A Numerical Algorithm for the Solution of Viscous Incompressible Flows on GPUs

Santiago D. Costarelli[1] Mario A. Storti[1,2]
Rodrigo R. Paz[1,2] Lisandro D. Dalcin[1,2]

[1] CONICET-INTEC-CIMEC

[2] Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional de Litoral

HPCLatam 2012
Buenos Aires, Argentina.

Motivation

Due to the difficulty on solving exact N-S equations, numerical schemes have been adopted to solve complex fluid flows patterns.
The most popular being F*M (for FDM, FVM or FEM).
In particular, FDM/FVM method on structured grids are very suitable to solve on GPGPU's due to the local requirement of data to solve the fields (Cellular automata[1]).
Our concern is

- a quick method in order to make real time computation, and also
- having enough precision to solve engineering problems.

[1] CA wiki.

The Continuum Model

The equations governing incompressible Newtonian fluid flows are

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho} \nabla p + \nu \Delta \mathbf{u} + \rho \mathbf{f}_e \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{1}$$

being

- $\mathbf{u}$: velocity,
- ρ: density,
- p: pressure,
- ν: kinematic viscosity,
- $\mathbf{f}_e$: body forces per volume unit.

Fractional-Step Method

In order to solve the momentum equations in (1)

- Artificial compressibility,
- Fractional steps ($\longrightarrow$ FFT).

So the process is basically a splitting, considering Forward Euler time integration the two equations are

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\nabla \cdot (\mathbf{u} \otimes \mathbf{u})^n + \nu \Delta \mathbf{u}^n \tag{2}$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1} \tag{3}$$

namely, the former having convective-diffusive terms and the later the pressure.

Fractional-Step Method

The Fractional-step has three steps

1. solving (2) (predictor step),
2. solving a Poisson equation for pressure

$$\Delta p^{n+1} = \frac{\rho}{\Delta t} \left(\nabla \cdot \mathbf{u}^* \right), \tag{4}$$

3. updating $\mathbf{u}^*$ (corrector step, equation (3)) and obtaining $\mathbf{u}^{n+1}$.

Using Adams-Bashforth 2^{th} order time integration, the momentum equations problem is

$$\mathbf{u}^* = \mathbf{u}^n + \frac{\Delta t}{2} \left[3R(\mathbf{u}^n) - R(\mathbf{u}^{n-1}) \right], \tag{5}$$

where

$$R\left(\mathbf{u}^n\right) = -\nabla \cdot \left(\mathbf{u} \otimes \mathbf{u}\right)^n + \nu \Delta \mathbf{u}^n. \tag{6}$$

Staggered Grids and QUICK stabilization

Using centered schemes for the velocity and pressure in the velocity correction methods produces

- odd-even nodes decoupling equations on pressure and velocity and
- spatial decoupling between pressure and velocity.

Solution: $\rightarrow$ Staggered Grids.

In order to get a *stable* solution for the convective terms, the QUICK approach is used. Its advantages includes

- truncation error $\mathcal{O}(\delta^3)$ on grids with a displacement of $\frac{\delta}{2}$ (where $\delta = \Delta x$, Δy, Δz, in order to represent the 3 spatial dimensions),
- mass conservation and other stability characteristics.

Solving the Linear System

- We have to solve a linear system $\mathbf{Ax} = \mathbf{b}$
- The Discrete Fourier Transform (DFT) is an orthogonal transformation $\tilde{\mathbf{x}} = \mathbf{Ox} = \mathrm{fft}(\mathbf{x})$.
- The inverse transformation $\mathbf{O}^{-1} = \mathbf{O}^T$ is the inverse Fourier Transform $\mathbf{x} = \mathbf{O}^T\tilde{\mathbf{x}} = \mathrm{ifft}(\mathbf{x})$.
- If the operator matrix $\mathbf{A}$ is *homogeneous* (i.e. the stencil is the same at all grid points) and the b.c.'s are periodic, then it can be shown that $\mathbf{O}$ diagonalizes $\mathbf{A}$, i.e. $\mathbf{OAO}^{-1} = \mathbf{D}$.
- So in the transformed basis the system of equations is diagonal

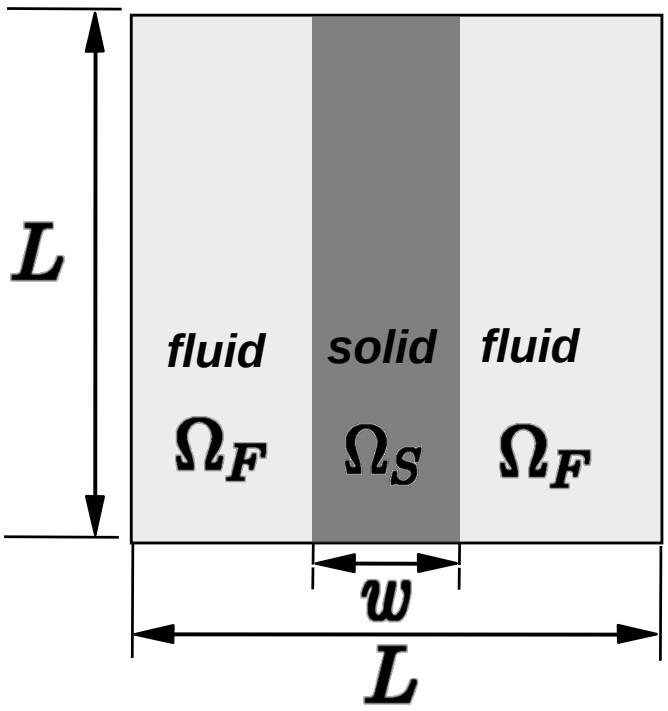
$$\begin{aligned} (\mathbf{OAO}^{-1})\,(\mathbf{Ox}) &= (\mathbf{Ob}), \\ \mathbf{D}\tilde{\mathbf{x}} &= \tilde{\mathbf{b}}, \end{aligned} \tag{7}$$

- For $N = 2^p$ the Fast Fourier Transform (FFT) is an algorithm that computes the DFT (and its inverse) in $O(N\log(N))$ operations.

Solving the Linear System

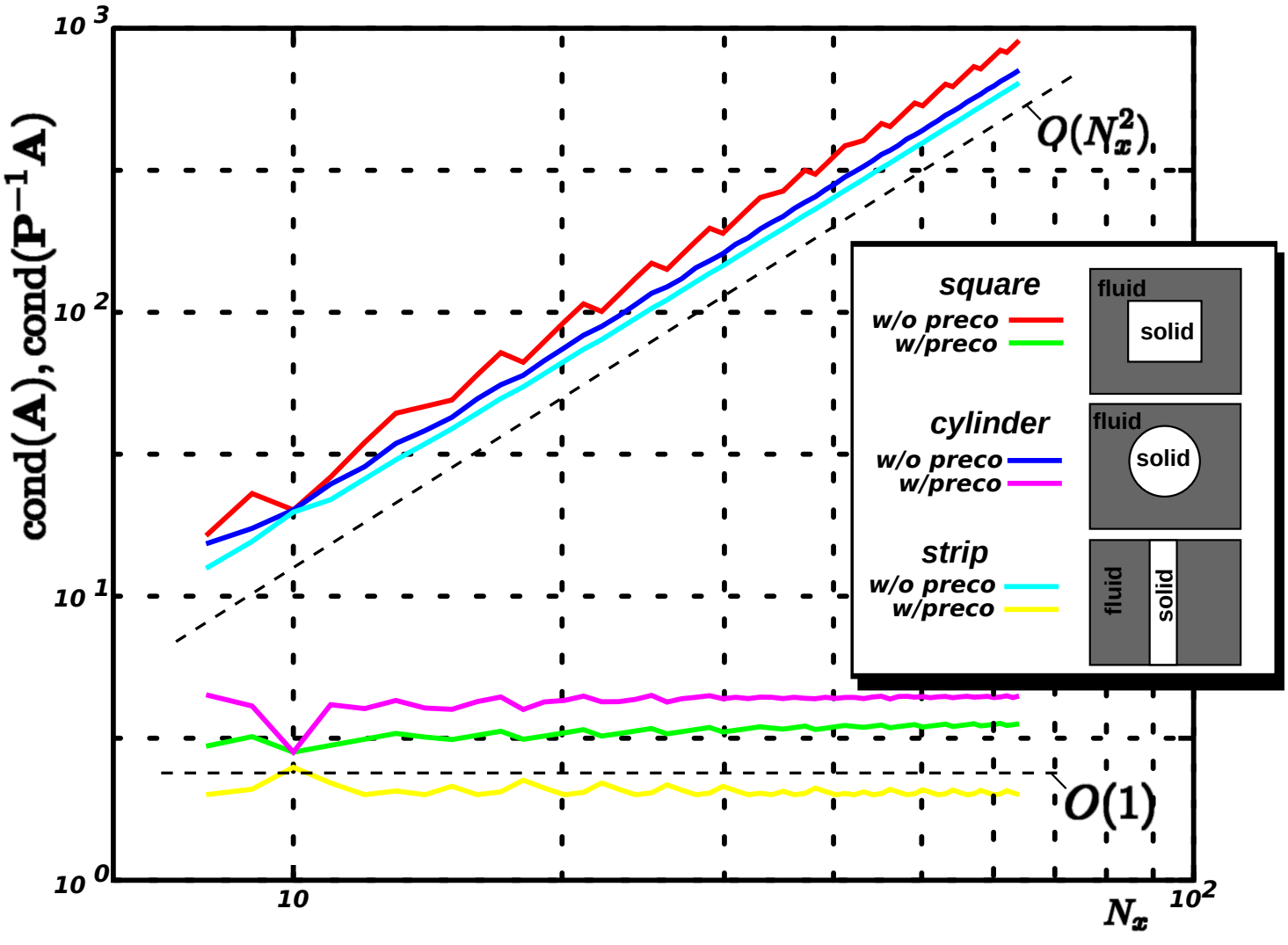
- So the following algorithm computes the solution of the system in $O(N \log(N))$ ops.
  - $\tilde{\mathbf{b}} = \mathrm{fft}(\mathbf{b})$, (transform r.h.s)
  - $\tilde{\mathbf{x}} = \mathbf{D}^{-1}\tilde{\mathbf{b}}$, (solve diagonal system $O(N)$)
  - $\mathbf{x} = \mathrm{ifft}(\tilde{\mathbf{x}})$, (anti-transform to get the sol. vector)
- Total cost: 2 FFT's, plus one element-by-element vector multiply (the reciprocals of the values of the diagonal of $\mathbf{D}$ are precomputed)
- In order to precompute the diagonal values of $\mathbf{D}$,
  - We take any vector $\mathbf{z}$ and compute $\mathbf{y} = \mathbf{A}\mathbf{z}$,
  - then transform $\tilde{\mathbf{z}} = \mathrm{fft}(\mathbf{z})$, $\tilde{\mathbf{y}} = \mathrm{fft}(\mathbf{y})$,
  - $D_{jj} = \tilde{y}_j / \tilde{z}_j$.

Solving the Linear System

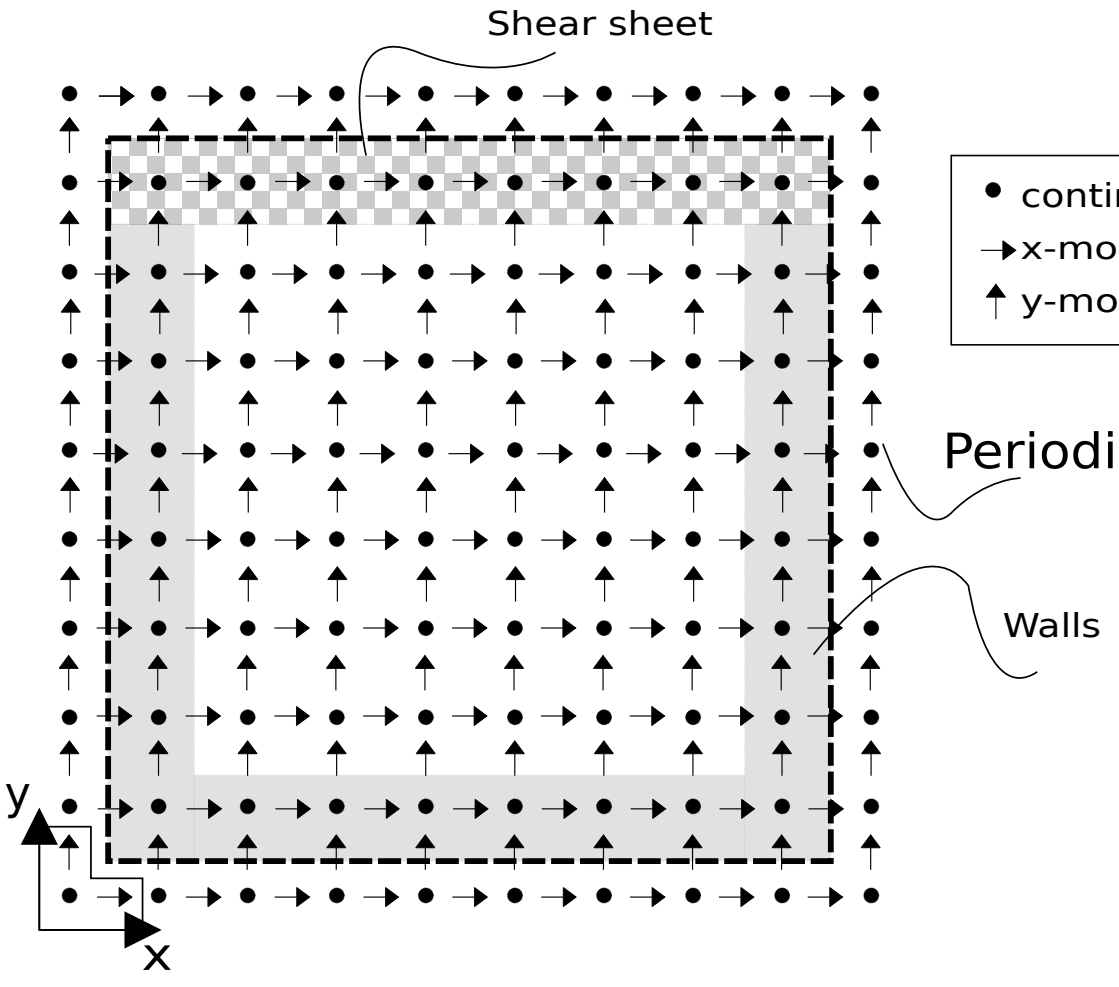


- The system to be solved generally is quite large.
- It can be seen that our domain is compose on fluid and solids.
- The Poisson solver gets *a very approximate* solution of the pressure field.
- In order to take into account the boundary condition an iterative method (CG+FFT) is used.
- It can be shown that the condition number of the preconditioned system remains constant with refinement.

Evolution of the condition number



LDC Test - Simple Precision



$Re = 1000.$

$2x128x128.$

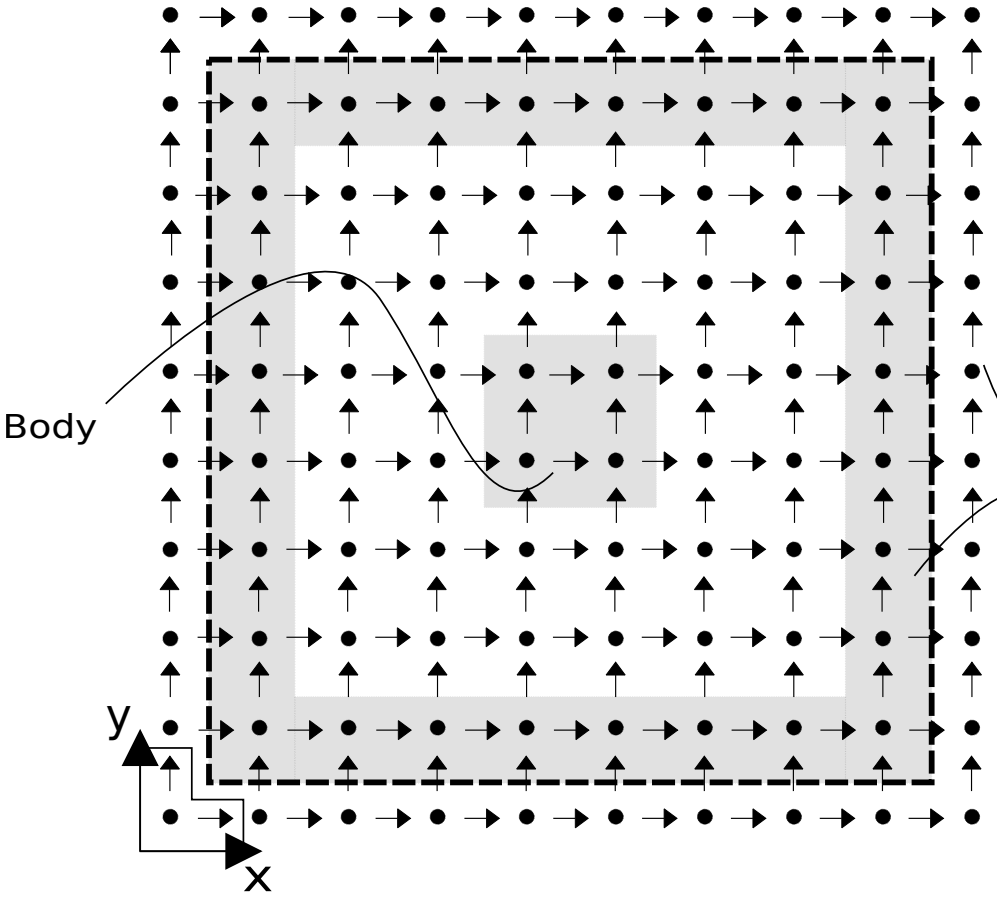
CG: 3 iterations.

Video: ldc test.

Tcomp/Tsim: 1.21.

Tvideo/Tsim: 0.87.

Cube Test - Simple Precision



$CFL \approx 0.5$.

$2x256x256$.

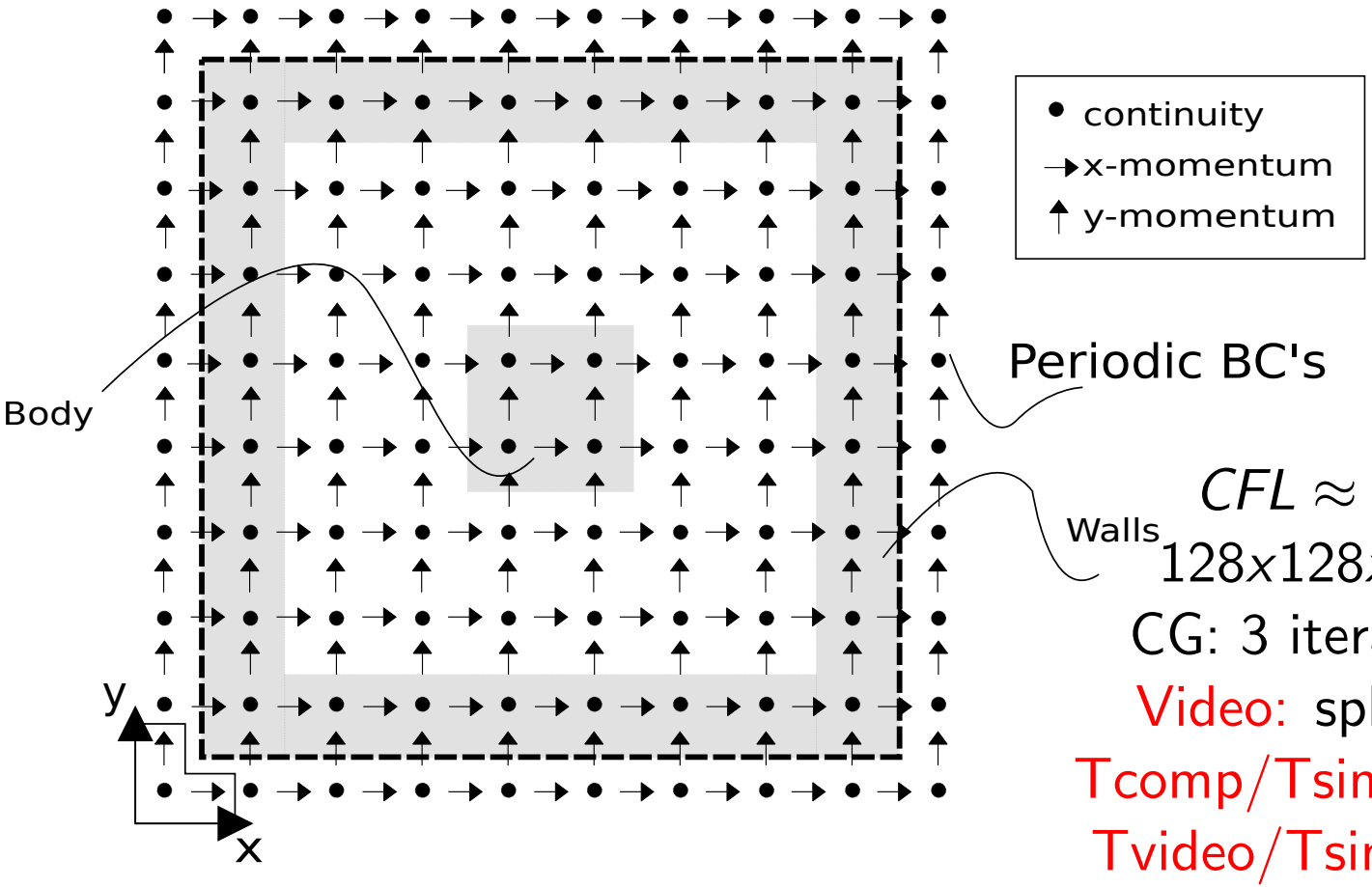
CG: 3 iterations.

Video: cube test.

Tcomp/Tsim: 3.63.

Tvideo/Tsim: 1.30.

Sphere Test - Simple Precision



$CFL \approx 0.5$.

$128x128x128$.

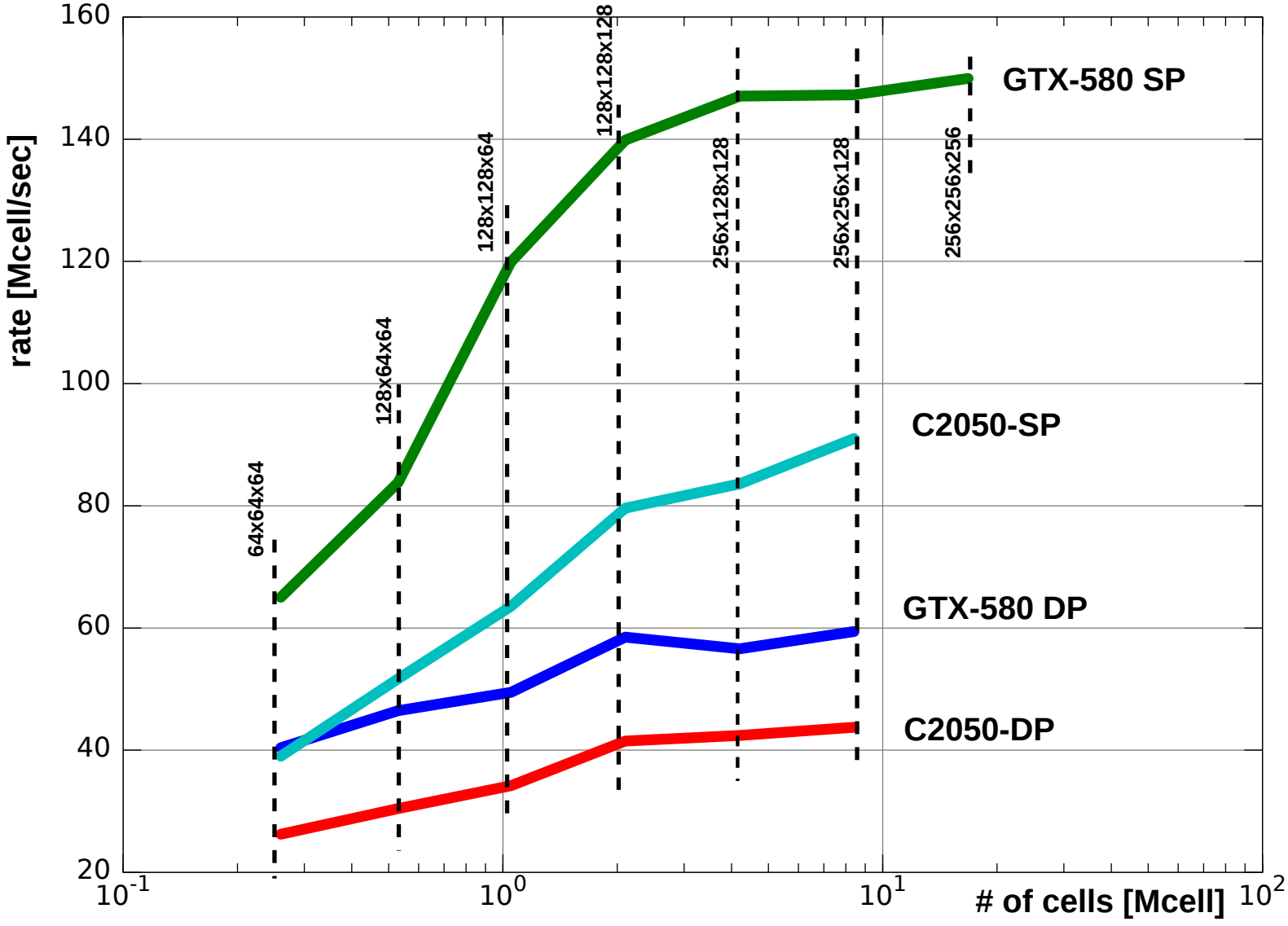
CG: 3 iterations.

Video: sph test.

Tcomp/Tsim: 15.42.

Tvideo/Tsim: 1.33.

GPGPU Results



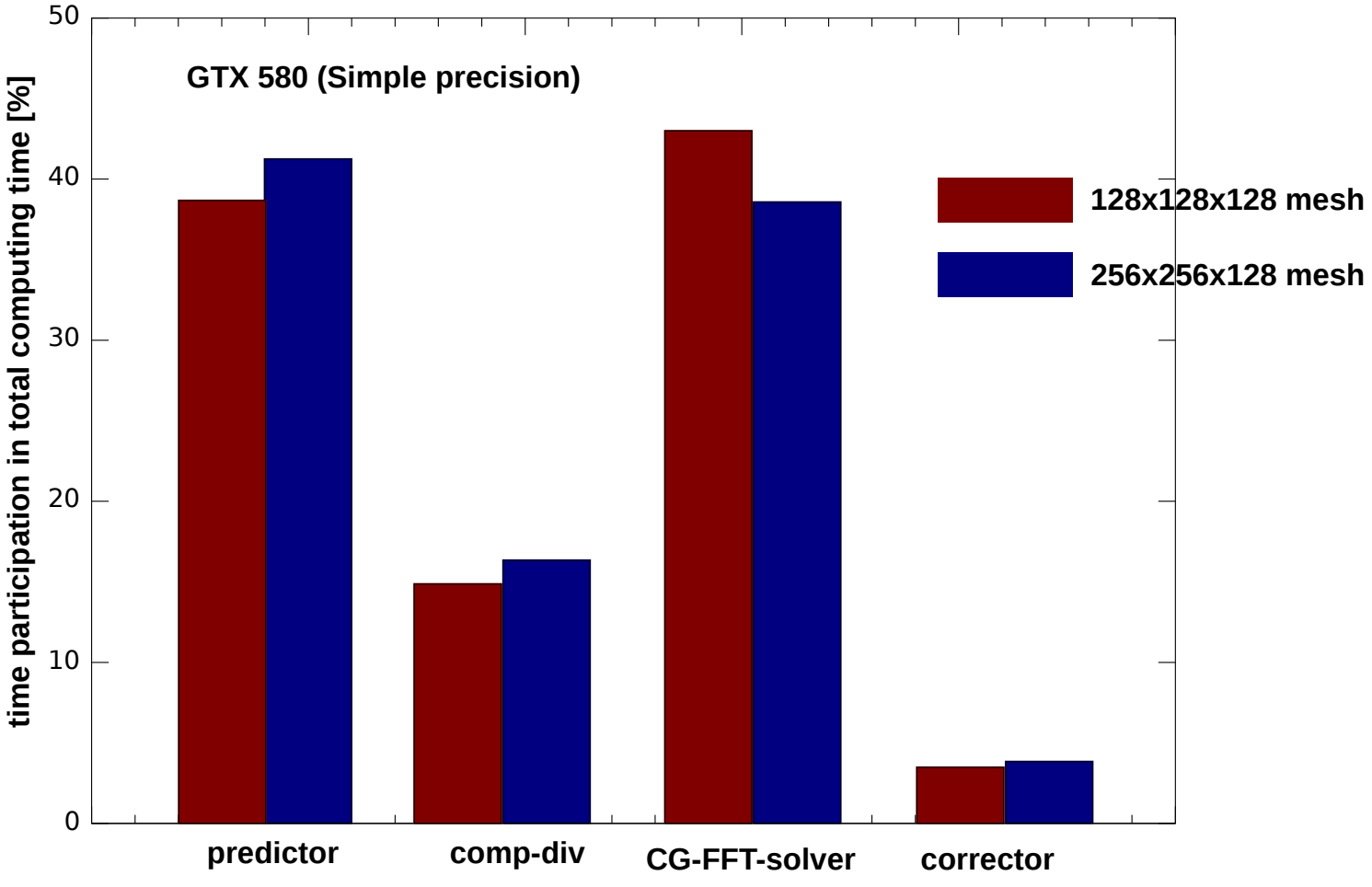
CPU Results

- i7-3820@3.60Ghz (Sandy Bridge), 1 core (sequential): 1.7 Mcell/sec
- i7-950@3.07 (Nehalem), 1 core (sequential): 1.51 Mcell/sec
- Cellrates with nthreads>1, and W3690@3.47Ghz not available at this time.
- BUT, we expect at most 7 to 10 Mcell/secs, so there is speedup factor of 8 to 10, with respect to the GPGPU (GTX-580, DP).

Results analysis

- For a 128x128x128 mesh ($\approx$ 2Mcell), we have a computing time of 2 Mcell/(140 Mcell/sec) = 0.014 secs/time step.
- That means 70 steps/sec.
- A von Neumann stability analysis shows that the QUICK stabilization scheme is unconditionally unstable if advanced in time with Forward Euler.
- With a second order Adams-Bashfort scheme the critical CFL is $\approx$ 0.588 (pure advection).
- For NS eqs. the critical CFL has been found to be somewhat lower ($\approx$ 0.5).
- If $L = 1$, $u = 1$, $h = 1/128$, $\Delta t = 0.5h/u = 0.004$ [sec], so that we can compute in 1 sec, 0.28 secs of simulation time. We say ST/RT=0.28.

Steps computing times



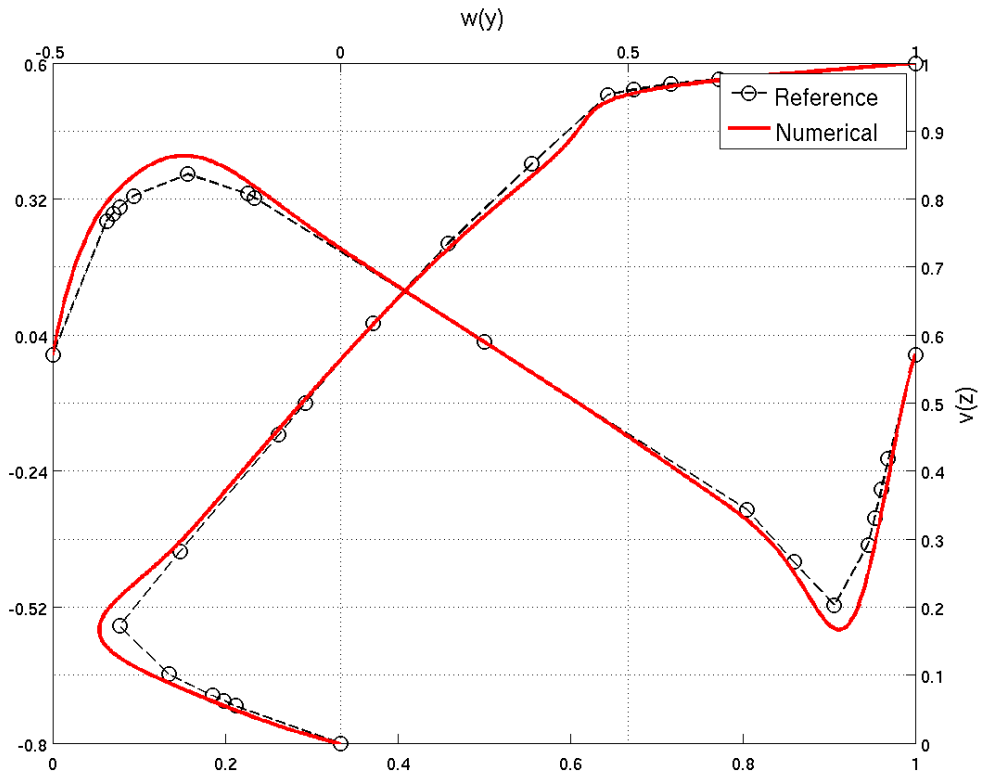
Considerations

- It can be seen on the previous image that the momentum equations are being solved in the same amount of time that the Poisson step.
  - Problem: QUICK - Too much shared memory and/or registers used (other methods are being tested).
- The sphere test show a very rough surface; currently refinement approaches are being tested but far from be fully programmed and tested.
  - Possible Problem: Poor drag computation[2].
- Flow around a cylinder shown good[3] Strouhal prediction (.259 at $Re = 1000$), but drag was not actually computed.

[2] More over, no boundary layer refinement is being done.

[3] Exp: 0.22, PFEM-2: 0.2475, Mittal= 0.25, OpenFOAM: 0.26.

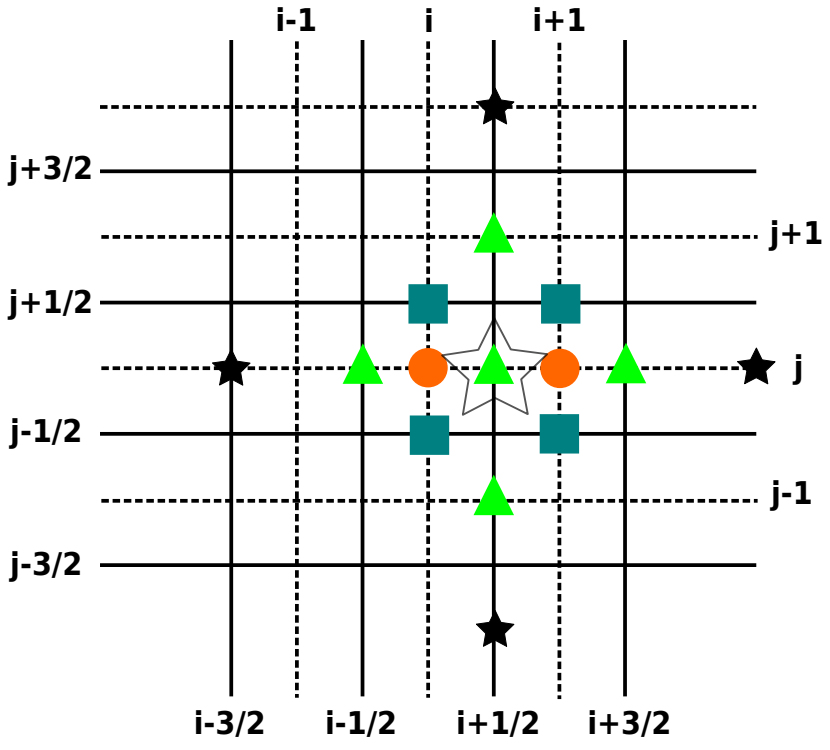
Considerations



- Preliminary results shows that the flow is being *accelerated*, in comparison to the reference data (Ghia et al. 1998).
- Other stabilization schemes are being tested.

Size: 2x512x512.

Conclusions



- QUICK method produces inefficiency due to a highly spread stencil (Too low flops/word relationship). Also, it seems like some artificial *acceleration* is being added.
- Staircase bodies reduce convergence and affect computation of drag forces.
- The speedup achieved, in comparison to a CPU, being 10-20x on SP and 7-10 on DP.

Acknowledge

This work has received financial support of

- Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1141/2007, PICT-0270/2008),
- Universidad Nacional del Litoral (UNL, Argentina, grants CAI+D 2009-65/334, CAI+D-2009-III-4-2) y
- European Research Council (ERC) Advanced Grant, Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG, Dir: Dr. Sergio Idelsohn).

Also we use some development tools under Free Software like GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software like VTK, among many others.