gamma_2} - \hat{p}^k_{\Gamma_2}\right),\ \ \forall k. \tag{17}$$

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

$$\hat{p}^{n+1,k}_{1|\Gamma_2} - \hat{p}^k_{\Gamma_2} = \hat{\delta^1_k}\left(\hat{p}^{n,k}_{2|\Gamma_1} - \hat{p}^k_{\Gamma_1}\right),\ \ \hat{p}^{n+1,k}_{2|\Gamma_1} - \hat{p}^k_{\Gamma_1} = \hat{\delta^2_k}\left(\hat{p}^{n,k}_{1|\Gamma_2} - \hat{p}^k_{\Gamma_2}\right), \tag{18}$$

where $\hat{\delta^1_k}$ and $\hat{\delta^2_k}$ 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 (18) where $n = 0$, of unknown $(\hat{p}^k_{\Gamma_1}, \hat{p}^k_{\Gamma_1})$. The same procedure applies with an arbitrary number of subdomains but then the matrix of the linear system corresponding to (18) 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 (9) we must add a preprocessing step that consist of solving the zero mode equation:

$$\frac{\partial^2 \hat{p}_0(x)}{\partial x^2} = \widehat{RHS}_0, \tag{19}$$

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

To summarize, the algorithm writes:

- prelim. 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 (9), the prediction and correction steps of the projection scheme are also parallelized: this implies one extra communication process with the neighboring 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 [34] to solve each subdomain. The operator matrix M is decomposed into a unit lower triangular matrix L, a 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 notice 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 steps. The UMFPACK decomposition provides particularly good performance in Matlab compare 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.

Figure 12 and Figure 13 reports 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 4 processors, and 283×1129 on 8 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 a NS solver in general grows faster than linearly with respect to the number of unknowns.

Let us notice 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 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 seconds on 8 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 neighbor subdomains only. We refer to the optimized implementation presented in [1] 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 in this paper 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 Caltagirone and co-workers (83). 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 provide 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 to easily access 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 [13].

Our next step is, however, to generalize the present work to simulation in three space dimensions by extending the capability of our parallel 3D NS solver presented in [8, 10] to blood flow simulation related to real clinical cases.

Thanks: We would like to thanks Chandler Wilkerson for his help on the MatlabMPI setting.

References

[1] N. Barberou, M.Garbey, M.Hess, M. Resch, T. Rossi, J.Toivanen and D.Tromeur Dervout, *On the Efficient Meta-computing of linear and nonlinear elliptic problems,* Journal of Parallel and Distributed Computing- special issue on grid computing, Vol63, Issue 5, pp 564-577, 2003.

[2] P.Angot, C.H.Bruneau and P.Fabrie, *A Penalisation Method to Take into Account Obstacles in Viscous Flows,* Numerische Mathematik, 81, pp 497-520, 1999.

[3] C.H.Bruneau, *Boundary Conditions on Artificial Frontiers for Incompressible and Compressible Navier Stokes Equations,* M2AN, Vol 34, No2, pp 303-314, 2000.

[4] E.Arquis and J.P. Caltagirone, *Sur Les Conditions Hydrodynamiques au Voisinage d'une Interface Milieu Fluide-Milieux Poreux: Application a la Convection Naturelle,* CRAS, Paris II, 299, pp1-4, 1984.

[5] T.F.Chan and L.A.Vese, *Active Contours without Edges,* IEE Transaction on Image Processing, 10 (2), pp 266-277, 2001.

[6] M.O.Deville, P.F.Fischer and E.H.Mund, *High Order Methods for Incompressible Fluid Flow,* Cambridge Monographs on Applied and Computational Mathematics, 2002.

[7] D. Gottlieb and C.W.Shu, *On the Gibbs Phenomenon and its Resolution*, SIAM Review, Vol 39, No4,pp 644-668, 1997.

[8] M. Garbey and Y.A. Kuznetsov and Y.V. Vassilevski, *A Parallel Schwarz Method for a Convection-Diffusion Problem,* SIAM J. Sci. Comput., Vol 22, No 3, 2000.

[9] M.Garbey and D. Tromeur Dervout, *On some Aitken like acceleration of the Schwarz Method,*, Int. J. for Numerical Methods in Fluids. 40 (12), pp 1493-1513, 2002.

[10] M.Garbey and Yu.V.Vassilevski, *A Parallel Solver for Unsteady Incompressible 3D Navier-Stokes Equations,* J. Parallel Computing, 27, pp363–389, 2001.

[11] M.Garbey and W.Shyy, *A Least Square Extrapolation Method for the A Posteriori Error Estimate of the Incompressible Navier Stokes Problem,* Int. J. for Numerical Methods in Fluids, No 48, pp 43-59, 2005.

[12] M.Garbey, B.Hadri and W.Shyy, *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, AIAA-2005-1386, 2005.

[13] M.Garbey, B.O.Dia, C.Picard and R.Tran Son Tay, *Heterogeneous domain decomposition for multi-scale problems,* 43rd Aerospace Sciences Meeting and Exhibit Conference, Reno January 2005, paper number: AIAA-24880

[14] D. Gottlieb and Chi-Wang Shu, *On the Gibbs Phenomenon and its Resolution,* SIAM review, Vol39, No 4, 644-668, 1997.

[15] R. Glowinski, *A fictitious Domain Approach to the Direct Numerical Simulation of Incompressible Flow Past Moving Rigid Bodies: Application to Particulate Flow,* JCP, Vol 162, pp 363-426, 2001.

[16] J.Kepner and S.Ahalt, *Matlab Mpi,* Journal of Parallel and Distributed Computing, Vol 64 (8), pp 997-1005, 2004.

[17] D.Metaxas, *Physics-Based Deformable Models: Application to Computer Vision, Graphics and Medical Imaging,* Kluwer-Academic Publishers, Nov. 1996.

[18] D. A. McDonald, *Blood Flow in Arteries,* Edward Arnold, third edition, 1990.

[19] MPI Forum, *MPI: A Message-Passing Interface Standard,* Document for a Standard Message-Passing Interface, University of Tennessee, 1994.

[20] W.L. Oberkampf, F.G. Blottner and D.Aeshliman, *Methodology for Computational Fluid Dynamics Code Verification and Validation*, 26th AIAA Fluide Dynamic Conference, June 19-22, 1995/ San Diego, CA, AIAA Paper 95-2226.

[21] S.Osher and N. Paragios, *Geometric Level Set Methods in Imaging, Vision and Graphics,* Springer Verlag, ISBN 0387954880, 2003.

[22] K. Perktold and M. Resch and H. Florian,*Pulsatile Non-Newtonian Flow Characteristics in a Three Dimensionnal Human Carotid Bifurcation Model,* ASME Journal of Biomechanical Engineering, Vol 113, pp 464-475, 1991.

[23] C.S.Peskin, *The Immersed Boundary Method,* Acta Numerica pp1-39, 2002.

[24] R.Peyret and T.T.Taylor, *Computational Methods for Fluid Flow,* Springer series in Computational Physics, 1985.

[25] A. Quarteroni, A. Veneziani, P. Zunino, *Mathematical and Numerical Modelling of Solute Dynamics in Blood Flow and Arterial Walls,* SIAM J. Numer. Anal., Vol 39, No. 2, pp. 1488-1511, 2002.

[26] K. Schneider and M.Farge, *Numerical Simuation of the Transient Flow Behavior in Tube Bundles Using a Volume Penalization Method,* Journal of Fluids and Structures, 20, pp 555-566, 2005.

[27] H.A.Schwarz, *Gesammelte Mathematische Abhandlungen,*2, 2nd ed., Springer Berlin, pp.133-143, 1980. *Vierteljahrsschrift der Naturforschenden Gesellschaf,* Zurich, Vol15, 1st ed. pp.272-286, 1870.

[28] J.A.Sethian, *Level Set Methods and Fast Marching Methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science,* Cambridge Monographs on Computational Mathematics, P.G.Ciarlet, A.Iserles, R.V.Kohn and M.H.Wright editors, 1999.

[29] W. Shyy, *Moving Boundaries in Micro-Scale Biofluid Dynamics,* Appl. Mech. Rev., Vol.54 No 5, pp 405-453, 2001.

[30] B. Smith, P. Bjorstad and W.Gropp, *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations,* Cambridge University Press, 1996.

[31] J.R.Womersley, *Method for the Calculation of Velocity, Rate of Flow and Viscous Drag in Arteries when the Pressure Gradient is Known,* J.Physiol. 127, pp 553-563, 1955.

[32] X. Y. Xu and M. W. Collins and C. J. H. Jones, *Flow Studies in Canine Aortas,* ASME J. of Biomech. Eng., Vol 114, No 11, pp 505-511, 1992.

[33] T. Ye and R.Mittal and H. S. Udaykumar and W. Shyy, *An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries,* JCP, Vol 156, pp 209-240, 1999.

[34] T. A. Davis and Iain S. Duff, *An Unsymmetric-pattern multifrontal method for sparse LU factorization,* SIAM J. Matrix Anal. Apll., Vol 18-1, pp 140-158, 1997.

[35] J.Kepner and S. Ahalt, *MatlabMPI,* Journal of Parallel and Distributed Computing, Vol 64-8, pp 997-1005, 2004.

[36] A.J. Chorin, *The numerical solution of the Navier-Stokes equations for an incompressible fluid,* Bull. Amer. Math. Soc., Vol 73, pp 928, 1967.

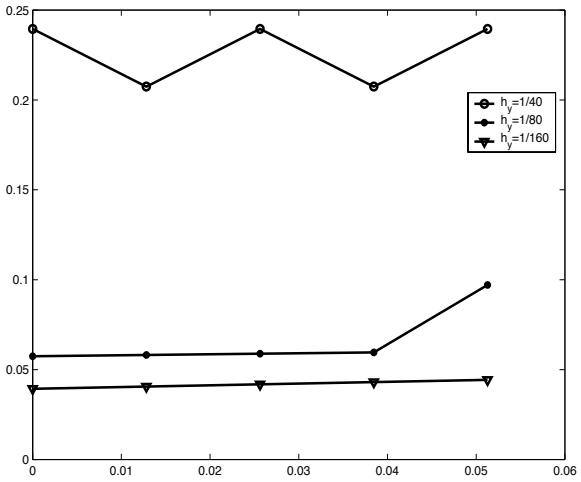

Figure 1: Sensitivity of the computation of the shear stress on the wall as a function of the horizontal position with *Method A*

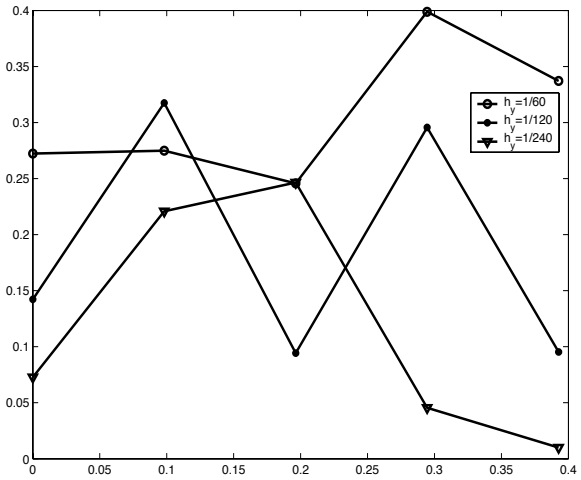

Figure 2: Sensitivity of the computation of the shear stress on the wall as a function of the angle α with *Method A* but no filtering case.

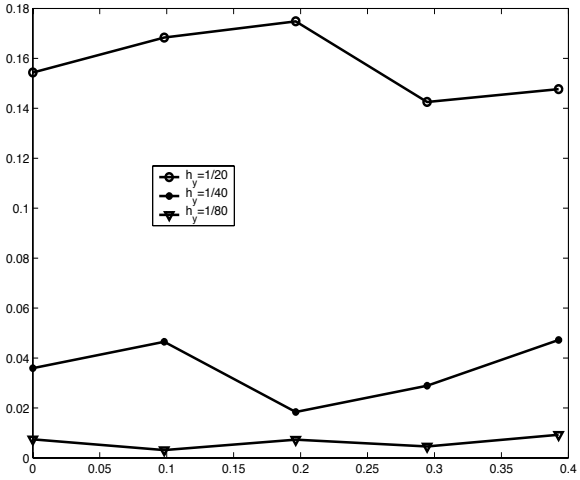

Figure 3: Sensitivity of the computation of the shear stress on the wall as a function of the angle α with *Method B.*

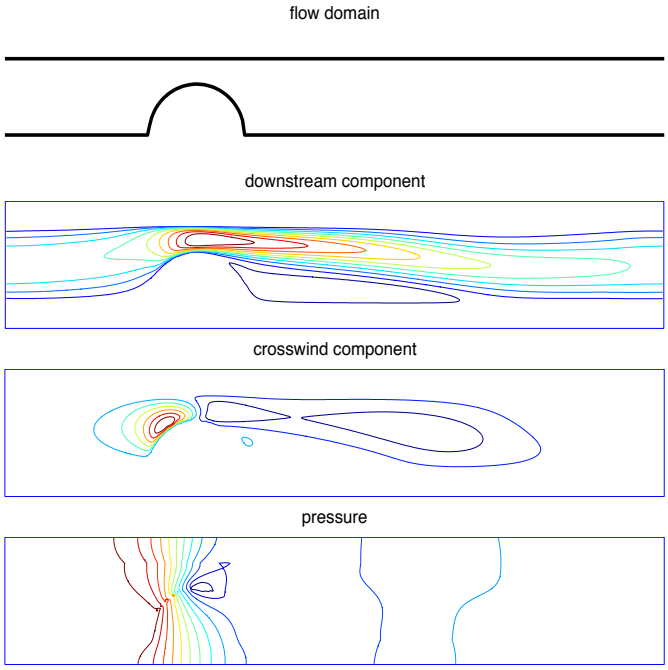

Figure 4: Steady flow with a 67% stenosis.

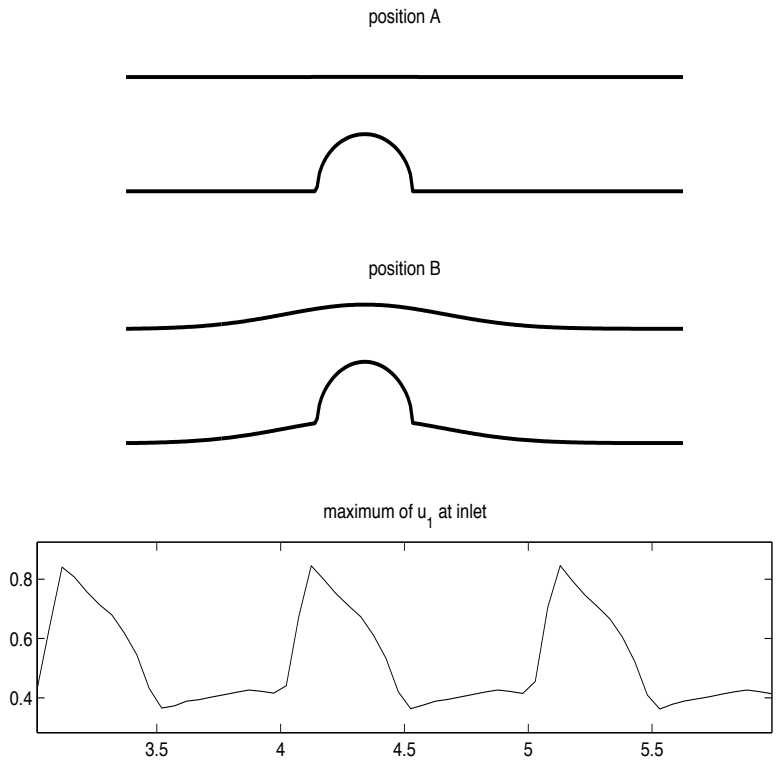

Figure 5: Pulsating flow problem.

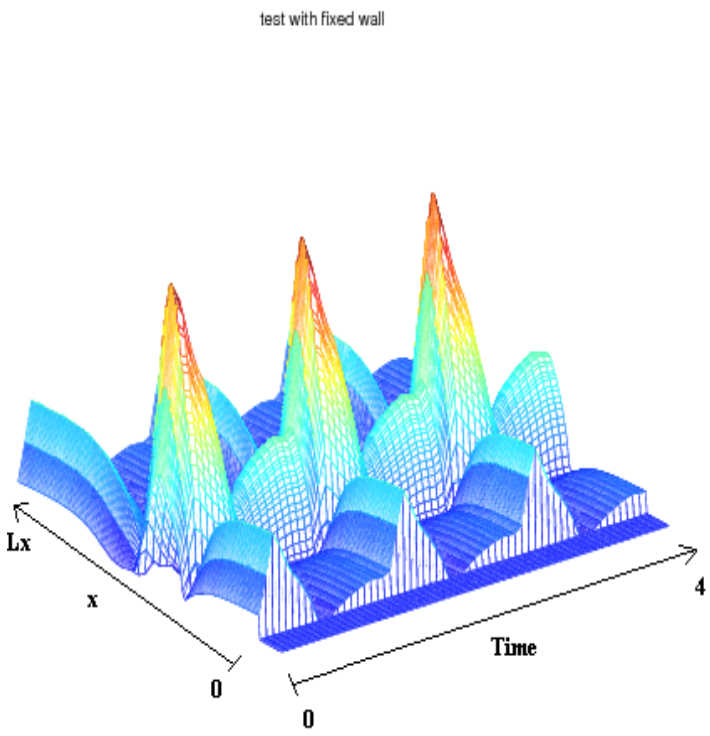

Figure 6: Shear stress on the lower wall.

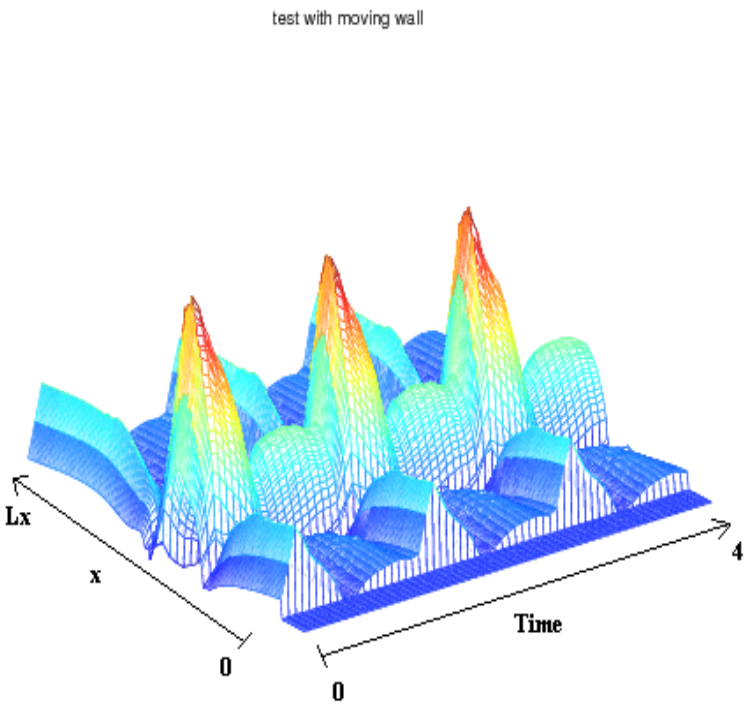

Figure 7: Shear stress on the *moving* lower wall.

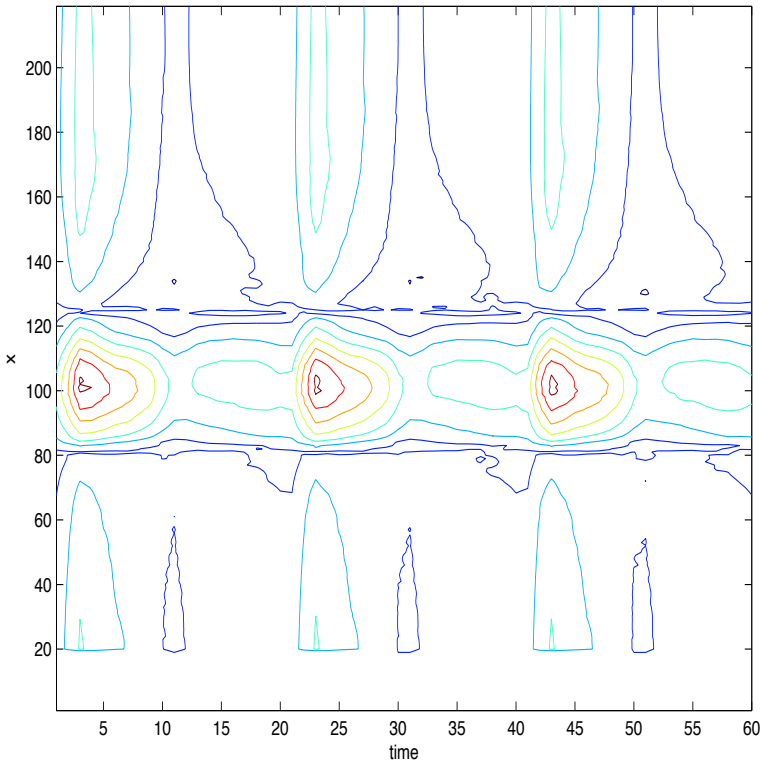

Figure 8: Contour plot of the Shear stress on the moving lower wall.

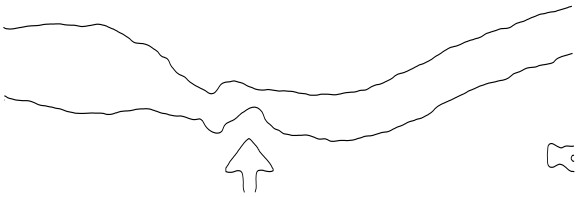

Figure 9: Image segmentation.

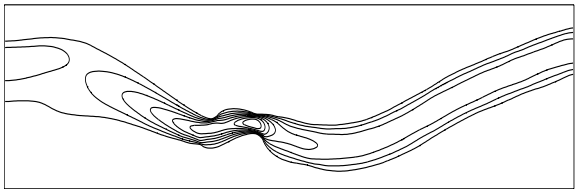

Figure 10: Flow amplitude for a stationary problem.

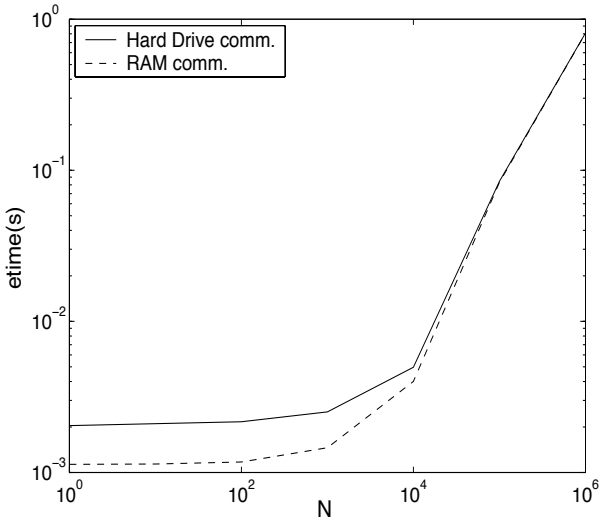

Figure 11: Clock time for one message-passing in MatlabMPI using either a communication directory on the hard drive or in main memory.

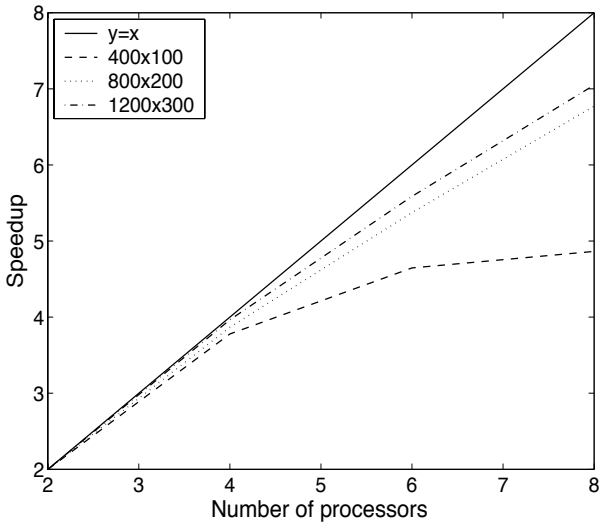

Figure 12: Parallel speed-up for the parallel Navier-Stokes code.

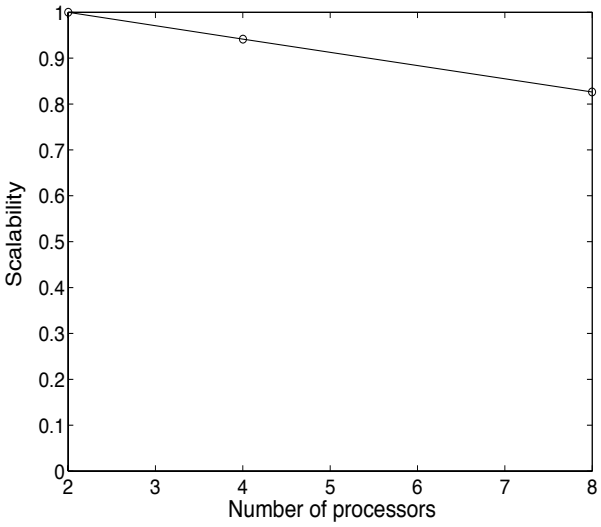

Figure 13: Parallel scalability for the Navier-Stokes code.