Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 7123-7145 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

A FFT PRECONDITIONING TECHNIQUE FOR THE SOLUTION OF INCOMPRESSIBLE FLOW WITH FRACTIONAL STEP METHODS ON GRAPHIC PROCESSING UNITS

Mario A. Storti[a], Rodrigo R. Paz[a], Lisandro D. Dalcín[a] and Santiago Costarelli[b]

[a]*Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), INTEC(CONICET-UNL), Predio CONICET-Santa Fe Colect. Ruta Nac. 168 Km 472, Paraje El Pozo, 3000 Santa Fe, Argentina {mario.storti dalcinl }@gmail.com, rodrigop@intec.unl.edu.ar, http://www.cimec.org.ar/mstorti*

[a,b]*Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral, Paraje El Pozo, 3000 Santa Fe, Argentina, santi.costarelli@gmail.com, http://www.fich.unl.edu.ar*

Keywords: Graphics Processing Units, Poisson equation

Abstract. The resolution of Computational Fluid Dynamics (CFD) problems on Graphic Processing Units (GPU's) requires of specialized algorithms due to the particular hardware architecture of these devices. Algorithms that fall in the category of cellular automata (CA) are the best fitted, for instance explicit Finite Volume or Finite Element methods. But in the case of incompressible flow it is not possible to develop a pure explicit algorithm, due to the essentially non-local character of the incompressibility condition. In this case the algorithms that are closer to an explicit approach, are *segregated* algorithms, like the *Fractional Step Method.* In these algorithms the more time consuming stage is (asymptotically for large problems) the solution of the Poisson's equation for pressure. A common choice for it's solution is the *IOP* (Iterated Orthogonal Projection) method, which requires a series of solutions on the complete mesh. In this work a variant of the *IOP*, called *Accelerated Global Preconditioning (AGP)*, is proposed. It is based on using a *Preconditioned Conjugate Gradient* (which is an *accelerated* iterative method, in contrast with the *stationary* scheme used in *IOP*) for the pressure on the fluid, and preconditioning with the solution on the *global* domain (fluid and solid). Of course, solving the problem on the global domain represents more computational work than solving the problem only in the fluid, but this can be faster in a structured mesh if a fast solvers as *Multigrid* or *Fast Fourier Transform (FFT)* is used. The main advantage of *AGP* over *IOP* is that it is an accelerated solver, whereas the *IOP* is stationary. In addition *AGP* iterates only on pressure, whereas *IOP* iterates on both pressure and velocity.

1 INTRODUCTION

GPU's (for *Graphics Processing Units)* are video boards of common use today in desktop PC's and graphical stations. They are computer processors specially oriented to reduce the load on the *CPU* from the renderization of complex graphics. With time they have evolved to complex systems and today they may contain many multiprocessors and a high amount of RAM on the board. They have a large nominal computing power (in the order of Teraflops) based on a massively parallel architecture and the paradigm of *SIMD (Single Instruction Multiple Data)*, i.e. the programmer writes *kernel* functions that perform a series of simple instructions over a large set of similar data (Molemaker et al., 2008; Bell and Garland, 2009).

Recently scientists and engineers have started to explore the possibility of using GPU's for HPC (High Performance Computing) (Molemaker et al., 2008; Ryoo et al., 2008; Mullen et al., 2009; Elsen et al., 2008; Goddeke et al., 2008; Lastra et al., 2009; Corrigan et al., 2010; Thibault and Senocak, 2009; Adams et al., 2007). This tendency has motivated that GPU manufacturers like Nvidia or ATI have begin to produce *General Purpose GPU's (GPGPU's)*, like the Nvidia Tesla, for instance, that are specially oriented to HPC.

On the other hand the renderization codes used in video games and special effects have increased constantly its complexity and today it is not uncommon to find *physical engines* (i.e. libraries that allow the programmer to simulate physical objects and its interaction with the environment) that solve PDE's (for Partial Differential Equations) as, for instance, the Navier-Stokes equations, in order to obtain visualizations more realistic (Elcott et al., 2008; Irving et al., 2006; Crane et al., 2008; P.Rinaldi et al., 2008; Wu et al., 2004). These codes are usually very fast, as compared with the codes used for engineering purposes, in part due to the use of GPU's, but also because they sacrifice *precision for speed*. For instance it is very common the employment of *embedded structured meshes* of constant step size. In this kind of scheme curved boundaries are represented by *staircase* (also known as *Lego*) surfaces. This is acceptable for visualization purposes, but the order of convergence of the scheme for heat or momentum transfer from such a surface may be greatly impaired, or even may be not convergent at all, which is invalidating for the application in an engineering code. Another point is the stopping criteria employed for the convergence of the iterative algorithms. The tolerances used in visualization codes are very lax compared to the needs for engineering codes. At the end this is not simply a change in a parameter in the computer code run, but it may change the whole algorithm. For instance *IOP (for Iterated Orthogonal Projection)* iterates fastly and is used frequently with very low iteration number (1 to 3 iterations typically) which results in poorly converged solutions. A new algorithm that has accelerated convergence and can be used practically to much lower tolerances is proposed in this article.

2 THE ITERATED ORTHOGONAL PROJECTION (IOP) SOLVER

The Navier-Stokes governing equations are

$$\begin{aligned} \frac{\partial u_i}{\partial t} + \frac{\partial (u_j u_i)}{\partial x_j} &= -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \Delta u_i, \\ \frac{\partial u_j}{\partial x_j} &= 0, \end{aligned} \tag{1}$$

where u_i are the components of velocity, ρ density, p is pressure, Δ is the Laplace operator, x_j are the spatial coordinates, and t is time.

The numerical scheme is based on a Fractional Step-like solver, using the *QUICK* (for

Quadratic Upstream Interpolation for Convection Kinematics, see Leonard (1979)) on a staggered grid. *QUICK* is an advection scheme with very low numerical dissipation and is well suited for structured finite difference schemes.

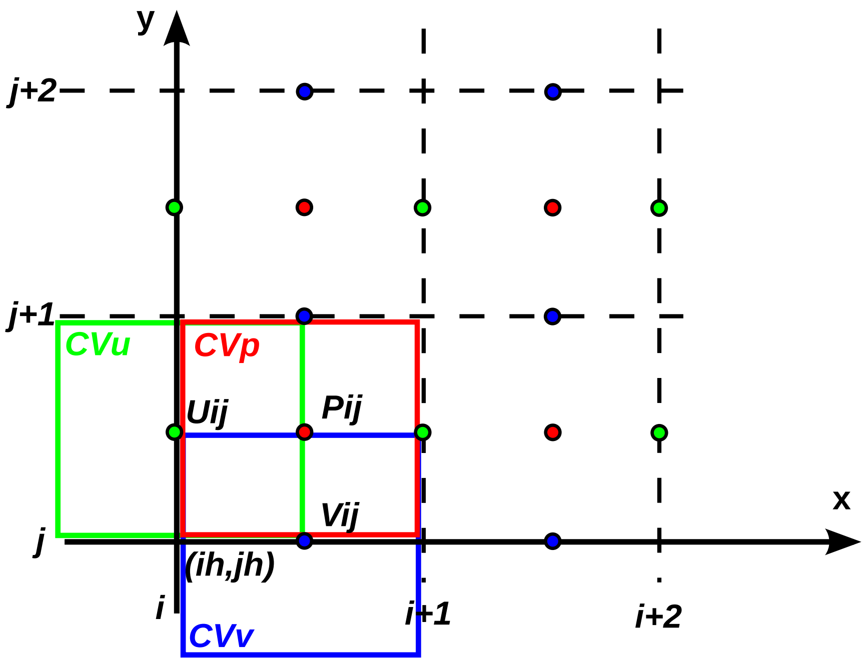

Figure 1: Staggered scheme (2D). Continuity cells (CVp, in red) are centered at the pressure nodes $\mathbf{x} = (i, j, k)h$. x-momentum cells (CVu, in blue) are centered at the u nodes $\mathbf{x} = (i, j + 1/2, k + 1/2)h$.

A rectangular box $0 \leq x/L_x, y/L_y, x/L_z \leq 1$ is discretized with $N_x \times N_y \times N_z$ continuity cells. The mesh size $h = L_j/N_j$ is assumed to be the same for all the spatial directions (see figure 1). The mesh is *staggered*, so that cells for the discrete balance of continuity and each of the momentum equations do not coincide. The continuity cells are centered around $\mathbf{x} = (i+1/2, j+1/2, k+1/2)h$ positions, and pressure values are assumed to be positioned at the center of these cells. The cells for x-momentum are shifted $h/2$ in the x direction, i.e. they are centered at $\mathbf{x} = (i, j + 1/2, k + 1/2)h$, and similarly, *mutatis mutandis*, for the other directions. Periodic boundary conditions on the external boundaries will be assumed for simplicity. Internal solid bodies will be treated as embedded and the details will be discussed later.

2.1 The predictor step. QUICK scheme

In the first stage, the velocity field $\mathbf{u}^n$ is advanced to an intermediate state $\mathbf{u}^{n+1,p}$

$$\frac{\mathbf{u}^{n+1,p}(\mathbf{x}) - \mathbf{u}^n(\mathbf{x})}{\Delta t} = (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \mathbf{g}. \tag{2}$$

where the supraindex p stands for *predictor*. This predicted field may be not divergence free, so that it is corrected via a Poisson stage (or *IOP*, which is similar) to be explained later. The *QUICK* implementation of the predictor stage is discussed in this section.

Let's consider the x component of the balance equation. For the discretization of the x-momentum balance the corresponding x-momentum cell is used, which is shifted $h/2$ in the x direction, as described before. The discrete equation is obtained by a *Finite Volume Method*

(FVM) approach, i.e. by performing a momentum balance on the cell as

$$\Omega \left(\frac{\mathbf{u}^{n+1,p}(\mathbf{x}) - \mathbf{u}^n(\mathbf{x})}{\Delta t} \right)_{(i,j+1/2,k+1/2)} = M^n_{x,(i,j+1/2,k+1/2)} + \Omega\, \mathbf{g}_{(i,j+1/2,k+1/2)}. \tag{3}$$

where Ω is the cell volume. Note that all terms are evaluated at the center of the x-momentum cells. The temporal and external force field terms are straightforward. The nonlinear convection term is evaluated as

$$\begin{aligned} M_{x,(i,j+1/2,k+1/2)} = S_x & \left[(u^c u^Q)_{i+1/2,j+1/2,k+1/2} - (u^c u^Q)_{i-1/2,j+1/2,k+1/2} \right] \\ & + S_y \left[(v^c u^Q)_{i,j+1,k+1/2} - (v^c u^Q)_{i,j,k+1/2} \right] \\ & + S_z \left[(w^c u^Q)_{i,j+1/2,k+1} - (w^c u^Q)_{i,j+1/2,k} \right], \end{aligned} \tag{4}$$

where the superscripts c and Q stand for *centered* and *QUICK* respectively, and the supraindex n has been dropped since all quantities are evaluated in t^n. The rationale behind this is that each of the terms represent the flux of momentum through a cell face. Each contribution involves the product of the velocity normal to the surface (which is approximated *centered*) with the velocity component which is being advected (which is approximated *QUICK*-upwinded). In Eq. (4) the advected component is always u (since the x-momentum is considered) whereas the normal velocity may be u, v, or w, depending on the face of the cell to be considered. Note that in the first term, different approximations (centered or *QUICK*) are used for the same component u depending on whether it is used as a normal velocity or an advected component.

The centered approximations are

$$\begin{aligned} u^c_{i+1/2,j+1/2,k+1/2} &= 1/2(u_{i,j+1/2,k+1/2} + u_{i+1,j+1/2,k+1/2}), \\ v^c_{i,j,k+1/2} &= 1/2(v_{i-1/2,j,k+1/2} + v_{i+1/2,j,k+1/2}), \\ w^c_{i,j+1/2,k+1} &= 1/2(w^c_{i-1/2,j+1/2,k+1} + w^c_{i+1/2,j+1/2,k+1}). \end{aligned} \tag{5}$$

whereas the *QUICK* upwinded approximations are

$$\begin{aligned} u^Q_{i-1/2,j+1/2,k+1/2} &= \begin{cases} (c_0 u_i + c_1 u_{i-1} + c_2 u_{i-2})_{j+1/2,k+1/2}, & \text{if } u^c_{i-1/2,j+1/2,k+1/2} > 0, \\ (c_0 u_{i-1} + c_1 u_i + c_2 u_{i+1})_{j+1/2,k+1/2}, & \text{if } u^c_{i-1/2,j+1/2,k+1/2} < 0, \end{cases} \\ u^Q_{i,j,k+1/2} &= \begin{cases} \left(c_0 u_{j+1/2} + c_1 u_{j-1/2} + c_2 u_{j-3/2}\right)_{i,k+1/2}, & \text{if } v^c_{i,j,k+1/2} > 0, \\ \left(c_0 u_{j-1/2} + c_1 u_{j+1/2} + c_2 u_{j+3/2}\right)_{i,k+1/2}, & \text{if } v^c_{i,j,k+1/2} < 0, \end{cases} \\ u^Q_{i,j+1/2,k} &= \begin{cases} \left(c_0 u_{k+1/2} + c_1 u_{k-1/2} + c_2 u_{k-3/2}\right)_{i,j+1/2}, & \text{if } w^c_{i,j+1/2,k} > 0, \\ \left(c_0 u_{k-1/2} + c_1 u_{k+1/2} + c_2 u_{k+3/2}\right)_{i,j+1/2}, & \text{if } w^c_{i,j+1/2,k} < 0. \end{cases} \end{aligned} \tag{6}$$

The coefficients c_j are $c_0 = 3/8$, $c_1 = 6/8$, $c_1 = -1/8$. They are the basis of *QUICK* and guarantee that the upwinded approximations are precise to third order.

2.2 The projection step in FSM

Once the predicted field $\mathbf{u}^{n+1,p}(\mathbf{x})$ is computed, it may not satisfy the divergence condition, neither the boundary conditions

$$(\mathbf{u}^{n+1,p} - \mathbf{u}_{\mathrm{bdy}}) \cdot \hat{\mathbf{n}} = 0, \quad \text{at } \Gamma_{\mathrm{bdy}}, \tag{7}$$

where $\mathbf{u}_{\mathrm{bdy}}$ is the velocity of the body Γ_{bdy} and $\hat{\mathbf{n}}$ its normal. In the standard Fractional Step method this conditions are enforced by computing a Poisson stage

$$\mathbf{u}^{n+1} = \mathbf{u}^{n+1,p} - \nabla P, \tag{8}$$

where $P = (\Delta t/\rho)p$ and p is pressure. P is computed through the following Poisson equation

$$\begin{aligned} \Delta P &= \nabla \cdot \mathbf{u}^{n+1,p}, \\ \left.\frac{\partial p}{\partial n}\right|_{\Gamma_{\mathrm{bdy}}} &= (\mathbf{u}_{\mathrm{bdy}} - \mathbf{u}^{n+1,p}) \cdot \hat{\mathbf{n}}. \end{aligned} \tag{9}$$

An alternative form to enforce these conditions is the *Iterated Orthogonal Projection (IOP)* method. Instead of computing and intermediate pressure field in order to enforce the divergence conditions, *IOP* directly iterates on the velocity field, i.e. it constructs a sequence $\mathbf{w}^k$ such that $\mathbf{w}^0 = \mathbf{u}^{n+1,p}$ and $\mathbf{w}^\infty = \mathbf{u}^{n+1}$. The sequence is generated via the consecutive application of the affine orthogonal projection operators $\mathbf{\Pi}_{\mathrm{div}}$ and $\mathbf{\Pi}_{\mathrm{bdy}}$,

$$\mathbf{w}^{k+1} = \mathbf{\Pi}_{\mathrm{bdy}}\mathbf{\Pi}_{\mathrm{div}}\mathbf{w}^k. \tag{10}$$

The projection operators are defined by

$$\begin{aligned} \mathbf{u}' = \mathbf{\Pi}_{\mathrm{div}}(\mathbf{u}) \quad &\Longrightarrow \quad \begin{cases} \mathbf{u}' = \mathbf{u} - \nabla P, \\ \Delta P = \nabla \cdot \mathbf{u}, \end{cases} \\ \mathbf{u}'' = \mathbf{\Pi}_{\mathrm{bdy}}(\mathbf{u}') \quad &\Longrightarrow \quad \begin{cases} \mathbf{u}'' = \mathbf{u}_{\mathrm{bdy}}, & \text{in } \Omega_{\mathrm{bdy}}, \\ \mathbf{u}'' = \mathbf{u}', & \text{in } \Omega_{\mathrm{fluid}}. \end{cases} \end{aligned} \tag{11}$$

2.3 Convergence of IOP iteration

One problem with *IOP* iteration is that the rate of convergence is linear, because it is a stationary method. For instance Figures 2 and 3 show the convergence of *IOP* iteration for a 16x16x16 and 64x64x64 meshes on a computational domain which is a cube of unit side L. The body domain is a circle of radius 0.3, i.e. $\Omega_{\mathrm{bdy}} = \{||\mathbf{x}|| \leq R = 0.3\}$.

However note that the rates of convergence are independent of mesh refinement. The convergence for both meshes start at a high convergence rate of less than 3 iter/OM (order of magnitude), and then it switches to a slower convergence of less than 15 iter/OM. The convergence rates are very close for the fine and coarse mesh, i.e. they are almost independent of mesh refinement.

3 THE ACCELERATED GLOBAL PRECONDITIONING

The proposed method is a preconditioning for embedded problems that is based on solving the problem in the complete mesh. It is similar to *IOP* in the sense that the whole strategy is based on the fact that there is a very efficient solution *on the global domain*, i.e. including the solid bodies; it can be either a multigrid or FFT solution.

Consider a situation like in figure 4, with a solid body described by the boundary Γ_{bdy}. This is embedded in a structured grid of constant mesh size h. In a Fractional Step Method a Poisson problem is solved *outside* the body, so that this is done by *assembling* the matrices of those cells that are in the fluid region. To do that the center of the cells are computed and it is checked whether the cell falls inside or outside the body. In this way the body is approximated by a

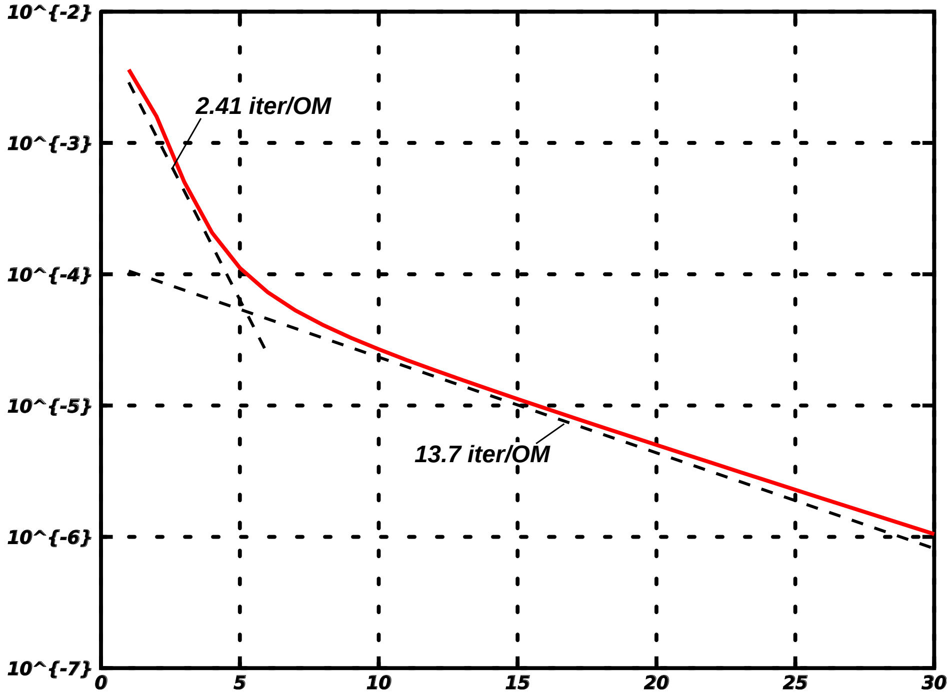

Figure 2: Convergence of *IOP* loop for a sphere of $R = 0.3$ in a cube of $L = 1$, with a coarse mesh of 16x16x16

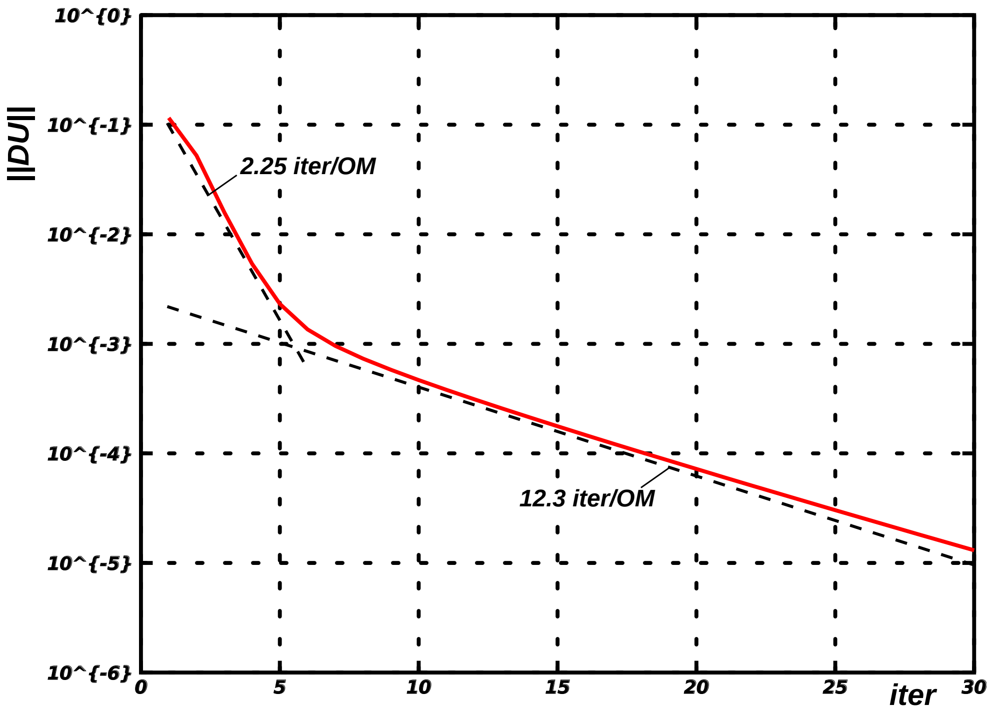

Figure 3: Convergence of *IOP* loop for a sphere of $R = 0.3$ in a cube of $L = 1$, with a coarse mesh of 64x64x64

staircase geometry as is shown in gray in the figure. In a FEM context the imposition of the homogeneous Neumann condition is done by simply assembling only those elements that are in the fluid part (filled in gray in the figure). The other elements that are not in gray are *ghost elements* and are not assembled for the solution of the Poisson problem. Only the pressure in the nodes connected to some element that is assembled are relevant, i.e. those that are marked in blue and red. Those that are marked in green are *ghost* and are not computed. From those that are computed, the set that are surrounded completely by computed elements (and then are not connected to ghost elements) are classified as *interior to the fluid*, (subindex F) and the rest are classified as *boundary* (subindex B, filled in red in the figure). So the Poisson problem is

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{12}$$

and the splitting of nodes induces a matricial splitting like this

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \mathbf{A}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_F \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_F \\ \mathbf{b}_B \end{bmatrix} \tag{13}$$

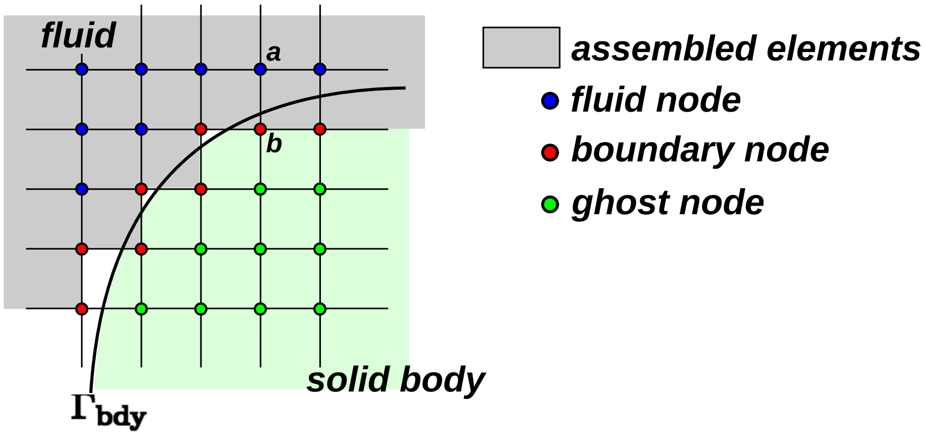

Figure 4: Description of nodes and elements used in the *AGP*

Now the preconditioning operator P will be described. First consider the whole matrix for the Laplace operator, i.e. assembling over fluid and ghost cells for F, B, and G nodes. A different symbol $\tilde{\mathbf{P}}$ will be used for this matrix since it is assembled on a different set of element/cells. The preconditioning is then defined formally as $\mathbf{y}_{FB} = \mathbf{P}\mathbf{x}_{FB}$, where $\mathbf{y}_{FB}$ is the solution of

$$\begin{bmatrix} \tilde{\mathbf{P}}_{FF} & \tilde{\mathbf{P}}_{FB} & \tilde{\mathbf{P}}_{FG} \\ \tilde{\mathbf{P}}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \tilde{\mathbf{P}}_{GF} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_G \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_G \end{bmatrix}. \tag{14}$$

However it can be seen that

- $\tilde{\mathbf{P}}_{FF} = \mathbf{A}_{FF}$ since the F nodes are those for which all elements are assembled in the Poisson problem.
- $\tilde{\mathbf{P}}_{FB} = \mathbf{A}_{FB}$, and $\tilde{\mathbf{P}}_{BF} = \mathbf{A}_{BF}$ since for instance, such a coefficient would link nodes as a and b in the figure. This coefficient comes from the assembly of all the elements that are connected to a *and* b, but since a is an F node, it means that all elements connected to a are assembled.
- $\tilde{\mathbf{P}}_{FG} = \tilde{\mathbf{P}}_{GF} = 0$ since F nodes are only connected to fluid elements and G are only connected to ghost elements, so that they can not share an element.

So

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} & \mathbf{0} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \mathbf{0} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_G \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_G \end{bmatrix}. \tag{15}$$

$\mathbf{x}_G$ can be eliminated from the bottom line, and then the following equations is obtained

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{BB}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix} \mathbf{x}_{FB} = \mathbf{y}_{FB}. \tag{16}$$

This allows to obtain an explicit expression for the preconditioning matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{BB}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix}. \tag{17}$$

A first consequence of this expression is that a lot of eigenvalues of the preconditioned matrix will be 1. Consider the space of all vectors $\mathbf{x}$ such that the boundary component B is null, i.e. the non null entries are only on F nodes, then

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{x}, \\ \mathbf{P}^{-1}\mathbf{A}\mathbf{x} &= \mathbf{x}, \end{aligned} \tag{18}$$

so that $\mathbf{x}$ is an eigenvector with eigenvalue 1.

The proposed technique is named *Accelerated Global Preconditioning (AGP)* because the solution of (14) is done on an infinite mesh with periodic boundary conditions so that it can be solved via FFT transform, which is very efficient, but the whole analysis doesn't depend on how the solution of this system is done; i.e. it may be obtained by multigrid iteration as well.

3.1 Numerical experiment computing condition numbers

The condition number of matrices for the Poisson problem have been computed with and without preconditioning (see figure 5).

- In the experiments the number of cells N_x along x ranges from 8 to 64.
- The Poisson problem is computed selecting the quadrangles whose center fall outside the body problem.
- In all cases the domain is the unit square with periodic boundary conditions.
- The bodies considered are: cylinder of radius 0.2, a vertical strip of width 0.5, and a square of side 0.5.
- The condition numbers are computed with Octave `cond()` function.

Note that in all cases the non preconditioned matrix condition number grows as $O(N_x^2)$, whereas *with the preconditioning it remains constant.*

3.2 The thin wall case

Note that the *AGP* preconditioning is based on the inclusion of the solid domain in the computation of the pressure Poisson equation. It is then clear that the weakness of such a method will be for geometries where the solid domain is, for instance, a thin layer, such that the fictitious cells added in the solid region allow the pressure gradients to be propagated through the fictitious solid domain. In other words, this preconditioning will be comparatively less efficient for geometries where the solid has a large aspect ratio. Off course the same argument is valid for *IOP*.

Consider the case where the fluid occupies the *interior* of a square

$$\Omega_{\text{fluid}} = \{(x, y) \quad / \quad \max(|x - L/2|, |y - L/2|) < L/2 - w\} \tag{19}$$

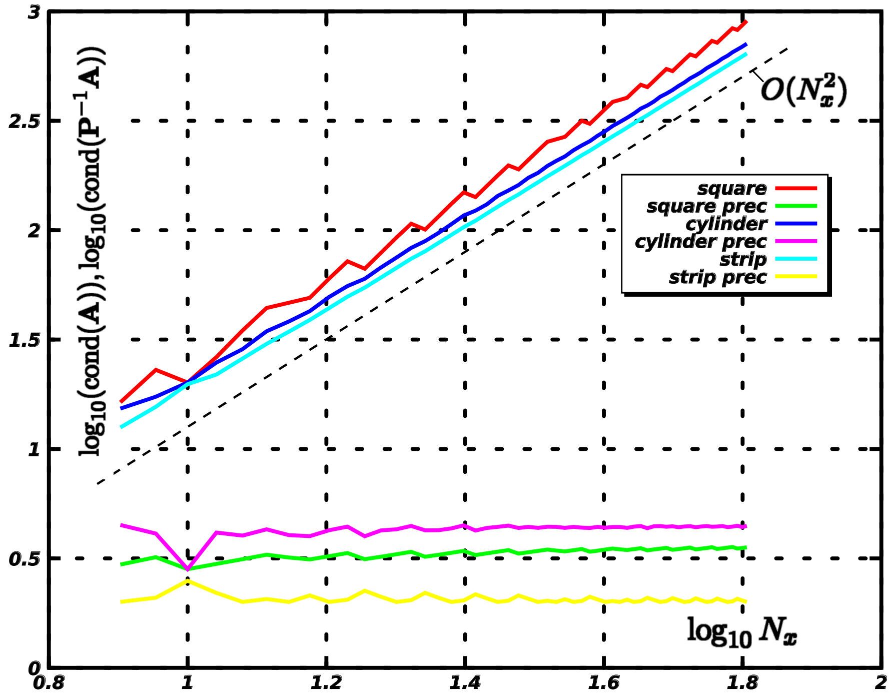

Figure 5: Condition number of Poisson problem with and without the AGP preconditioning

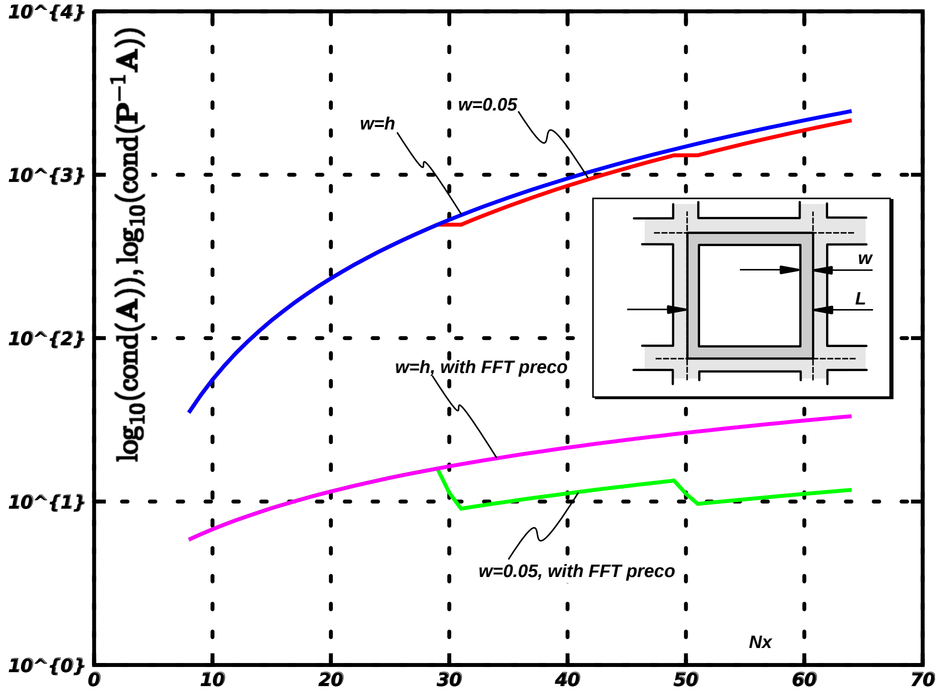

Figure 6: Condition number for Poisson problem on a square, with and without AGP preconditioning.

where w is the width of the wall (see figure 6). Due to the periodic boundary conditions this is equivalent to an infinite set of squares of side $L-2w$ filled with fluid and separated by solid walls of width $2w$. The condition number for the Poisson problem with and without preconditioning is shown in the figure.

The case where the width of the wall varies is considered so as to have always one element (in fact two elements due to the periodic b.c.'s) separating the squares. On the other hand if a fixed value $w = 0.05$ is taken then the condition number of the preconditioned case is kept bounded, jumping each time that a new element layer is included in the wall. Note that the amount of element layers in the wall is the maximum n such that $(n - 1/2)h < w$ so that the values of N_x for which a new element is added in the wall are

$$N_x = \frac{(n - 1/2)L}{w}, \tag{20}$$

where n is an integer number. For $w = 0.05$ this gives $N_x = 10, 30, 50, 70...$ and effectively it can be seen in the figure that the condition number of the preconditioned case jumps at those values of N_x.

3.3 Computation of the condition number in terms of the eigenvalues of the Steklov operators

The eigenvalues of the Steklov operators can be computed in closed form for the case of a cylinder in an infinite flow. Recall that the convergence of the *AGP* scheme is controlled by the condition number of the preconditioned operator

$$\mathrm{cond}(P^{-1}A) = \frac{\max|\,\mathrm{eig}(P^{-1}A)|}{\min|\,\mathrm{eig}(P^{-1}A)|} \tag{21}$$

As all are positive and definite operators, the eigenvalues λ are real and positive, and $0 \leq \lambda \leq 1$. Also there are a lot of eigenvalues that are unity $\lambda = 1$, they correspond to the space of functions that satisfy the boundary condition $(\partial\phi/\partial n) = 0$ at Γ. However, the Krylov methods iterate only on the space perpendicular to it, and it can be shown the condition number of the preconditioned Steklov operator

$$\tilde{\mathcal{S}} = (\mathcal{S}_F + \mathcal{S}_S)^{-1}\mathcal{S}_F. \tag{22}$$

has to be computed.

$$\kappa(\tilde{\mathcal{S}}) = \frac{\max[\mathrm{eig}(\tilde{\mathcal{S}})]}{\min[\mathrm{eig}(\tilde{\mathcal{S}})]} \tag{23}$$

For simple geometries like a semi-infinite plane, a strip and a cylinder the eigenvalues of $\mathcal{S}_F$, $\mathcal{S}_S$ can be computed. In addition, it turns out that the eigenfunctions are the same, so that the spectral decomposition of the sum $\mathcal{S}_F + \mathcal{S}_S$ and the preconditioned operator are available.

Recall if $V_\Gamma = \{\text{real valued functions on } \Gamma\}$ then the $\mathcal{S}_F : V_\Gamma \to V_\Gamma$ operator is defined as follows. $w = \mathcal{S}_F(v)$ if $w = (\partial\phi/\partial n)|_\Gamma$, where ϕ satisfies

$$\begin{cases} \Delta\phi = 0, \text{ in } \Omega_F \\ \phi_\Gamma = v, \end{cases} \tag{24}$$

where $\hat{\mathbf{n}}$ is the normal to Γ exterior to Ω_F. The same definition, *mutatis mutandis*, applies to $\mathcal{S}_S$.

3.3.1 The semiplane

The geometry consists on a semiplane

$$\begin{aligned} \Omega_F &= \{\mathbf{x} = (x, y) \quad / \quad x < 0\} \\ \Omega_S &= \{\mathbf{x} = (x, y) \quad / \quad x > 0\} \\ \Gamma &= \{\mathbf{x} = (x, y) \quad / \quad x = 0\} \end{aligned} \tag{25}$$

By symmetry of translation in the y direction the eigenfunctions must be plane waves of the form $v = \mathrm{e}^{iky}$ with k real. The solution to the Laplace problem on the fluid and solid are

$$\begin{aligned} \phi &= \mathrm{e}^{iky}\mathrm{e}^{-|k|x}, \ \mathbf{x} \in \Omega_S, \\ \phi &= \mathrm{e}^{iky}\mathrm{e}^{|k|x}, \ \mathbf{x} \in \Omega_F, \end{aligned} \tag{26}$$

and then

$$\mathrm{eig}\{\mathcal{S}_F(v)\} = \mathrm{eig}\{\mathcal{S}_S(v)\} = |k|, \tag{27}$$

so that all the eigenvalues of $\tilde{\mathcal{S}}$ are $\lambda_n = 1/2$, and $\kappa(\tilde{\mathcal{S}}) = 1$.

3.3.2 The cylinder

In this case

$$\begin{aligned} \Omega_S &= \{||\mathbf{x}|| < R\}, \\ \Omega_F &= \{||\mathbf{x}|| > R\}, \\ \Gamma &= \{||\mathbf{x}|| = R\}, \end{aligned} \tag{28}$$

By rotational symmetry the eigenfunctions are

$$\begin{aligned} v_{n,+} &= \cos(n\theta), \quad n \geq 0 \\ v_{n,-} &= \sin(n\theta), \quad n > 1 \end{aligned} \tag{29}$$

where r, θ are polar coordinates at the center of the cylinder. The solution at both domains are

$$\begin{aligned} \phi_{n,\pm,F} &= r^{-n}\begin{Bmatrix}\cos(n\theta) \\ \sin(n\theta)\end{Bmatrix}, \\ \phi_{n,\pm,S} &= r^{n}\begin{Bmatrix}\cos(n\theta) \\ \sin(n\theta)\end{Bmatrix}, \end{aligned} \tag{30}$$

so that the eigenvalues and eigenfunctions of both operators are the same

$$\lambda(n, \pm, F/S) = \frac{n}{R} \tag{31}$$

and again $\lambda_{n,\tilde{\mathcal{S}}} = 1/2$, $\kappa(\tilde{\mathcal{S}}) = 1$.

Comparison with numerical values. There seems to be a discrepancy between the values of for the cylinder as computed numerically and reported in figure 5 and the theoretical prediction given before. The difference is due to the fact that in the discrete case with the technique of embedding of the solid in a homogeneous constant step mesh (see the gray mesh in figure 4), the staircase boundary of the mesh generates spurious eigenvalues that cause the smaller eigenvalues (smoother eigenfunctions) not to converge to the theoretically predicted spectrum. However, if a FEM boundary fitted mesh is used (see figures 7) then the theoretical result of $\lambda_{n,F} = \lambda_{n,S} = n/R$ is much better satisfied (see Table 1). Note that the eigenvalues for the solid domain $\lambda_{n,S}$ are much more closer to the theoretical prediction. This is caused by the fact that for the fluid there is also the problem of interference with the external boundary condition, i.e. the fluid domain is enclosed in an artificial boundary of side $L = 1$, and this introduces an error, which will disappear as $R/L \to 0$.

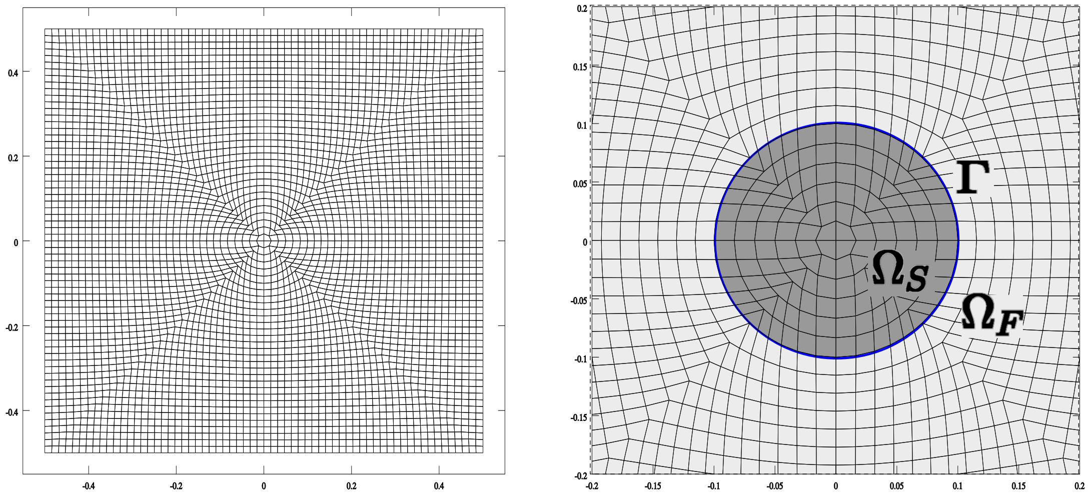

Figure 7: Boundary fitted FEM mesh of 64x64 elements for the cylinder. Right: global view of the mesh. Left: detail at the solid domain.

Eigenvalue index n	$\lambda_{n,F}$	$\lambda_{n,S}$	$\lambda_{n,\tilde{S}}$ (num.)	$\lambda_{n,S/F}$ (analytic)
1	9.38	10.00	0.4840	10
2	20.19	20.01	0.5022	20
3	30.88	30.46	0.5034	30
4	41.74	40.98	0.5046	40
5	53.72	52.49	0.5058	50

Table 1: Comparison of theoretical and numerical values for the $\mathcal{S}_{S/F}$ operators for the cylinder with diameter $R = 0.1$ and a boundary fitted FEM mesh of 64x64 elements. Note that the values for $\lambda_{n,\tilde{S}}$ are much closer to 0.5 than for the staircase geometry.

3.3.3 The infinite strip

The solid domain is $\Omega_S = \{|x| \leq w/2\}$ where w is the width of the strip. The space V_Γ are pairs of functions on both sides of the strips. By translation invariance in the y direction the eigenfunctions must satisfy have

$$v = \begin{cases} a\,\mathrm{e}^{iky}, & \text{for } x = -w/2 \\ b\,\mathrm{e}^{iky}, & \text{for } x = +w/2 \end{cases} \tag{32}$$

it can be shown by symmetry $x \to -x$ that the eigenfunctions are the symmetric ($a = b$) and antisymmetric ($a = -b$) combinations, so that

$$v_{k,\pm} = \begin{cases} \pm\mathrm{e}^{iky}, & \text{for } x = -w/2, \\ \mathrm{e}^{iky}, & \text{for } x = +w/2. \end{cases} \tag{33}$$

The corresponding solution for the symmetric modes at $\Omega_{F,S}$ are

$$\phi_{k,+} = \begin{cases} \mathrm{e}^{iky+|k|(w/2-x)}, & \text{for } |x| \geq w/2, \\ \dfrac{\cosh(kx)}{\cosh(kw/2)}\mathrm{e}^{iky}, & \text{for } |x| \leq w/2, \end{cases} \tag{34}$$

and the corresponding eigenvalues

$$\begin{aligned} \lambda(k,+,F) &= |k|, \\ \lambda(k,+,S) &= k\tanh(kw/2). \end{aligned} \tag{35}$$

And for the antisymmetric eigenfunctions,

$$\phi_{k,-} = \begin{cases} \mathrm{sign}(x)\mathrm{e}^{iky+|k|(w/2-x)}, & \text{for } |x| \geq w/2, \\ \dfrac{\sinh(kx)}{\sinh(kw/2)}\mathrm{e}^{iky}, & \text{for } |x| \leq w/2, \end{cases} \tag{36}$$

and the corresponding eigenvalues

$$\begin{aligned} \lambda(k,-,F) &= |k|, \\ \lambda(k,-,S) &= k\coth(kw/2). \end{aligned} \tag{37}$$

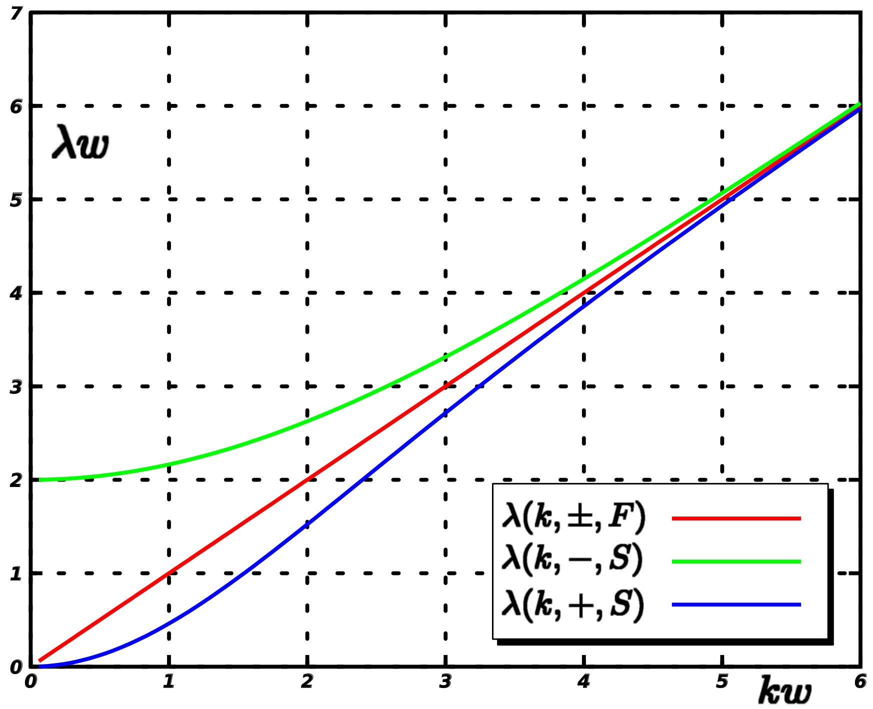

Figure 8: Eigenvalues of Steklov operators for the solid strip case.

Both the symmetric and antisymmetric eigenvalues van be seen in figure 8. Note that the eigenvalues of the fluid operator $\lambda(k,\pm)$ are the same for the symmetric and antisymmetric case and independent of the strip width w, since the fluid domain is completely decoupled by the strip, and then the eigenvalues of the Steklov operator $\mathcal{S}_F$ are the same as those of each semiplane $x > w/2$ and $x < w/2$.

On the other hand, with respect to the eigenvalues of the Steklov operator of the strip (solid domain) $\mathcal{S}_S$, both symmetric and antisymmetric eigenvalues behave like $\sim |k|$ for $kw \to \infty$. This is because for kw large means that the wavelength of the eigenfunction is much smaller than the width of the strip and then again, the behavior of the eigenvalues is the same as for a infinite semiplane.

However, when kw is small the behavior of the symmetric and antisymmetric branches is very different. Remember that, as per definition of the Steklov operator, given an eigenfunction u it can be imposed as a Dirichlet boundary condition on the interface Γ, solve the Laplace equation for the field ϕ in the corresponding domain and compute the flux $v = (\partial\phi/\partial n)$. As u is an eigenfunction, v should be proportional to u and the proportionality constant is the

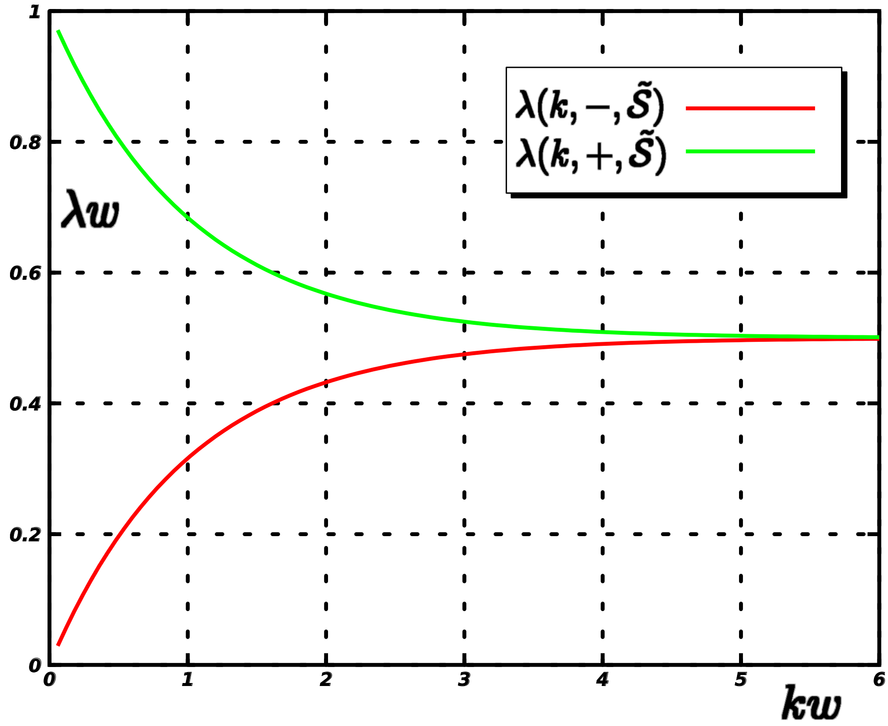

Figure 9: Eigenvalues of preconditioned Steklov operator for the symmetric and antisymmetric branches.

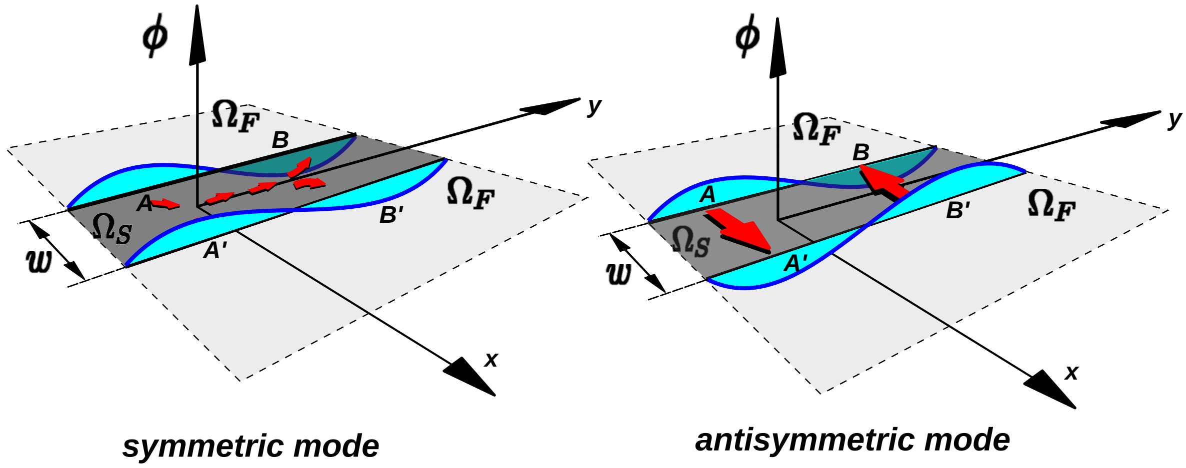

Figure 10: Explanation of the behavior of the Steklov eigenvalues for large wavelengths.

eigenvalue λ. First consider the symmetric branch. If a sinusoidal value on the left boundary $x = -w/2$ is imposed (see figure 10) then for the symmetric mode there are exactly the same Dirichlet boundary condition on the other boundary $x = +w/2$. As a result facing points inside the strip like A, A' or B, B' have equal values of temperature ϕ imposed and then the heat flow is very low, which means a small eigenvalue. On the other hand, for the antisymmetric mode, opposing points have the same absolute temperature but of opposed sign, and the heat flow is very high (the red arrows in the figure). This explains why for low wavenumber k the eigenvalues of the symmetric mode are smaller, with a behavior $\lambda \propto k^2$ for $k \to 0$. For the antisymmetric mode for low wavenumber the eigenvalue is larger with a behavior $\lambda w \to 2$ for $kw \to 0$.

This last limit is simple to understand. Effectively, for very small wavenumber the conduction in the y direction can be neglected, and so for an eigenfunction $u = \pm\cos ky$ at $x = \pm w/2$ the solution is

$$\phi = \frac{2x}{w}\cos ky, \tag{38}$$

so that

$$v = (\partial\phi/\partial n)|_{x=w/2} = \frac{2}{w}\cos ky = \frac{2}{w}u. \tag{39}$$

So that, $\lambda w = 2$.

The eigenvalues of the preconditioned Steklov operators are plot in figure 9. They are given by the expressions

$$\begin{aligned}\lambda(k,+,\tilde{\mathcal{S}}) &= \frac{|k|}{|k| + k\tanh(kw/2)},\\ \lambda(k,-,\tilde{\mathcal{S}}) &= \frac{|k|}{|k| + k\coth(kw/2)}.\end{aligned} \tag{40}$$

Note that for both symmetric and antisymmetric mode the eigenvalues tend to $1/2$ for $kw \to \infty$, as for an infinite semiplane. On the other hand, for small kw the eigenvalues of the symmetric mode are larger than $1/2$, and those for the antisymmetric modes are smaller. So, if the eigenvalues for the symmetric modes are considered the condition of the preconditioned Steklov operator is 2, whereas if the antisymmetric modes are considered the condition number tends to ∞ since the smallest eigenvalue tends to 0 for $kw \to 0$.

This is expected, since the *AGP* preconditioning is based on solving the Laplace equation on the global domain, fluid and solid, instead in solving only on the fluid. Thus, this preconditioning is good whenever the fluxes in the solid domain are small. But this is exactly the case for the symmetric modes, and the opposite happens for the antisymmetric modes.

3.3.4 Estimation of the condition number for thin walls

The analysis of the infinite strip shows that a situation where the *AGP* preconditioning is deteriorated is when there are thin walls in the solid geometry, since in that case the modes that are antisymmetric about the axis of the solid produce large heat fluxes in the solid, and this is an indication of bad performance of the preconditioning. So for elongated solid geometries with, say, a typical length of L and a typical width of $w \ll L$, an estimate of the condition number of the preconditioned operator can be obtained by taking L as the maximal wavelength and the estimate gives a condition number of

$$\kappa(\tilde{\mathcal{S}}) \sim \frac{|k_{\min}| + k_{\min}\coth(k_{\min}w/2)}{|k_{\min}|} \tag{41}$$

where $k_{\min} = 2\pi/L$. By algebraic manipulation this can be simplified to

$$\kappa(\tilde{\mathcal{S}}) \sim 1 + \coth(\frac{\pi w}{L}) \sim \frac{L}{\pi w}, \tag{42}$$

i.e. the condition number is *proportional to the aspect ratio* of the solid domain.

3.4 Compute times for AGP

The *AGP* method has been implemented with *CUDA* and several experiments have been run on a Nvidia Tesla C1060 in double precision. Meshes running from 64^3 to 256^3 have been tested and the computing times per iteration and the number of iterations needed to reach a residual convergence of 10^{-6} are reported on Table 2. In the table it is also reported the time needed to compute two FFT's. It is clear that in the limit the FFT's is the most CPU time consuming stage of the whole algorithm. As a reference the time for FFT's was measured versus the number of cells and is shown in Figure 11. The plot shows the computing rate in Gflops according to a standard estimate of the operation count

$$r[\text{Gflops}] = \frac{5N\log_2(N)}{\text{elapsed time[s]}} \tag{43}$$

where N is the total number of cells. The performance of the Tesla C1060 is in the range of 25 to 30 Gflops in the range of interest. It has been reported that the new Tesla C2050 (Fermi) GPU has a performance of 80 Gflops for the FFT.

Mesh size	iters	Time/iter. [s]	Rate [Mcells/iter/s]	FFT time [s] (%)
$64 \times 64 \times 64$	7	0.00340	76.91	0.00174 (51.2%)
$64 \times 128 \times 128$	5	0.0107	97.97	0.00776 (72.5%)
$128 \times 128 \times 128$	7	0.0184	113.44	0.0163 (88.2%)
$256 \times 256 \times 256$	6	0.184	90.96	0.149 (80.8%)

Table 2: Computing times for the Poisson stage on a Nvidia Tesla C1060. The global (fluid+solid) problem is solved with FFT. The number of iterations to attain a relative residual of 10^{-6} is also shown. The last columns is time spent in performing the two FFT's.

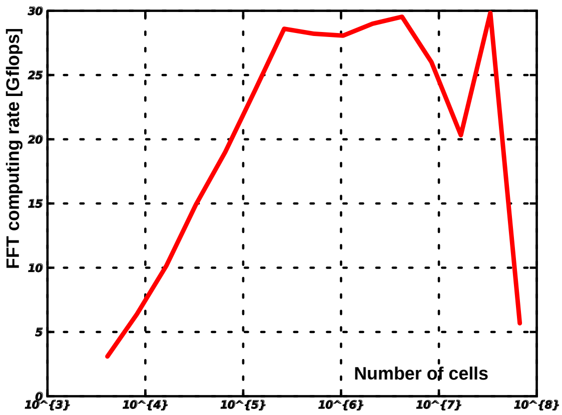

Figure 11: Performance of the CUDA FFT library in Gflops according to Eq. (43)

4 CONVERGENCE RATE OF IOP

As the basis for *IOP* is the same as for *AGP* it is not surprising that its weakness is also for solid domains with large aspect ratio. In fact the convergence for both methods depend on the eigenvalue spectrum of the same operator. However recall that *AGP* is an *accelerated* method (i.e. a Krylov space method) whereas *IOP* is an stationary one, and so the convergence of the first is much faster. The rate of convergence for *IOP* will be discussed in this section.

It will be shown below that the amplification matrix for the *IOP* iteration scheme is

$$\mathbf{Q} = (\mathbf{A}_F + \mathbf{A}_S)^{-1}\mathbf{A}_S. \tag{44}$$

The eigenvalues of $\mathbf{Q}$ are all $0 \leq \lambda < 1$, for convergence, and the rate of convergence is $\max(\mathrm{eig}(\mathbf{Q}))$. Note that $\mathbf{Q} = \mathbf{I} - \tilde{\mathcal{S}}$, and then an eigenvalue close to one for $\mathbf{Q}$ means a slow rate of convergence for *IOP*. But an eigenvalue close to one for $\mathbf{Q}$ implies an eigenvalue close to zero for $\tilde{\mathcal{S}}$, and then a bad condition number, and poor convergence for *AGP*.

In order to show (44), the iteration matrix will be computed explicitly for the 1D case. Recall that the *IOP* iteration for solid bodies at rest is a fixed point iteration scheme that can be written as

$$\mathbf{u}' = \mathbf{\Pi}_{\mathrm{div}} \mathbf{\Pi}_F \mathbf{u}, \tag{45}$$

where $\mathbf{\Pi}_F$ is the projection matrix that makes a velocity field to satisfy the boundary condition, i.e. it makes it null at the nodes in the solid. In the case of bodies at rest it is an homogeneous projection operator that is simply a diagonal matrix with 0's or 1's depending on whether the velocity node is in the fluid or solid domains (Ω_F and Ω_S). For moving bodies the projection is affine, i.e. not homogeneous, because the velocity of the body is non null, but the treatment given here can be extended to that case also. On the other hand $\mathbf{\Pi}_{\mathrm{div}}$ is defined as $\mathbf{u}' = \mathbf{\Pi}_{\mathrm{div}}\mathbf{u}$ where $\mathbf{u}'$ is computed by solving the following problem

$$\begin{aligned} \mathbf{A}\mathbf{p} &= \mathbf{G}^T\mathbf{u}, \\ \mathbf{u}' &= \mathbf{u} - \mathbf{G}\mathbf{P}. \end{aligned} \tag{46}$$

By eliminating $\mathbf{p}$, a compact form for the projection operator is obtained

$$\mathbf{\Pi}_{\mathrm{div}} = \mathbf{I} - \mathbf{G}\mathbf{A}^{-1}\mathbf{G}^T. \tag{47}$$

Now, the iteration scheme can be written as

$$\mathbf{u}' = \mathbf{u} - \mathbf{G}\mathbf{A}^{-1}\mathbf{r}, \tag{48}$$

where

$$\mathbf{r} = \mathbf{G}\mathbf{\Pi}_F\mathbf{u}. \tag{49}$$

And it is clear that the scheme converges as $\mathbf{r} \to 0$. Premultiplying (45) by $\mathbf{G}\mathbf{\Pi}_F$

$$\begin{aligned} \mathbf{G}^T\mathbf{\Pi}_F\mathbf{u}' &= \mathbf{G}^T\mathbf{\Pi}_F\mathbf{\Pi}_F\mathbf{u} - \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{A}^{-1}\mathbf{G}^T\mathbf{\Pi}_F\mathbf{u}, \\ \mathbf{r}' &= \mathbf{r} - \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{A}^{-1}\mathbf{r}, \\ \mathbf{r}' &= (\mathbf{I} - \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{A}^{-1})\mathbf{r}. \end{aligned} \tag{50}$$

So the rate of convergence of the scheme is governed by the largest eigenvalue of the step amplification matrix

$$\mathbf{Q} = \mathbf{I} - \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{A}^{-1}. \tag{51}$$

Now, note that in the continuum $\mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}$ is the Laplace operator acting only in the fluid domain. It can be shown that in the case of the staggered FVM scheme used in this work the following holds true

$$\mathbf{G}^T\mathbf{\Pi}_F\mathbf{G} = \mathbf{A}_F, \tag{52}$$

where $\mathbf{A}_F$ is the Laplace operator assembled only on those edges whose center fall in the domain region. In terms of equation (14)

$$\mathbf{A}_F = \begin{bmatrix} \tilde{\mathbf{P}}_{FF} & \tilde{\mathbf{P}}_{FB} & \mathbf{0} \\ \tilde{\mathbf{P}}_{BF} & \tilde{\mathbf{P}}_{BB} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{53}$$

For instance consider the 1D case. The domain $[0, L]$ is discretized with N cells of constant mesh step $h = 1/N$ and with periodic boundary conditions. The pressure nodes are located at $x_{pj} = (j + 1/2)h$ and the velocity nodes at $x_{uj} = jh$, for $j \in [0, N)$. Then the gradient operator is

$$(\nabla p)_j = \frac{p_j - p_{j-1}}{h}, \quad \text{(periodic)} \tag{54}$$

and the corresponding matrix is

$$\mathbf{G} = \frac{1}{h} \begin{bmatrix} 1 & 0 & 0 & \dots & -1 \\ -1 & 1 & 0 & \dots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{55}$$

With respect to the projection operator, assume that the solid domain occupies the central N_S *velocity* cells, preceded and followed by N_{F1} and N_{F2} fluid cells (see figure 12). The projection

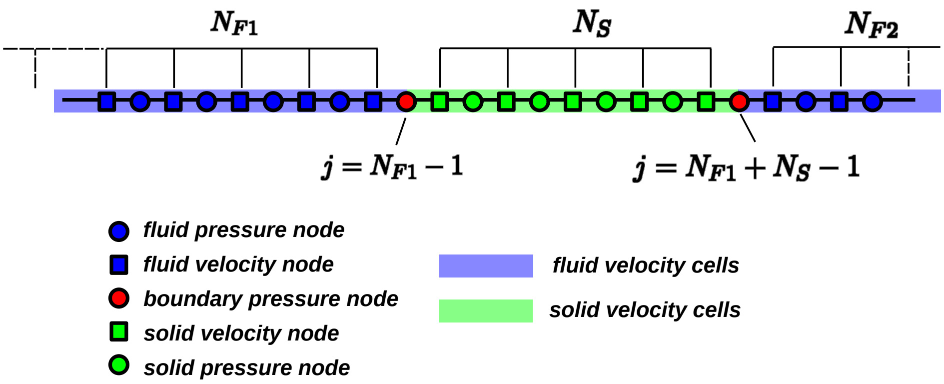

Figure 12: Staggered FVM 1D mesh.

operator is then

$$\mathbf{\Pi}_F = \operatorname{diag}\{\overbrace{1, 1, \dots, 1}^{N_{F1}}, \overbrace{0, \dots, 0}^{N_S}, \overbrace{1 \dots, 1}^{N_{F2}}\}. \tag{56}$$

and then it can be verified by direct computation that (52) is valid, if the fluid, solid, and boundary nodes are considered as shown in figure 12. Then

$$\mathbf{A}_F\mathbf{p} = \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{p} = \\ \begin{cases} \dfrac{-p_{j+1} + 2p_j - p_{j-1}}{h^2}, & \text{for } j < N_{F1} - 1 \text{ and } j > N_{F1} + N_S - 1, \text{(fluid nodes, periodic)}, \\ \dfrac{p_j - p_{j-1}}{h}, & \text{for } j = N_{F1} - 1, \text{ (boundary node)}, \\ \dfrac{p_j - p_{j+1}}{h}, & \text{for } j = N_{F1} + N_s - 1, \text{ (boundary node)}, \\ 0, & \text{for } N_{F1} - 1 < j < N_{F1} + N_s - 1, \text{ (solid nodes)}. \end{cases} \tag{57}$$

Then replacing (52) in (51)

$$\begin{aligned} \mathbf{Q} &= \mathbf{I} - \mathbf{G}^T\mathbf{\Pi}_F\mathbf{G}\mathbf{A}^{-1} \\ &= \mathbf{I} - \mathbf{A}_F\mathbf{A}^{-1} \\ &= \mathbf{A}_S\mathbf{A}^{-1} \end{aligned} \tag{58}$$

Again, it can be shown that in fact the rate of convergence is governed by iteration on the boundary values, so that the rate of convergence is given by the

$$\begin{aligned} \text{conv. rate}(\text{IOP}) &= \max[\text{eig}(\mathbf{Q})], \\ \mathbf{Q} &= \mathbf{I} - \mathbf{G}^T \mathbf{\Pi}_F \mathbf{G} \mathbf{A}^{-1}, \\ &= \mathbf{I} - \mathbf{A}_F \mathbf{A}^{-1}, \\ &= \mathbf{A}_S \mathbf{A}^{-1}. \end{aligned} \tag{59}$$

Convergence of *IOP* is controlled by the maximum eigenvalue from $\text{eig}(\mathbf{A}_S\mathbf{A}^{-1})$, which is by a similarity transformation the same as $\text{eig}(\mathbf{A}^{-1}\mathbf{A}_S) = 1 - \text{eig}(\mathbf{A}^{-1}\mathbf{A}_F)$. As $\mathbf{A}, \mathbf{A}_F, \mathbf{A}_S$ are symmetric and positive definite matrices it results that the eigenvalues of $\text{eig}(\mathbf{A}^{-1}\mathbf{A}_S)$ and $\text{eig}(\mathbf{A}^{-1}\mathbf{A}_F)$ are in the range $0 \le \lambda \le 1$.

4.1 Reduction of the global domain problem to the boundary problem

First, let us note that all vectors that have only a component in the fluid part have eigenvalue one

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \mathbf{x}_F \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \\ \mathbf{A}_F \mathbf{x} &= \begin{bmatrix} \mathbf{A}_{FF}\mathbf{x}_F \\ \mathbf{A}_{BF}\mathbf{x}_F \\ \mathbf{0} \end{bmatrix} = \mathbf{A}\mathbf{x}, \end{aligned} \tag{60}$$

whereas those that have only a component in the solid have eigenvalue 0

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{x}_S \end{bmatrix}, \\ \mathbf{A}_F \mathbf{x} &= \mathbf{0}, \\ \mathbf{A}^{-1}\mathbf{A}_F \mathbf{x} &= \mathbf{0}. \end{aligned} \tag{61}$$

The rate of convergence is governed then by the maximum eigenvalue that are in the range $0 < \lambda < 1$, i.e. excluding these previous ones. As the matrices are symmetric, the eigenvectors must be orthogonal with respect to $\mathbf{A}$ so that the remaining eigenvalues are obtained by imposing that they must be orthogonal for both previous families or eigenvectors,

$$\begin{aligned} &\begin{bmatrix} \mathbf{y}_F^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{A}\mathbf{x}, \ \forall \mathbf{y}^F, \\ &\implies \mathbf{A}_{FF}\mathbf{x}_F + \mathbf{A}_{FB}\mathbf{x}_B = 0, \\ &\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{y}_S \end{bmatrix} \mathbf{A}\mathbf{x}, \ \forall \mathbf{y}^S, \\ &\implies \mathbf{A}_{SB}\mathbf{x}_B + \mathbf{A}_{SS}\mathbf{x}_S = 0, \end{aligned} \tag{62}$$

from where

$$\begin{aligned} \mathbf{x}_F &= -\mathbf{A}_{FF}^{-1}\mathbf{x}_B, \\ \mathbf{x}_S &= -\mathbf{A}_{SS}^{-1}\mathbf{x}_B. \end{aligned} \tag{63}$$

Replacing these expressions in the eigenvalue equation

$$\mathbf{A}_F \mathbf{x} = \lambda \mathbf{A}\mathbf{x}, \tag{64}$$

and performing the block decomposition

$$
\begin{aligned}
\mathbf{A}_F &= \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} & \mathbf{0} \\ \mathbf{A}_{BF} & \mathbf{A}_{BB,F} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\mathbf{A}_S &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{BB,S} & \mathbf{A}_{BS} \\ \mathbf{0} & \mathbf{A}_{SB} & \mathbf{A}_{SS} \end{bmatrix}, \\
\mathbf{A} &= \mathbf{A}_F + \mathbf{A}_S,
\end{aligned}
\tag{65}
$$

results in that the first and last rows of equations are satisfied automatically due to (62) and the middle row results in

$$
\mathbf{A}_{BF}\mathbf{x}_F + \mathbf{A}_{BB,F}\mathbf{x}_B = \lambda(\mathbf{A}_{BF}\mathbf{x}_F + \mathbf{A}_{BB}\mathbf{x}_B + \mathbf{A}_{SS}\mathbf{x}_S), \tag{66}
$$

i.e.

$$
\mathbf{S}_F\mathbf{x}_B = \lambda\mathbf{S}\mathbf{x}_B, \tag{67}
$$

where

$$
\begin{aligned}
\mathbf{S}_F &= \mathbf{A}_{BB,F} - \mathbf{A}_{BF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FB}, \\
\mathbf{S}_S &= \mathbf{A}_{BB,S} - \mathbf{A}_{BS}\mathbf{A}_{SS}^{-1}\mathbf{A}_{SB}, \\
\mathbf{S}_S &= \mathbf{S}_F + \mathbf{S}_S.
\end{aligned}
\tag{68}
$$

So that the convergence of *IOP* is governed by the spectrum of the operator $\mathbf{S}^{-1}\mathbf{S}_F$. Note that this is the discrete version of the continuum operator in (22) that controls the convergence of the *AGP* method.

4.2 Variation of IOP based on pressure

A drawback of the *IOP* method is that it iterates over both the velocity and pressure vectors. But the method can be slightly modified so as to only iterate on pressure, keeping the same rate of convergence. However, the result can be proved to be equivalent to the standard *IOP* algorithm in the continuum but in the discrete case it is the same only under certain conditions.

The pressure based variant of *IOP* (*PB-IOP* in what follows) is based on solving the Poisson problem

$$
\mathbf{A}_F\mathbf{p} = \mathbf{b}, \tag{69}
$$

by the following stationary method

$$
\mathbf{p}^{n+1} = \mathbf{p}^n - \mathbf{P}^{-1}(\mathbf{A}\mathbf{p}^n - \mathbf{b}). \tag{70}
$$

The amplification operator is the same as for the standard *IOP* $\mathbf{A}^{-1}\mathbf{A}_F$ and so the convergence rate is the same. However both convergence to the same solution if

$$
\mathbf{G}^T\mathbf{\Pi}_F\mathbf{G} = \mathbf{A}_F. \tag{71}
$$

which is indeed true for the FVM approximation used in this work.

5 CONCLUSIONS

The *Accelerated Global Preconditioning (AGP)* algorithm for the solution of the Poisson equation specially oriented to the solution of Navier-Stokes equations on GPU hardware was presented. It shares some features with the well known *IOP* iteration scheme. As a summary of the comparison between both methods, the following issues may be mentioned

- Both solvers are based on the fact that an efficient preconditioning that consists in solving the Poisson problem on the global domain (fluid+solid). Of course, this represents more computational work than solving the problem only in the fluid, but this can be faster in a structured mesh with some fast solvers as Multigrid or *FFT*.
- Both solvers have their convergence governed by the spectrum of the $\mathbf{S}^{-1}\mathbf{S}_F$, however
  - *IOP* is a *stationary method* and its limit rate of convergence is given by

$$\begin{aligned} &\|\mathbf{r}^{n+1}\| \leq \gamma_{\rm IOP}\|\mathbf{r}^n\| \\ &\gamma_{\rm IOP} = 1 - \lambda_{\rm min}, \\ &\lambda_{\rm min} = \min(\mathrm{eig}(\mathbf{S}^{-1}\mathbf{S}_F)). \end{aligned} \tag{72}$$

    In particular if there is no solid, then $\mathbf{S}_F = \mathbf{S}$, $\mathbf{S}^{-1}\mathbf{S}_F = \mathbf{I}$, $\lambda_{\rm min} = 1$, and $\gamma_{\rm IOP} = 0$, so that the scheme converges in one iteration.
  - *AGP* is a preconditioned *Krylov space method* and its convergence is governed by the condition number of $\mathbf{S}^{-1}\mathbf{S}_F$, i.e.

$$\begin{aligned} \kappa(\mathbf{A}^{-1}\mathbf{A}_F) &= \frac{\max(\mathrm{eig}(\mathbf{A}^{-1}\mathbf{A}_F))}{\min(\mathrm{eig}(\mathbf{A}^{-1}\mathbf{A}_F))}, \\ &= \frac{1}{\min(\mathrm{eig}(\mathbf{A}^{-1}\mathbf{A}_F))}, \\ &= \frac{1}{\min(\mathrm{eig}(\mathbf{S}^{-1}\mathbf{S}_F))}, \\ &= \frac{1}{\lambda_{\rm min}}, \end{aligned} \tag{73}$$

  Again, when there is no fluid $\mathbf{S}^{-1}\mathbf{S}_F = \mathbf{I}$, $\lambda_{\rm min} = 1$, and $\kappa(\mathbf{A}^{-1}\mathbf{A}_F) = 1$, so that *AGP* converges also in one iteration.
- It has been shown that $\lambda_{\rm min} = O(1)$, i.e. it *does not degrade with refinement*, so that *IOP* has a linear convergence with limit rate $O(1)$.
- By the same reason, the condition number for *AGP does not degrade with refinement*.
- *IOP* iterates over both the velocity and pressure fields, whereas *AGP* iterates only on the pressure vector (which is better for implementation on GPU's).

6 ACKNOWLEDGMENTS

This work has received financial support from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, PIP 5271/05), Universidad Nacional del Litoral (UNL, Argentina, grant CAI+D 2009-65/334) y Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1506/2006, PICT-1141/2007, PICT-0270/2008).

The authors made extensive use of *Free Software* (http://www.gnu.org) as GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software as Vtk among many others. In addition, many ideas from these packages have been inspiring to them.

REFERENCES

Adams S., Payne J., and Boppana R. Finite difference time domain (FDTD) simulations using graphics processors. *HPCMP Users Group Conference*, 0:334–338, 2007. doi:10.1109/HPCMP-UGC.2007.34.

Bell N. and Garland M. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–11. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-744-8. doi:10.1145/1654059.1654078.

Corrigan A., Camelli F.F., Löhner R., and Wallin J. Running unstructured grid-based CFD solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 2010. (in press).

Crane K., Llamas I., and Tariq S. Chapter 30 - Real-Time Simulation and Rendering of 3D Fluids. 2008. doi:10.1.1.88.9775.

Elcott S., Tong Y., Kanso E., Schröder P., and Desbrun M. Stable, circulation-preserving, simplicial fluids. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–11. ACM, New York, NY, USA, 2008. doi:10.1145/1508044.1508061.

Elsen E., LeGresley P., and Darve E. Large calculation of the flow over a hypersonic vehicle using a GPU. *J. Comput. Phys.*, 227(24):10148–10161, 2008. ISSN 0021-9991. doi:10.1016/j.jcp.2008.08.023.

Goddeke D., Strzodka R., Mohd-Yusof J., McCormick P., Wobker H., Becker C., and Turek S. Using GPU's to improve multigrid solver performance on a cluster. *Int. J. Comput. Sci. Eng.*, 4(1):36–55, 2008. ISSN 1742-7185. doi:10.1504/IJCSE.2008.021111.

Irving G., Guendelman E., Losasso F., and Fedkiw R. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph.*, 25(3):805–811, 2006. ISSN 0730-0301. doi:10.1145/1141911.1141959.

Lastra M., Mantas J.M., Ure na C., Castro M.J., and García-Rodríguez J.A. Simulation of shallow-water systems using graphics processing units. *Math. Comput. Simul.*, 80(3):598–618, 2009. ISSN 0378-4754. doi:10.1016/j.matcom.2009.09.012.

Leonard B. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comput. Methods Appl. Mech. Eng.*, 19(1):59–98, 1979.

Molemaker J., Cohen J.M., Patel S., and Noh J. Low viscosity flow simulations for animation. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 9–18. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2008. ISBN 978-3-905674-10-1.

Mullen P., Crane K., Pavlov D., Tong Y., and Desbrun M. Energy-preserving integrators for fluid animation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-726-4. doi:10.1145/1576246.1531344.

P.Rinaldi, Bauza C.G., Vénere M., and Clausse A. Paralelización de autómatas celulares de aguas superficiales sobre placas gráficas. In A. Cardona, M. Storti, and C. Zuppa, editors, *Mecánica Computacional Vol. XXVII*, volume XXVII, pages 2943–2957. 2008.

Ryoo S., Rodrigues C.I., Baghsorkhi S.S., Stone S.S., Kirk D.B., and Hwu W.m.W. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82. ACM, New York, NY, USA, 2008. ISBN 978-1-59593-795-7. doi:10.1145/1345206.1345220.

Thibault J.C. and Senocak I. CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. In AIAA, editor, *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition (Disc 1)*. 2009.

Wu E., Liu Y., and Liu X. An improved study of real-time fluid simulation on GPU: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):139–146, 2004. ISSN 1546-4261. doi:10.1002/cav.v15:3/4.