Cluster Comput (2014) 17:243–254
DOI 10.1007/s10586-013-0329-9

GPGPU implementation of the BFECC algorithm for pure advection equations

Santiago D. Costarelli · Mario A. Storti · Rodrigo R. Paz · Lisandro D. Dalcin · Sergio R. Idelsohn

Received: 1 March 2013 / Accepted: 9 October 2013 / Published online: 27 November 2013

Abstract In the present work an implementation of the Back and Forth Error Compensation and Correction (BFECC) algorithm specially suited for running on General-Purpose Graphics Processing Units (GPGPUs) through Nvidia's Compute Unified Device Architecture (CUDA) is analyzed in order to solve transient pure advection equations. The objective is to compare it to a previous explicit version used in a Navier-Stokes solver fully written in CUDA. It turns out that BFECC could be implemented with unconditional stable stability using Semi-Lagrangian time integration allowing larger time steps than Eulerian ones.

Keywords GPGPU · CUDA · BFECC · Semi-Lagrangian · Level-Set · Navier-Stokes

1 Introduction

In recent years GPGPU's (General-Purpose Graphics Processing Units) are being used in HPC (High Performance Computing), specially for problems that can be solved with Cellular Automata (CA) algorithms. In particular, a great effort is being oriented to exploit the great computing power of this hardware on fluid mechanics problems.

In previous works [1–3] a Fractional Step solver for the Navier-Stokes (NS) equations was presented. All steps in the solver were explicit, except for the projection step, which includes the solution of a Poisson equation, which is performed with a Fast Fourier Transform (FFT) technique. Explicit methods fall in the category of CA algorithms and then can be implemented very efficiently on GPGPUs. The momentum equations were solved using a stabilization technique called Quadratic Upwinded Interpolation for Convection Kinematics (QUICK) [4]. The performance of the solver was limited because QUICK performs the computation using a large stencil and is limited by the CFL (Courant-Friedrichs-Lewy) condition. In this article an alternative algorithm for the solution of the momentum equations based on Back and Forth Error Compensation and Correction (BFECC, [5, 6]) is explored. This solver is more efficient since allows for larger CFL numbers; its stability is restricted only by the diffusion term. The algorithm was implemented on Nvidia's GPGPUs using the Compute Unified Device Architecture (CUDA) [7, 8].

This article is an extended version of the presentation "*A Numerical Algorithm for the Solution of Viscous Incompressible Flow on GPU's*" presented at HPCLatAm 2012 [9].

2 Governing equations

The equations being solved are the classical NS equations for incompressible viscous fluid flows

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f},$$
$$\nabla\cdot\mathbf{u} = 0. \tag{1}$$

S.D. Costarelli · M.A. Storti (✉) · R.R. Paz · L.D. Dalcin · S.R. Idelsohn
CIMEC, INTEC - Universidad Nacional del Litoral y CONICET, Santa Fe, Argentina
e-mail: mario.storti@gmail.com

S.R. Idelsohn
Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain

S.R. Idelsohn
International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia (UPC), Gran Capitán s/n, 08034 Barcelona, Spain

where $\mathbf{u}$ is the velocity field, p the pressure field, ρ the density (constant), ν the kinematic viscosity (constant) and $\mathbf{f}$ a body force per unit volume. These equations are intended to be solved with several combination of boundary and initial conditions.

Considering $\mathbf{w}_0$ as an approximation to the solution of $\mathbf{u}$ at time $t = n\Delta t$, one can obtain the solution at $t + \Delta t$ performing successively the following operations [10]

– *Force*: Add force terms

$$\mathbf{w}_1(\mathbf{x}, t) = \mathbf{w}_0(\mathbf{x}, t) + \Delta t \mathbf{f}(\mathbf{x}, t). \tag{2}$$

– *Advection*: the BFECC method following the characteristic field is used. This results in an unconditionally stable step. This is going to be explained on the following sections.
– *Diffusion*: The diffusion equation

$$\frac{\partial \mathbf{w}}{\partial t} = \nu \nabla^2 \mathbf{w}, \tag{3}$$

is solved in the interval $[t, t + \Delta t]$, with $\mathbf{w}(t) = \mathbf{w}_2$, $\mathbf{w}(t + \Delta t) = \mathbf{w}_3$. If periodic boundary conditions are applied, this step can be solved implicitly with an efficient Fast Fourier Transform (FFT) based solver, avoiding the restriction on the Fourier number

$$\mathrm{Fo} = \nu \Delta t / h^2 < \mathrm{Fo}_{\mathrm{crit}} \tag{4}$$

typical for explicit methods. $\mathrm{Fo}_{\mathrm{crit}}$ is a nondimensional constant depending on the spatial dimension and discretization stencil. For the standard second order Poisson discretization used in this paper $\mathrm{Fo}_{\mathrm{crit}} = 1/(2n_d)$, where n_d is the number of spatial dimensions.
– *Projection*: the resulting velocity field is projected onto the divergence free space. This is achieved solving a Poisson equation,

$$\mathbf{w}_4(\mathbf{x}, t) = \mathbf{w}_3(\mathbf{x}, t) - \nabla p, \tag{5}$$

where p is defined as the solution of [11]

$$\begin{aligned} &\nabla^2 p = \nabla \cdot \mathbf{w}_3 \quad \text{in } \Omega, \\ &\nabla p \cdot \mathbf{n} = \mathbf{w}_3 \cdot \mathbf{n} \quad \text{in } \partial\Omega, \end{aligned} \tag{6}$$

$\Omega \subset \mathbb{R}^n$ being the domain on a n-dimensional space, $\partial\Omega$ its boundary and $\mathbf{n}$ the outward normal to ∂D.

2.1 The method of characteristics

The scope of this article is to focus on the solution of the advection step of the exposed solver. Let's consider for the moment a scalar field F that is being advected by the velocity field $\mathbf{u}$,

$$\frac{D_{(m)}F}{Dt} = \frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F = 0, \tag{7}$$

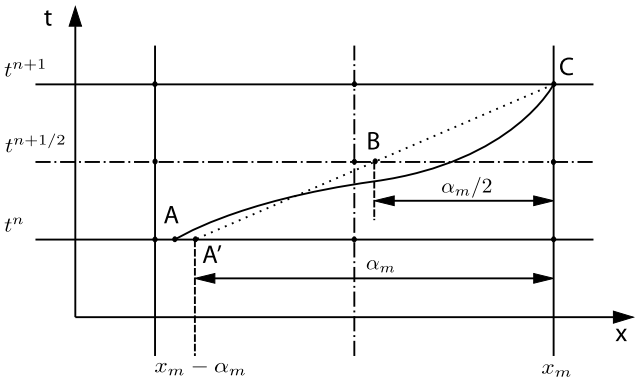


Fig. 1 *Solid line* is a trajectory that connects A and C points. *Dashed line* $\overline{A'C}$ is an approximation based on using the velocity field as a first order predictor

where $D_{(m)}(\cdot)/Dt$ stands for a material derivative, i.e. following fluid particles. An arbitrary dimensional space can be considered, in 3D $\mathbf{x} = (x, y, z)$. Even if the method of characteristics can be explained abstractly (without spatial discretization), to fix ideas a simple Cartesian grid with constant mesh size h is going to be used.

The approach for solving this kind of equations follows that presented on [5]. Considering Eq. (7) in 1D (extensions to higher dimensions are trivial), a splitting reveals that

$$\frac{\partial F}{\partial t} + \frac{dx}{dt}\frac{\partial F}{\partial x} = 0, \tag{8}$$

where

$$\frac{dx}{dt} = u(x, t), \tag{9}$$

that can be solved leading to an ordinary differential equation for the particle position x. Once the trajectories of the particles have been determined, the solution of (8) is obtained by simply stating that the value of F is constant throughout the trajectory [12]. This last equation relates, in particular, points A and C of Fig. 1. The objective is to track back the trajectory passing through C at time $t + \Delta t$ to the point defined as A at time t. This should be done following the trajectory (solid curve); but in the discrete case this is approximated using the velocity field as a first order predictor of the previous position (dashed curve); reaching point A'.

Components of the velocity field on positions located at the mid interval of the actual computational grid are required (i.e. $\mathbf{u}(\mathbf{x}, t^{n+\frac{1}{2}})$). One option is a second order extrapolation of the form

$$\mathbf{u}\big(\mathbf{x}, t^{n+\frac{1}{2}}\big) = \frac{1}{2}\big(3\mathbf{u}\big(\mathbf{x}, t^n\big) - \mathbf{u}\big(\mathbf{x}, t^{n-1}\big)\big). \tag{10}$$

Then, (9) can be integrated to $O(\Delta t^2)$ [13] solving for $\boldsymbol{\alpha} = \Delta \mathbf{x}$ such that

$$\frac{\Delta \mathbf{x}}{\Delta t} = \frac{\boldsymbol{\alpha}}{\Delta t} = \mathbf{u}\left(\mathbf{x} - \frac{\boldsymbol{\alpha}}{2}, t^{n+\frac{1}{2}}\right), \tag{11}$$

whose solution can be obtained by a Fixed-Point iteration

$$\boldsymbol{\alpha}^{(k+1)} = \Delta t \mathbf{u}\left(\mathbf{x} - \frac{\boldsymbol{\alpha}^{(k)}}{2}, t^{n+\frac{1}{2}}\right), \tag{12}$$

taking $\boldsymbol{\alpha}^0 = \mathbf{0}$ on the first iteration $k = 0$. It was found that using two iterations was typically sufficient; this coincides with the results reported by [5].

A sufficient condition for convergence of the iterative procedure is shown in [14], and basically states that Δt must be smaller than the reciprocal of the maximum absolute value of the velocity gradient components, i.e. for the 2D case

$$\Delta t < \left[\max\left(|u_{,x}|, |u_{,y}|, |v_{,x}|, |v_{,y}|\right)\right]^{-1} \tag{13}$$

where u and v are the velocity components on x and y direction respectively). Having the value of $\boldsymbol{\alpha}$, (7) is easily solved by stating that the value of F is constant along the approximated trajectory $\overline{A'C}$, i.e.

$$\frac{F(\mathbf{x}, t^{n+1}) - F(\mathbf{x} - \boldsymbol{\alpha}, t^n)}{\Delta t} = 0. \tag{14}$$

Trilinear (in 3D) spatial interpolation is used when evaluating $\boldsymbol{\alpha}$ and when solving (14). Linear interpolation for $\boldsymbol{\alpha}$ and tricubic spline interpolation for F was proposed in [5] in order to reduce numerical damping. Tricubic spline interpolation has also the property of exactly conserving mass for solenoidal flows. Analysis of the stability properties of the semi-Lagrangian advection scheme [15] shows that it is possible to integrate (14) for CFL numbers greater than one; where $\text{CFL} = u_{\max} \Delta t / h$, and $u_{\max}$ being the maximum velocity in the flow field.

2.2 Back and forth error compensation and correction

Following [16] let's suppose that there exists some operator $L(.,.)$ that performs an approximate solution to the advection operator, i.e.

$$F^{n+1} = L\left(\mathbf{u}, F^n\right). \tag{15}$$

In the continuum case a forward operation $L(-\mathbf{u}, \cdot)$ exactly compensates the backward operation $L(\mathbf{u}, \cdot)$ (see Fig. 2), i.e. after applying a backward and forward operation the same initial field is obtained. (Note that this is valid only for the pure advection case, with no diffusion). Of course, this is no longer true for the discrete operators due to numerical errors. Lets say that the backward operation introduces an error denoted as e. If this error is purely dissipative, then the forward operation will introduce the same error, so after the backward and forward steps a field $\bar{F} = F^n + 2e$. Then, an explicit expression for e can be readily obtained as

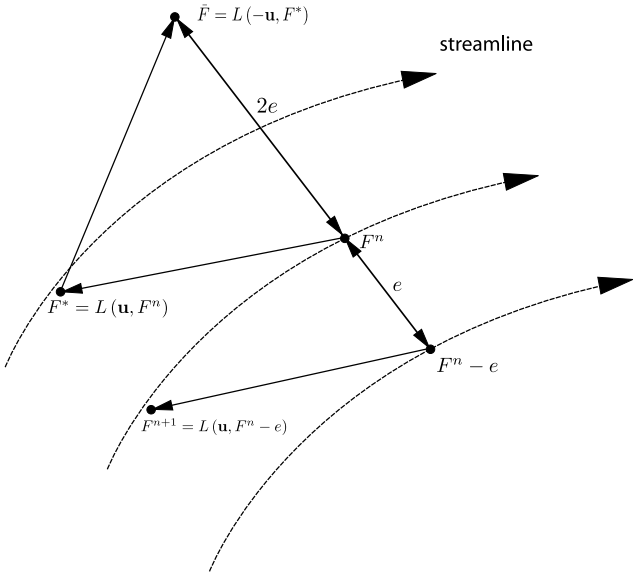


Fig. 2 Schematic BFECC operation over a streamline field and using $L(.,.)$ as the advection operator for the scalar field F

$$e = -\frac{1}{2}\left(F^n - \bar{F}\right). \tag{16}$$

This error can be subtracted (this is called as *error compensation*, hence the name of the method) from F^n and then the field is advected. This method has been proven to be second order accurate in both, time and space [17].

Considering the advection operator $L(.,.)$ as the Semi-Lagrangian one explained on Sect. 2.1, BFECC is defined as follows

$$F^* = L\left(\mathbf{u}, F^n\right)$$
$$\bar{F} = L\left(-\mathbf{u}, F^*\right)$$
$$F^* = F^n + \left(F^n - \bar{F}\right)/2$$
$$F^{n+1} = L\left(\mathbf{u}, F^*\right).$$

In this way the order of accuracy of the Semi-Lagrangian scheme can be raised from one to two at the expense of computing three times the advection operator [17].

As there is no strong evidence on the time dependence of e a variation of the algorithm was proposed in [17]. After the first two steps, forward and backward, the error is computed and used as a correction directly for the backward solution $L(\mathbf{u}, F^n)$ instead of applying a new forward step with a modified field. In this article the standard BFECC is used, since the modified version introduced previously performed worst (in terms of accuracy) in the cases studied.

The proposed scheme does not preserve positivity, i.e. it does not satisfy the Discrete Maximum Principle [17], i.e. overshoots/undershoots may be observed at sharp discontinuities. A basic correction was proposed in the mentioned reference and tested on the transport of a 1D square signal. This correction was implemented by the authors, but the improvement obtained was not worth for the extra computational effort needed.

3 Numerical examples: scalar transport

In typical solid-fluid applications F is used for representing the region occupied by each phase, the fluid occupying the region Ω where $F \geq 0$ and the other the complement.

3.1 Unbiased level-set contouring algorithm (ULCA)

First of all, given an initial region Ω the level set function F must be computed. The algorithm used in this article follows that presented on [18]. It turns out that even with coarse grids (i.e. 50×50) the numerical error obtained by the algorithm developed differs at most by 1 %.

3.2 Error indicator based on the L1-norm computation of the characteristic function

One way to assess the quality of the discrete transport algorithm is to compute the area (area in 2D, volume in 3D) occupied by one fluid, i.e. the $F \geq 0$ region, for instance, since for a solenoidal field this area should be preserved. So the error indicator would be

$$E_a = \big||\Omega^n| - |\Omega^0|\big|. \tag{17}$$

In this equation the absolute value $|\cdot|$ for sets stands for the area (measure) of the set. Ω^0 (Ω^n) is the region where the initial (final) level set function is nonnegative $F^0 \geq 0$ ($F^n \geq 0$).

But checking only area changes may be misleading, because the transport algorithm may distort the region keeping the same area. If the velocity field is a pure rotation, then after a full rotation the geometry F^n should be the same as the initial F^0. Then the improved error indicator E (see [19]) consists basically in measuring the area where the indicator functions $H(F)$ for the initial and final configuration do not coincide, i.e. the sum of the red and green areas in Fig. 3,

$$\begin{aligned} E &= \int_\Omega \frac{1}{L}\big|H\big(F^0\big) - H\big(F^n\big)\big|d\Omega \\ &= |\Omega^n - \Omega^0| + |\Omega^0 - \Omega^n|. \end{aligned} \tag{18}$$

where the Heaviside function $H(.)$ is defined as $H(\phi) = 1$ if $\phi \leq 0$ and $H(\phi) = 0$ otherwise. Note that in the second line of (18) the subtraction operator stands for set difference. This E indicator is null only if the regions are coincident. Note that if the absolute value is removed in the definition of E with the H differences then we would get the area difference, i.e.

$$\begin{aligned} E_a &= \big||\Omega^n| - |\Omega^0|\big| \\ &= \left|\int_\Omega \big(H\big(F^0\big) - H\big(F^n\big)\big)d\Omega\right|. \end{aligned} \tag{19}$$

The error indicator E can be computed numerically by subintegration, i.e.

- Refine the computational grid into smaller subcells, i.e. each cell is divided in $n_{\text{sc}} \times n_{\text{sc}}$ subcells (typically $n_{\text{sc}} = 10$).
- Interpolate bilinearly the values of F^0 and F^n over this finer mesh.
- Compute the L_1-norm as

$$\begin{aligned} E &= \int_\Omega \frac{1}{L}\big|H\big(F^0\big) - H\big(F^n\big)\big|d\Omega \\ &\approx \frac{\Omega_c}{L}\sum_{i,j=1}^{N}\big|H\big(F^0_{ij}\big) - H\big(F^n_{ij}\big)\big|, \end{aligned} \tag{20}$$

  where L is the expected perimeter, N is the amount of subcells per dimension, and Ω_c is the subcell area.

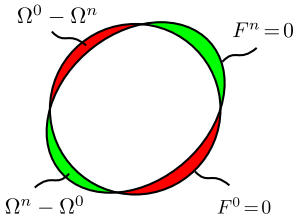


Fig. 3 Definition of the L_1 error E in the level set indicators

For velocity fields that are not rigid body rotations, the benchmark can be extended in the following way. The initial level set function F^0 is advected under the velocity field $\mathbf{u}$ for a certain time $T/2$ to a deformed state $F^{n/2}$. Then this distorted geometry is advected back with the velocity field reversed ($\mathbf{u} \to -\mathbf{u}$). Again, the same region should be recovered and the error indicator E can be used.

3.3 Performance measurement

The efficiency of the algorithm is assessed by computing the rate at which cells are processed, using the following data

- N_{it}: number of iterations performed;
- N_{cells}: number of cells computed on each time step;
- t_{total}: time (in seconds) taken to perform one complete BFECC iteration;

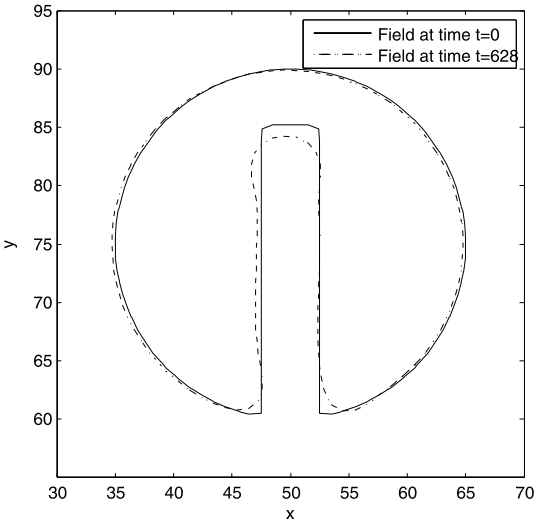


Fig. 4 Zalesak's disk results after one full revolution with 100 grid point per direction and CFL = 4.9

with $N_{cells} = (N_{dim})^d$, d being the number of dimensions and N_{dim} the amount of grid points per direction (only structured Cartesian constant-mesh-size grids are considered). So the computing rate is computed as

$$\text{Rate} = \frac{N_{it} \times N_{cells}}{t_{total}} \left[\frac{cells}{sec}\right]. \tag{21}$$

3.4 2D examples

3.4.1 Zalesak's disk

This test consists in the advection of a region composed of a circle with a slot [20].

- The computational domain is $\Omega \subset \mathbb{R}^2 : [0; 100] \times [0; 100]$;
- The advected region is a circle centered at (50; 75) with a radius of 15 and a slot of width 5 and height 25.
- The velocity field is a rigid body rotation around the center of the domain with a period of 628 time units,

$$\begin{aligned} u &= (\pi/314)(50 - y), \\ v &= (\pi/314)(x - 50). \end{aligned} \tag{22}$$

The initial field and the error indicator (see Sect. 3.2) after one revolution are shown on Fig. 4 and Table 1. The CFL number was 4.9. For reference the results obtained with a particle level set method [21] are shown.

The results obtained by BFECC are similar in accuracy to those obtained by the particle method, but far more accurate than those obtained by other simple level set methods (see [21]).

3.4.2 Single vortex

While Zalesak's disk test is a good indicator of numerical diffusion in an interface-capturing method, it does not test the ability to preserve small scale properties of the fluid flow. The single vortex benchmark proposed in [22] tests this property of tearing flows.

Table 1 Zalesak's disk. Error indicator values obtained with the present BFECC code. In parentheses numerical results obtained by [21] using Particle Level-Set method and Semi-Lagrangian time integration

	Grid cells	Initial area	Area loss [%]	L_1-error
	Theoretical	581.85	–	–
One revolution	50	574.38	4.98 (3.09)	0.85 (0.434)
	100	581.14	0.78 (1.07)	0.26 (0.181)
	200	580.92	0.11 (0.22)	0.06 (0.105)
Two revolutions	50	574.38	4.66 (2.66)	1.75 (0.610)
	100	581.14	4.02 (1.01)	0.52 (0.206)
	200	580.92	0.34 (0.22)	0.12 (0.103)

The velocity field is defined by the stream function

$$\psi(\mathbf{x}) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \cos\left(\frac{\pi t}{T}\right), \tag{23}$$

so, the velocity components are

$$\begin{aligned} u &= \psi_{,x} = \sin^2(\pi x) \sin(2\pi y) \cos\left(\frac{\pi t}{T}\right), \\ v &= -\psi_{,y} = -\sin^2(\pi y) \sin(2\pi x) \cos\left(\frac{\pi t}{T}\right). \end{aligned} \tag{24}$$

The test considers a unit square computational domain with a circle of radius 0.15 placed at (0.5; 0.75). The initial conditions, the maximum stretching and the results after the forward and backward steps with $T = 8$, $N = 256$ grid points per dimension and CFL = 4.9 are shown on Fig. 5. The error indicator (see Sect. 3.2) is shown in Table 2.

3.5 3D examples

3.5.1 Volume preservation computation

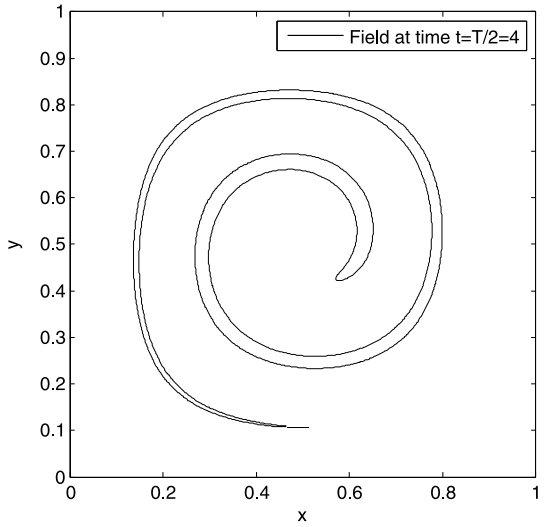
In this test the volume preservation of the scheme [23] is checked. The volume is computed as

$$\int_{\Omega} H(\phi) d\mathbf{x} = h^3 \sum_{i,j=1}^{N} \tilde{H}(\phi_{i,j}) \tag{25}$$

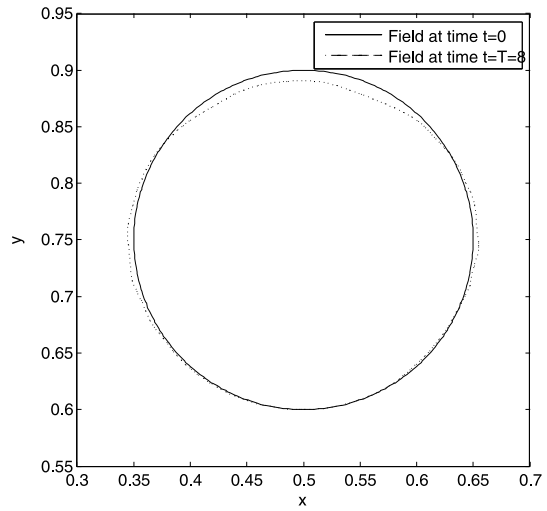
where $\tilde{H}$ is a regularized version of the Heaviside function,

$$\tilde{H}(x) = \begin{cases} 0, & \text{if } x < -\epsilon \\ \frac{1}{2} + \frac{x}{2\epsilon} + \frac{1}{2\pi} \sin(\frac{\pi x}{\epsilon}), & \text{if } -\epsilon \le x \le \epsilon, \\ 1, & \text{if } x > \epsilon \end{cases} \tag{26}$$

and $\epsilon = 1.5h$.



(a) Field at time t = T/2 = 4.



(b) Field at time t = T = 8.

Fig. 5 Single vortex test using 256 grid points per direction and CFL = 4.9

Table 2 Single vortex test. Area loss and error indicator values obtained with the present BFECC code. In parentheses numerical results obtained using Particle Level-Set method and Semi-Lagrangian time integration [23]

Grid cells	Initial area	Area loss [%]	L1-error
Theoretical	0.0707	–	–
64	0.0443	37.09 (1.81)	0.033 (0.00300)
128	0.0652	7.66 (0.71)	0.014 (0.00100)
256	0.0696	1.44 (0.35)	0.003 (0.00059)

3.5.2 Zalesak's disk

This benchmark is the natural extension of the 2D version (see Sect. 3.4.1) to 3D. The domain $\Omega \subset \mathbb{R}^3 : [0; 100] \times [0; 100] \times [0; 100]$ and 100 grid points per direction. The disk is now a sphere centered at (50; 75; 50) with radius 15 and a slot of width 5 and height 25. The velocity field is rigid body rotation centered at $z = 50$, is

$$u = (\pi/314)(50 - y),$$
$$v = (\pi/314)(x - 50), \tag{27}$$
$$w = 0.$$

The level set after one complete revolution is shown in Fig. 6. The numerical results obtained are shown on Table 3 being the accuracy comparable to that achieved with particle methods.

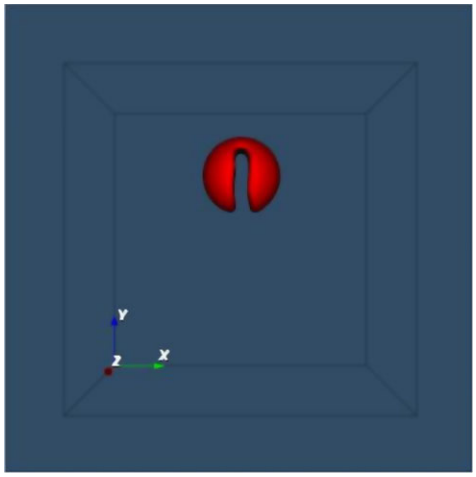


Fig. 6 3D Zalesak's disk after one complete revolution with 100 points per dimension and CFL = 4.9

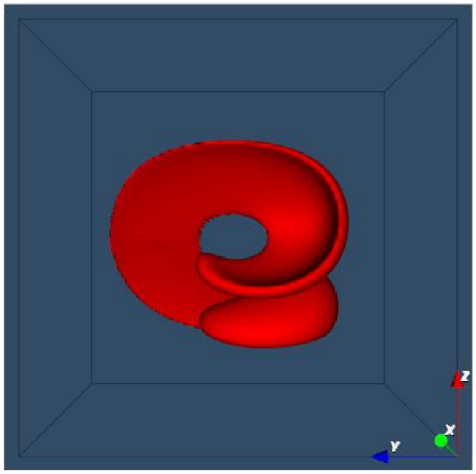
Table 3 Volume loss for the Zalesak's disk problem with the present method. In parentheses results obtained with Particle Level-Set method [23] and Semi-Lagrangian time integration [24]

Cells per dimension	Initial volume	Volume loss [%]
100	11176	0.51 (2.3 [23])
128	11106	0.27 (0.9 [24])

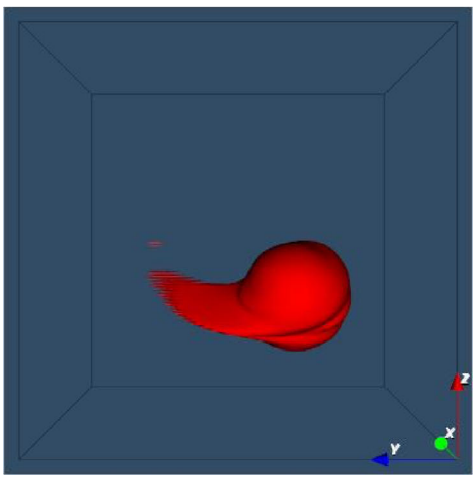
3.5.3 Deformation field

First proposed by LeVeque [25] this is a 3D test combining deformations on x-y and x-z planes. The velocity field is given by

$$u = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right),$$
$$v = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi x)\cos\left(\frac{\pi t}{T}\right), \tag{28}$$
$$w = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{T}\right),$$



(a) Field at time t = T/2 = 1.5.



(b) Field at time t = T = 3.

Fig. 7 3D deformation field test using 128 points per direction and CFL = 4.9

Table 4 Volume loss for the 3D deformation field case with the proposed BFECC algorithm. In parentheses values obtained with Particle Level-Set [23] and Semi-Lagrangian time integration [24]

Cells per dimension	Initial volume	Volume loss [%]
100	0.014237	3.76 (2.6 [23])
128	0.014197	4.49 (1.8 [24])

modulated in time, being the half period $T = 3$. The domain is a unit cube and the advected region is initially a sphere of radius 0.15 placed at (0.35; 0.35; 0.35). Since the velocity field is reversed for $t > T/2$ the geometry at the final position $T = 3$ should be the same as for the initial position $t = 0$. The numerical results are shown in Table 4. Figure 7 shows the field at maximum stretching $t = T/2 = 1.5$, and the final position at time $t = T = 3$. This test is particularly difficult due to small scale structures that can not be resolved, leaving a trailing tail behind the sphere not present in the initial condition. However the results obtained with the present algorithm are much better than those reported with comparable algorithms [23, 24].

4 Applications to Navier-Stokes equations

4.1 Solving momentum equations by BFECC and semi-Lagrangian time integration

The extension of the application of the method of characteristics with BFECC described in Sect. 2.1 to the advection term in the NS equations will be described. The material derivative is used to approximate the advection term as follows

$$\frac{D_{(m)}\mathbf{u}(\mathbf{x};t)}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}^* \cdot \nabla \mathbf{u} = \mathbf{0}, \tag{29}$$

where $\mathbf{u} = \{u, v, w\}$ are the velocity components on x, y and z directions. $\mathbf{u}^*$ is an intermediate (predicted) velocity field.

Staggered grids are used to prevent uncoupling of the velocity and pressure fields [2]. This must be taken into account in the application of the method of characteristics, as explained in Sect. 2.1, since velocities are now located in non coincident grids. Considering for instance pure transport of the x-velocity vector alone, the discrete equation is

$$\left(\frac{D_{(m)}u(\mathbf{x};t)}{Dt}\right)_{i+1/2,j,k} = 0, \tag{30}$$

then

$$\boldsymbol{\alpha}^{(l+1)} = \Delta t \mathbf{u}^*\left(\mathbf{x} - \boldsymbol{\alpha}^{(l)}/2; t + \Delta t\right)_{i+1/2,j,k}. \tag{31}$$

where l is the iteration index in the fixed point algorithm.

The explicit expressions of each component of $\boldsymbol{\alpha}$ are as follows

$$\alpha_x^{(l+1)} = \Delta t u^*_{i+1/2-\alpha_x^{(l)}/2,\, j-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2}, \tag{32}$$

$$\alpha_y^{(l+1)} = \Delta t \left(v^*\right)^c_{i+1/2-\alpha_x^{(l)}/2,\, j-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2}, \tag{33}$$

$$\alpha_z^{(l+1)} = \Delta t \left(w^*\right)^c_{i+1/2-\alpha_x^{(l)}/2,\, j-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2}, \tag{34}$$

where $()^c$ stands for *co located* fields, i.e.

$$\left(v^*\right)^c_{i,j+1/2,k} = \frac{\Delta t}{4}\Big\{ v^*_{i+1-\alpha_x^{(l)}/2,\, j+1/2-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2} \tag{35}$$

$$+ v^*_{i-\alpha_x^{(l)}/2,\, j+1/2-\alpha_y^{(l)}/2,\, k+1-\alpha_z^{(l)}/2} \tag{36}$$

$$+ v^*_{i+1-\alpha_x^{(l)}/2,\, j-1/2-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2} \tag{37}$$

$$+ v^*_{i-\alpha_x^{(l)}/2,\, j-1/2-\alpha_y^{(l)}/2,\, k-\alpha_z^{(l)}/2}\Big\}. \tag{38}$$

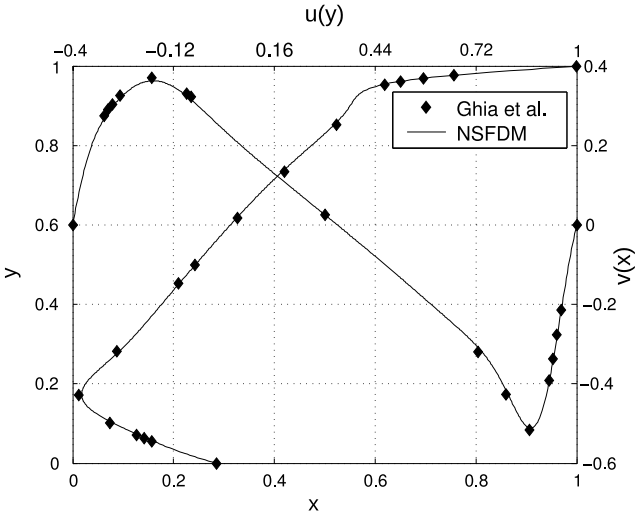


Fig. 8 Lid driven square cavity. Results obtained at Re = 1000 using a grid of 512 × 512

Finally, after convergence of the fixed point iteration ($\boldsymbol{\alpha}^{(l)} \to \boldsymbol{\alpha}$),

$$u^{n+1}_{i+1/2,j,k} = u^{n}_{i+1/2-\alpha_x/2,\, j-\alpha_y/2,\, k-\alpha_z/2}. \tag{39}$$

The same scheme is applied for the transport of the other velocity components v and w.

4.2 Poisson equation for pressure

The projection step (as described in Sect. 2) is solved using the Conjugate Gradient (CG) iterative solver with a FFT preconditioning. A complete analysis in this subject can be found in [1]. Of course, the number of iterations is a key point for the performance of the algorithm. The values reported in the numerical simulations have been obtained with tolerance ($O(10^{-3})$) which requires typically 3 iterations. The FFT is computed with the CUFFT implementation that is included in CUDA [26].

4.3 Numerical examples

4.3.1 2D lid-driven cavity

This is a classical internal flow test in a square domain of side length L. The shear velocity $u = 1$, $v = 0$ is imposed at the top, whereas all the other sides are imposed the non-slip condition ($u = v = 0$). The numerical results obtained at Re = 1000 (taking L as characteristic length) are shown on Fig. 8, compared with the results of Ghia et.al. [27]. The performance obtained, measured in [secs/Mcells] (i.e. seconds of computation to compute one million cells), is shown on Table 5. In this case stability is diffusion-controlled (see Eq. (4)), so the CFL is effectively reduced to CFL < 0.48. This is not reflected in the rates of the tables because they are expressed in cells/sec, but it reduces the efficiency of the code, since the rate of computation is proportional to CFL.

Table 5 Lid driven square cavity at Re = 1000. Performance measured in [Mcells/sec] on a NVIDIA GTX 580

	Simple	Double
64 × 64	0.62	0.61
128 × 128	2.50	2.22
256 × 256	9.09	7.14
512 × 512	25.00	16.66

4.3.2 2D flow past circular cylinder

This is a homogeneous stream of unperturbed velocity u_∞ impinging on a cylinder of diameter D (see [28–31]). The length L and height H of the computational domain are related $L = H = 15D$. This relation was chosen in order to minimize the adverse effects of boundary conditions on the numerical results. The body is represented as a staircase geometry. Non-slip boundary conditions are imposed on the solid boundary.

The far field velocity $\mathbf{u}_\infty = (u_\infty, 0)$ is fixed on the inlet, top and bottom boundaries with $u_\infty = 1$. At the downstream boundary a Neumann-type boundary condition for the velocity is specified that corresponds to zero viscous stress vector [32]. The kinematic viscosity is adjusted in order to reach a predefined Reynolds number (Re_D).

The most important physical quantities are the drag and lift forces $F_{x,y}$, and the vortex shedding frequency f. The corresponding nondimensional quantities are the drag and lift coefficients, and the Strouhal number,

$$C_{d,l} = \frac{F_{x,y}}{\frac{1}{2}\rho {u_\infty}^2 D}, \qquad \mathrm{St} = \frac{f D}{u_\infty}. \tag{40}$$

To compute drag and lift a momentum balance over a control surface enclosing a square region containing the cylinder is used [33]. Since the control surface is close to the body, the error from not considering the inertia terms in the fluid inside the control surface can be neglected. The vortex shedding frequency is computed by counting the number of periods (crossing by zero) in lift history.

The results obtained at $\mathrm{Re}_D = 1000$ are shown on Table 6, and the performance obtained is shown on Table 7.

In this case, the stability of the scheme is controlled by advection, not diffusion, so a CFL up to 5 can be used. A time history of the coefficients is shown on Fig. 9.

4.3.3 3D lid-driven cavity

This is the natural extension of the 2D lid driven square cavity example in Sect. 4.3.1 to 3D. Non-slip condition $u = v = w = 0$ is imposed at all the walls, except at the top, where $u = 1$, $v = w = 0$. The numerical results obtained at

Table 6 Flow past cylinder at $\mathrm{Re}_D = 1000$

		C_d	C_l	St
2D	Present formulation	1.56	1.3	0.211
	PFEM-2 [31]	1.639	1.63	0.2475
	FEM [30]	1.48	1.36	0.21
3D	Experimental	1.00 [28]		0.21 [29]
	Present formulation	1.021	0.533	0.183
	PFEM-2 [31]	1.16	0.2 to 0.3	0.185
	OpenFOAM [31]	1.22	0.5	0.195

Table 7 Flow past cylinder at $\mathrm{Re}_D = 1000$. Performance measured in [Mcells/sec] on a NVIDIA GTX 580

		Simple	Double
2D	256×128	5.0	4.5
	512×256	16.7	11.1
	1024×512	33.3	20.0
3D	$64 \times 256 \times 256$	20.0	7.7

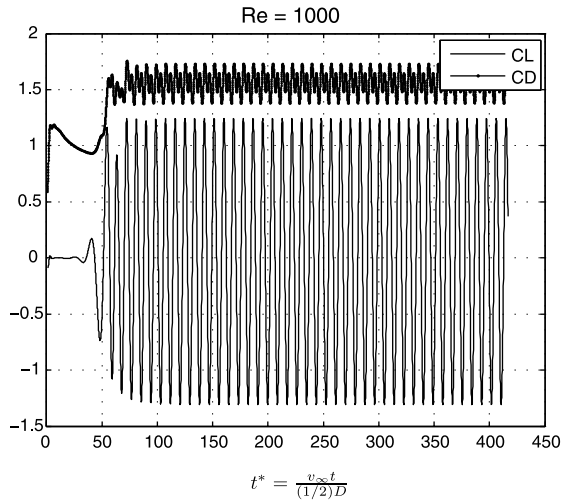


Fig. 9 Results obtained at $\mathrm{Re} = 1000$ using a grid of 1024×512

$\mathrm{Re} = 1000$ are shown on Fig. 10, and the performance obtained is shown on Table 8.

The performance of the code for performing a *real time computation*, that is computing the simulation of the physical process at the same as the physical system evolves (see [1]) will be analyzed. From Table 8, the performance obtained for the $128^3 \approx 2$ [Mcells] mesh is about 20 [Mcells/sec], so that 10 time steps per second of computing time can be performed. As the time step for this case is $\Delta t = 0.01$ [s] it can be seen that 0.1 [s] of simulation can be performed in 1 [s] of computation. In this case also, for the finest meshes the stability constraint is diffusion-controlled, i.e. the critical time step is given by (4).

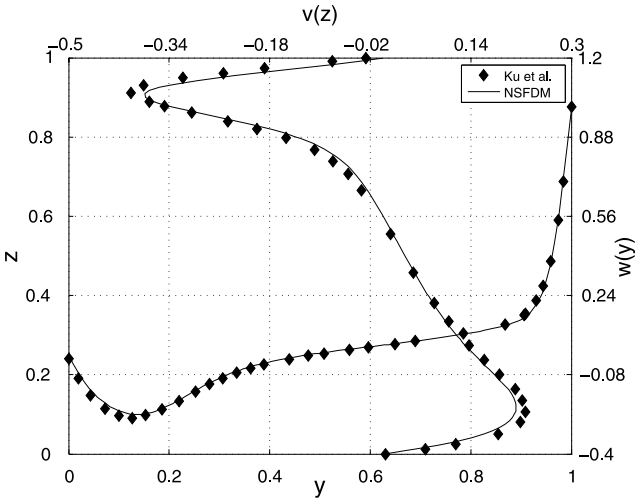


Fig. 10 Results obtained at $\mathrm{Re} = 1000$ using a grid of $128 \times 128 \times 128$

Table 8 Cubic cavity. Computing rates for the whole NS solver (one step) in [Mcell/sec] obtained with the BFECC and QUICK algorithms on a NVIDIA GTX 580. 3 Poisson iterations were used

Simple	[Mcell/sec]	[Mcell/sec]
Cells	QUICK	BFECC
$64 \times 64 \times 64$	29.09	12.38
$128 \times 128 \times 128$	75.74	18.00
$192 \times 192 \times 192$	78.32	17.81
Double	[Mcell/sec]	[Mcell/sec]
Cells	QUICK	BFECC
$64 \times 64 \times 64$	15.9	5.23
$128 \times 128 \times 128$	28.6	7.29
$192 \times 192 \times 192$	30.3	7.52

4.3.4 3D flow past circular cylinder

The cylinder case has been also run in the 3D case, with periodic boundary conditions along the cylinder axis direction. Performance results obtained at $\mathrm{Re}_D = 1000$ are shown on Table 6 and time history of the coefficients is shown on Fig. 11. The performance obtained is shown on Table 7. In this case stability is advection-controlled and $\Delta t = 0.023$ was used. The mesh has 7.1 Mcells and 0.23 [s] of simulation can be performed in 1 [s] of computation.

4.4 Performance analysis

The performance obtained by the present algorithm for advancing one time step in the NS solver (the four stages as described in Sect. 2) in the case of the 3D cubic cavity is shown on Table 8. As a reference, this performance is compared with a previous implementation of the solver using QUICK [1]. Details of the two solvers follows

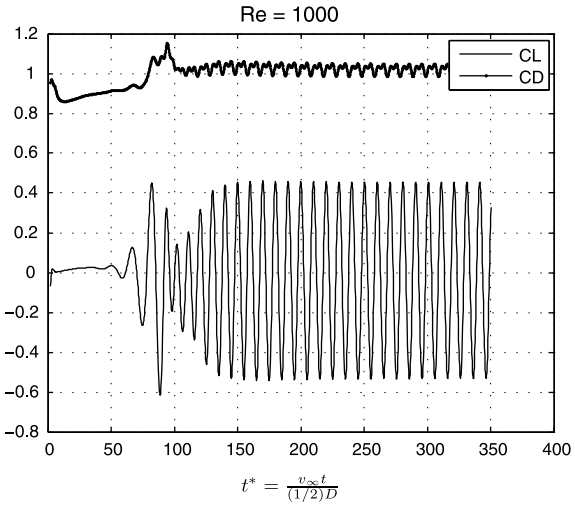


Fig. 11 Results obtained at $\mathrm{Re}_D = 1000$ using a grid of $96 \times 384 \times 192$

– QUICK: this version of the momentum advection uses a high performance algorithm that makes use of shared memory, but has the drawback of a tight CFL condition, namely CFL < 0.58 for the advection operator. For the NS solver the condition is even somewhat tighter, CFL < 0.5 (see [3]).
– BFECC: the solution of the momentum equations using the BFECC algorithm described in this article. The CUDA implementation uses global memory. Note that while the great advantage of this implementation is that allows for CFL > 1, the cost of tracking the trajectories is proportional to the CFL number, and in addition this has the drawback that restricts the possibility of optimization, due to the spreading in memory accesses. Nevertheless, it will be shown that this algorithm is faster than the QUICK version.

The CUDA kernel proposed is memory bound due to many small accesses from global memory. This results in poor performance achievement but with the advantage of being unconditional stable, in other words, the time step can be chosen without any additional constraints. Even if semi-Lagrangian integration allows CFL > 1, another stability limit can be reached, namely the stability of the diffusion step (viscosity term). If this step is solved explicitly then the Fourier number (4) is restricted to $\mathrm{Fo} < \mathrm{Fo}_{\mathrm{crit}}$. In terms of CFL this restriction can be rewritten as

$$\mathrm{CFL} < \mathrm{Fo}_{\mathrm{crit}} \mathrm{Re}_h, \tag{41}$$

where Re_h is the cell size based Reynolds number. Normally, in many fluid mechanics computations of practical interest the global Re number is very high, and a mesh as fine as possible so as to have a $\mathrm{Re}_h = O(1)$ would be desirable. However that would be computationally too expensive and usually $\mathrm{Re}_h > 1$. But on the other hand Re_h can not be too large, otherwise the small scale features of the flow are lost.

This restriction is overcome by solving the diffusion equation implicitly with a high performance FFT-based pure transient diffusion equations solver. With this modification the NS solver would be unconditionally stable, except for nonlinear instabilities.

Regarding the performance results shown in Table 8, it can be seen that the computing rate of QUICK is at most $4\times$ faster than that of BFECC. So BFECC is more efficient than QUICK whenever used with CFL > 2, being the critical CFL for QUICK 0.5. The CFL used in our simulations is typically CFL ≈ 5, provided that the stability is not diffusion-controlled, see equation (4) and, thus, at this CFL the BFECC version runs 2.5 times faster than the QUICK version.

As a reference, the QUICK algorithm was implemented in CPU using the GNU `g++` compiler (optimization flags `-O3 -funroll-loops`), obtaining a rate of 3.5 Mcell/sec on an Intel i7-3820@3.47 GHz (Sandy Bridge microarchitecture) for large 3D meshes (above 1 Mcell), i.e. 8.6 times slower with respect to the GPU(QUICK) version. Note that this speedup obtained on the GPU is close to the 8x speedup factor obtained for the FFT [1]. This is normal, because for the QUICK implementation a large part of the computing time is spent in the Poisson step. The BFECC(GPU) is only 2.15 times faster than the QUICK(CPU) version in Mcells/sec, but taking into account that the CFL is 10 times larger, the overall speedup is 21.5, i.e. BFECC(GPU) is 21.5 times faster than QUICK(CPU) in computing one second of the same physical process.

Note that the computing rate highly increases with mesh size in the 2D case, both for the lid driven square cavity (see Table 5) and the cylinder flow (see Table 7), whereas it is almost constant or increases slightly in 3D cases. The reason for this is that 2D meshes have much fewer cells; in some cases the finest 2D cell has as much cells as the coarsest 3D one. (For instance the 512^2 and 64^3 meshes have the same number of cells.) Note that the same behaviour is observed for the GPU(QUICK) version and the CUFFT implementation itself [1].

A version of BFECC using shared memory is currently under development by performing the advection at CFL > 1 with repeated applications of advection substeps at CFL < 1. This split in smaller CFL numbers needs higher interpolation schemes, otherwise the dissipation is higher. So, on one hand, more efficient (in terms of Mcell/sec due to the GPGPU capabilities) algorithms may be developed but, on the other hand, high resolution interpolation schemes need to be introduced [17].

5 Conclusions

A GPGPU implementation of the BFECC algorithm with Nvidia's CUDA platform was tested against two classical benchmarks: Zalesak's disk and LeVeque's deformation field, in two and three dimensions. The proposed algorithm is more efficient than the explicit version using QUICK. Currently the performance is limited by the poor pattern access of the global memory, and also BFECC requires three access per time step as compared with QUICK with requires only one, but this is compensated by the higher CFL numbers that can be achieved. Numerical results are presented for the full Navier-Stokes solver, having a better performance than the QUICK version for CFL > 2.

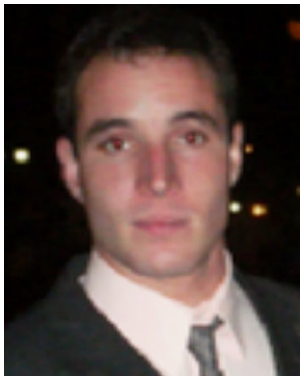
Acknowledgements This work has received financial support of Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1141/2007, PICT-0270/2008, PICT-2492/2010), Universidad Nacional del Litoral (UNL, Argentina, grants CAI+D 2009-65/334, CAI+D-2009-III-4-2) y European Research 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.

References

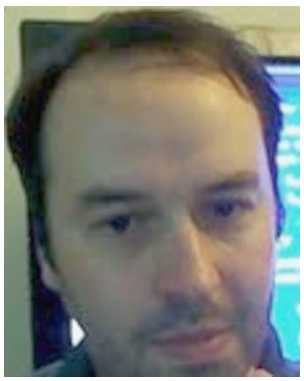
1. Storti, M.A., Paz, R.R., Dalcin, L.D., Costarelli, S.D., Idelsohn, S.R.: A FFT preconditioning technique for the solution of incompressible flow on GPUs. Comput. Fluids **74**(0), 44–57 (2013)
2. Molemaker, J., Cohen, J., Patel, S., Noh, J.: Low viscosity flow simulations for animation. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 9–18. Eurographics Association (2008)
3. Costarelli, S., Paz, R., Dalcin, L., Storti, M.: Resolución de las ecuaciones de Navier-Stokes utilizando CUDA. In: Möller, O., Signorelli, J.W., Storti, M.A. (eds.) Mecánica Computacional, vol. XXX, pp. 2979–3008 (2011)
4. Leonard, B.P.: A stable and accurate convective modelling procedure based on quadratic upstream interpolation. Comput. Methods Appl. Mech. Eng. **19**(1), 59–98 (1979)
5. Staniforth, A., Côté, J.: Semi-Lagrangian integration schemes for atmospheric models. A review. Mon. Weather Rev. **119**(9), 2206–2223 (1991)
6. Costarelli, S.D., Storti, M.A., Paz, R.R., Dalcin, L.D., Idelshon, S.A.: Solving 3d viscous incompressible Navier-Stokes equations using CUDA. In: García, C., Printista, M. (eds.) Proceedings of HPCLatAm 2013, Mendoza, Argentina, pp. 69–79 (2013)
7. Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming with CUDA. ACM Queue **6**(2), 40–53 (2008)
8. Farber, R.: CUDA Application Design and Development. Morgan Kaufmann, San Mateo (2011)
9. Costarelli, S., Storti, M., Paz, R., Dalcín, L., Idelsohn, S.: A numerical algorithm for the solution of viscous incompressible flow on GPU's. In: V Latin American Symposium on High Performance Computing HPCLatAm 2012, Buenos Aires, Argentina (2012)
10. Stam, J.: Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 121–128. ACM Press/Addison-Wesley, New York/Reading (1999)
11. Chorin, A., Marsden, J.: A Mathematical Introduction to Fluid Mechanics. Springer, New York (1990)
12. Bleecker, D., Csordas, G.: Basic Partial Differential Equations. Chapman & Hall/CRC Press, London/Boca Raton (1992)
13. Robert, A.: A stable numerical integration scheme for the primitive meteorological equations. Atmos.-Ocean **19**(1), 35–46 (1981)
14. Pudykiewicz, J., Benoit, R., Staniforth, A.: Preliminary results from a partial LRTAP model based on an existing meteorological forecast model. Atmos.-Ocean **23**(3), 267–303 (1985)
15. Bates, J., McDonald, A.: Multiply-upstream, semi-Lagrangian advective schemes: analysis and application to a multi-level primitive equation model. Mon. Weather Rev. **110**(12), 1831–1842 (1982)
16. Kim, B., Liu, Y., Llamas, I., Rossignac, J.: Advections with significantly reduced dissipation and diffusion. IEEE Trans. Vis. Comput. Graph. **13**(1), 135–144 (2007)
17. Selle, A., Fedkiw, R., Kim, B., Liu, Y., Rossignac, J.: An unconditionally stable MacCormack method. J. Sci. Comput. **35**(2), 350–371 (2008)
18. Caiden, R., Fedkiw, R., Anderson, C.: A numerical method for two-phase flow consisting of separate compressible and incompressible regions. J. Comput. Phys. **166**(1), 1–27 (2001)
19. Sussman, M., Fatemi, E.: An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. SIAM J. Sci. Comput. **20**(4), 1165–1191 (1999)
20. Zalesak, S.: Fully multidimensional flux-corrected transport algorithms for fluids. J. Comput. Phys. **31**(3), 335–362 (1979)
21. Enright, D., Losasso, F., Fedkiw, R.: A fast and accurate semi-Lagrangian particle level set method. Comput. Struct. **83**(6), 479–490 (2005)
22. Puckett, E., Almgren, A., Bell, J., Marcus, D., Rider, W.: A high-order projection method for tracking fluid interfaces in variable density incompressible flows. J. Comput. Phys. **130**(2), 269–282 (1997)
23. Enright, D., Fedkiw, R., Ferziger, J., Mitchell, I.: A hybrid particle level set method for improved interface capturing. J. Comput. Phys. **183**(1), 83–116 (2002)
24. Mei, X., Decaudin, P., Hu, B., Zhang, X.: Real-time marker level set on GPU. In: 2008 International Conference on Cyberworlds, pp. 209–216. IEEE Press, New York (2008)
25. LeVeque, R.: High-resolution conservative algorithms for advection in incompressible flow. SIAM J. Numer. Anal. **33**(2), 627–665 (1996)
26. Nvidia. CUDA CUFFT Library (2007)
27. Ghia, U., Ghia, K.N., Shin, C.: High-Re solutions for incompressible flow using the Navier-Stokes equations and multigrid method. J. Comput. Phys. **48**, 387–411 (1982)
28. Achenbach, E.: Experiments on the flow past spheres at very high Reynolds numbers. J. Fluid Mech. **54**(03), 565–575 (1972)
29. Roshko, A.: On the Development of Turbulent Wakes from Vortex Streets. NACA Rep. 1191, 25U. S. Government Printing Office, Washington (1955)
30. Mittal, S., Kumar, V.: Flow-induced vibrations of a light circular cylinder at Reynolds numbers 1000 to 10000. J. Sound Vib. **245**(5), 923–946 (2001)
31. Idelsohn, S.R., Nigro, N.M., Gimenez, J.M., Rossi, R., Marti, J.M., Oñate, E.: A fast and accurate method to solve the incompressible Navier-Stokes equations. Eng. Comput. **30**(2), 197–222 (2013)
32. Mittal, S.: Computation of three-dimensional flows past circular cylinder of low aspect ratio. Phys. Fluids **13**, 177 (2001)

33. Lai, M.C., Peskin, C.S.: An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. J. Comput. Phys. **160**(2), 705–719 (2000)



Santiago D. Costarelli is a Ph.D. student at CIMEC (Research Center on Computational Methods, www.cimec.com.ar). He received his major in Computer Engineering in 2011 from the Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral. His interests are computational fluid mechanics and high performance computing.

Mario A. Storti is a senior scientist at at CIMEC (Research Center on Computational Methods, www.cimec.com.ar, Argentina). He received his degree in Physics from Instituto Balseiro, Universidad Nacional de Cuyo, San Carlos de Bariloche. Argentina, in 1983. In 1990 he received the Ph.D. degree in Chemical Technology at Universidad Nacional del Litoral, Santa Fe, Argentina, where he is currently a professor in postgraduate and undergraduate programs. Recently, he received the 2012 Award for Senior Scientists from the Argentine Association for Computational Mechanics (AMCA).



Rodrigo R. Paz is a senior scientist at LSTC (USA) and CONICET (Argentina). He received his degree in Aeronautical and Mechanical Engineering from Universidad Nacional de Córdoba, Argentina, in 1999. In 2006, after postgraduate studies in France (Université Pierre et Marie Currie—Paris VI and Faculté des Sciences of the Université Jean Monnet in Saint Etienne) he received the Ph.D. degree in Engineering Sciences and Computational Mechanics from Universidad Nacional del Litoral, Argentina, where he is currently a professor in postgraduate and undergraduate programs. His research interests include computational fluid dynamics, domain decomposition methods, parallel computing, finite elements method, weak/strong fluid/structure coupling, absorbing boundary conditions, and multiphase flows. Recently, he received the 2012 Award for Young Scientists (age under 40) from the Argentine Association for Computational Mechanics (AMCA).

Lisandro D. Dalcin is an assistant researcher at the National Council for Scientific and Technological Research of Argentina (Consejo Nacional de Investigaciones Científicas y Tecnológicas) since 2010. He obtained his Ph.D. in Engineering from Universidad Nacional del Litoral (Santa Fe, Argentina) in 2008. His research interests are in scientific computing in distributed memory architectures, medium to large scale numerical simulation, and development of programming tools mixing Python and C/C++/Fortran.

Sergio R. Idelsohn is an ICREA Research Professor and Senior Researcher at CIMNE in Barcelona, Spain. He received his degree in Mechanical Engineering in 1970 at the National University of Rosario, Argentina, and completed in 1974 a Ph.D. degree at the Liege University in Belgium. In 1996 he was promoted to the top degree in the research career in Argentina. On February 1981 he founded CIMEC (Research Center on Computational Methods, www.cimec.com.ar), a research centre specialized in Computational Mechanics. In 1985 he was the founder and first President of the Argentinean Association of Computational Mechanics (AMCA, www.amca.org). In the period 2002–2010 he was the Secretary General of the International Association for Computational Mechanics (IACM, www.iacm.info). Since 2006 he is ICREA Research Professor and Senior Researcher at CIMNE in Barcelona, Spain. His key research lines are the following: Particle Finite Element Methods, Meshless Methods, Phase Change Problems, Reduction Methods, Quasi-Newton Methods. He is regularly invited to deliver Keynote or Plenary Lectures in the main International Conferences in Computational Engineering Science. The most important are in the World Congress on Computational Mechanics (Vienne 2002, Beijing 2004, Venice 2008, Sydney 2010 and Sao Paulo 2012). In addition, to his scientific activity he has developed an intensive task in the transfer of the outcome of his research to the industrial sector in Argentina and now in Spain.