Simulation Modelling Practice and Theory 18 (2010) 217–229

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

# An accelerated algorithm for Navier–Stokes equations

M. Venturin [a,*], R. Bertelle [b], M.R. Russo [b]

[a] Dip. Informatica, Università di Verona, Ca' Vignal 2, Strada Le Grazie, 37134 Verona, Italy
[b] Dip. Matematica Pura ed Applicata, Università degli Studi di Padova, Via Trieste 63, 35131 Padova, Italy

ARTICLE INFO

*Article history:*
Received 24 July 2009
Received in revised form 19 October 2009
Accepted 26 October 2009
Available online 10 November 2009

*Keywords:*
Navier–Stokes equations
Fractional step method
Characteristic-Based-Split (CBS) scheme
Vector extrapolation methods
Minimum Polynomial Extrapolation (MPE) method

ABSTRACT

In this paper, the numerical solution of the Navier–Stokes equations by the Characteristic-Based-Split (CBS) scheme is accelerated with the Minimum Polynomial Extrapolation (MPE) method to obtain the steady state solution for evolution incompressible and compressible problems.

The CBS is essentially a fractional time-stepping algorithm based on an original finite difference velocity-projection scheme where the convective terms are treated using the idea of the Characteristic-Galerkin method. In this work, the semi-implicit version of the CBS with global time-stepping is used for incompressible problems whereas the fully-explicit version is used for compressible flows.

At the other end, the MPE is a vector extrapolation method that transforms the original sequence into another sequence converging to the same limit faster then the original one without the explicit knowledge of the sequence generator.

The developed algorithm, tested on two-dimensional benchmark problems, demonstrates the new computational features arising from the introduction of the extrapolation procedure to the CBS scheme. In particular, the results show a remarkable reduction of the computational cost of the simulation.



## 1. Introduction

The numerical approximation of the steady-state Navier–Stokes equations by fractional step methods typically requires a large amount of computational time which is related to the total number of iterations, the size of the problem and the used algorithm. As a consequence, the convergence rate may become slow and it is interesting to apply acceleration techniques to improve the overall iterative scheme. In this work, the Characteristic-Based-Split (CBS) scheme is combined with the Minimum Polynomial Extrapolation (MPE) method to solve Navier–Stokes problems.

The CBS is an iterative scheme based on a fractional time-stepping algorithm with a finite difference velocity-projection scheme where the convective terms are treated using the idea of the Characteristic-Galerkin method (for details see [1–4]). In particular, in the splitting part, the pressure gradient is omitted from the momentum equations and an intermediate velocity field is estimated by the Characteristic-Galerkin method. Then, the continuity equation is solved using the intermediate vector field values of the previous step obtaining the new pressure or density values. Finally, the velocity field is corrected with the new computed pressure or density term. Then, any advection–diffusion scalar equation, such as those for energy or temperature, is solved using the Characteristic-Galerkin method. In particular, the Characteristic-Galerkin method stabilize advective terms using a Taylor series expansion of the temporal derivative along the characteristic leading to the introduction of consistent stabilization artificial diffusion terms.

* Corresponding author. Tel.: +39 333 2524391.
*E-mail address:* manolo.venturin@univr.it (M. Venturin).

1569-190X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2009.10.008

The CBS scheme has been employed for several compressible and incompressible flow problems [3,4] and it has been extended to several other applications including shallow water flows [5,6], solid dynamics [7,8], thermal and porous medium flow problems [9–13], turbulent-incompressible flows [14] and viscous-elastic flows [15]. Recently, an efficient and accurate explicit free matrix procedure combined with the standard Artificial Compressibility method has been presented in [9,10] for incompressible fluid flow computation. Moreover, an improved version of the CBS code using a local element-size calculation procedure in the streamline direction and a local variable smoothing approach are reported in [16] for the solution of inviscid compressible flow problems.

The MPE method is a vector extrapolation technique used to accelerate the convergence of sequences. An important feature of this method is that it does not require an explicit knowledge of the sequence generator so the acceleration procedure can be directly apply to the solution of linear and nonlinear systems as a black-box routine. The algorithm is based on a weight sum of the sequence generated (by the CBS) where the weights are related to the coefficients of the minimal polynomial of the operator involved.

The MPE method has a wide range of applications in different fields, as [17–19]. In particular, in the PDE context we cite [20–22]. For a deeper understanding and further applications and numerical examples, the interested reader should refer to [23–25] and references therein.

The performance of the developed algorithm that combine the Characteristic-Based-Split scheme with the Minimum Polynomial Extrapolation method has been investigated for solving steady state incompressible and compressible Navier–Stokes equations. The incompressible examples consider both an isothermal forced convection problem and a non-isothermal natural convection problem. The compressible example solves the inviscid Navier–Stokes equations for the NACA0012 airfoil problem. Moreover, the incompressible examples are solved with a semi-implicit version of the CBS scheme with global time-stepping while the compressible problem is solved with a fully-explicit version of the code.

The paper is organized as follows. The first two sections briefly review the CBS and MPE methods providing some implementation details. Then, in the next section, we provide some preliminary comparisons results of the developed algorithm. Finally, some concluding remarks are given.

## 2. The Characteristic-Based-Split (CBS) scheme

In this work three benchmark problems are considered in order to investigate the performance of the developed algorithm. The first two examples refer to the incompressible Navier–Stokes equations (natural and forced convection problems) while the last one refers to a compressible case (NACA0012 airfoil problem). Moreover, isothermal (forced convection) and non-isothermal (natural convection) incompressible Navier–Stokes equations are considered in the first two examples. Hence, we recall the governing equations for the incompressible and compressible examples and then we briefly review the numerical solution strategy used by the CBS scheme. We wish to remark that the objective of this work is not to review the CBS scheme in detail nor to give a complete treatment of the Navier–Stokes equations but, to analyze the behaviour of the CBS accelerated version of the algorithm provided in the following sections.

### 2.1. Governing equations for the incompressible examples

The governing equations for the incompressible Navier–Stokes problems may be written in conservation form as [4,26]

Continuity equation

$$\frac{\partial u_i}{\partial x_i} = 0$$

Momentum equations

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j}(u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right) + g_i$$

Temperature equation

$$\frac{\partial T}{\partial t} + \frac{\partial}{\partial x_i}(u_i T) = \frac{\partial}{\partial x_i}\left(\alpha \frac{\partial T}{\partial x_i}\right)$$

where Einstein's summation convention is used. The unknown variables are the velocity field $u_i$, the pressure $p$ and the temperature $T$. Here, $\mu$ is the kinematic viscosity term of the fluid, $g_i$ takes into account gravitational forces of the system and $\alpha = k/(\rho c_p)$ is the thermal diffusion term which depends on the fluid density $\rho$, on the thermal conductivity $k$ and on the specific heat $c_p$. The previous system is completed with appropriate initial and boundary conditions. For the forced convection problem the gravitational forces are neglected, and the temperature equation is not considered.

In the application of the CBS scheme it is necessary to use the dimensionless form of the equations. For the incompressible examples we can use two non-dimensional scaling strategies depending on the nature of the considered problem (natural or forced convection). For natural convection problems the following scales are employed

$$x_i^\star = \frac{x_i}{L}, \quad u_i^\star = \frac{u_i L}{\alpha}, \quad p^\star = \frac{pL^2}{\alpha^2}, \quad t^\star = \frac{t\alpha}{L^2}$$
$$T^\star = \frac{T - T_L}{T_H - T_L}, \quad \mathrm{Pr} = \frac{\mu}{\alpha}, \quad \mathrm{Ra} = \frac{g\gamma(T_H - T_L)L^3}{\mu\alpha}$$

where the superscript ★ indicates a non-dimensional quantity. Here, Pr and Ra are the Prandtl and Rayleigh numbers, $L$ is the characteristic length, $T_H$ and $T_L$ are, respectively, the higher and the lower temperatures, $g$ is the gravity acceleration and $\gamma$ is the coefficient of thermal expansion. For forced convection problems the following scales are used

$$x_i^\star = \frac{x_i}{L}, \quad u_i^\star = \frac{u_i}{u_\infty}, \quad p^\star = \frac{p}{u_\infty^2}, \quad t^\star = \frac{tu_\infty}{L}$$
$$\mathrm{Pr} = \frac{\mu}{\alpha}, \quad \mathrm{Re} = \frac{u_\infty L}{\mu}$$

where Re is the Reynolds number and $u_\infty$ is the reference velocity.

### 2.2. Governing equations for the compressible example

The governing equations, in the Einstein's notation, for the inviscid compressible Navier–Stokes problem are [4,26]

Continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0$$

Momentum equations

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) + \frac{\partial p}{\partial x_i} = 0$$

Energy equation

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial}{\partial x_i}(\rho E u_i) + \frac{\partial}{\partial x_i}(u_i p) = 0$$

where the unknowns are the fluid density $\rho$, the fluid velocity field $u_i$, the total energy $E$ and the pressure $p$. To close the system the following constitutive laws are used

$$p = \rho RT, \quad e = c_v T, \quad E = e + \frac{u_i u_i}{2}, \quad c^2 = \gamma RT, \quad \gamma = \frac{c_p}{c_v}$$

where $R$ is a constant, $T$ is the temperature, $e$ is the internal energy per unit mass, $c$ is the speed of sound and $\gamma$ is the ratio of specific heats at constant pressure $c_p$ and constant volume $c_v$. The problem is completed by the specification of initial and boundary conditions.

The main scaling factors used in the compressible example are

$$x_i^\star = \frac{x_i}{L}, \quad u_i^\star = \frac{u_i}{u_\infty}, \quad \rho^\star = \frac{\rho}{\rho_\infty}$$

where the subscript $\infty$ represents a reference free stream value and $L$ is a reference length.

### 2.3. Solution procedure

The CBS scheme is a well known time iterative procedure for the solution of the Navier–Stokes equations and it has been discussed in many earlier publications. In particular, the CBS algorithm used in our simulations is the same reported in [1–3] and consists of a fractional procedure for the solution of the momentum and the continuity equations followed by a procedure based on the Characteristic-Galerkin method for the solution of any advection equation such as those for energy or temperature.

The basic temporal steps of the CBS scheme can be summarized as follows:

- solve the intermediate momentum equation obtained from the momentum equation without the pressure gradient term;
- solve the modified continuity equation for the pressure or density variable;
- correct the intermediate velocity field using the new values of the pressure or density;
- solve any advection equation such as those for temperature or energy.

By removing the pressure term from the momentum equation the fractional step method introduces a first-order error into the momentum equation [27]. This allows to circumvent the Ladyshenskaya–Bubuska–Brezzi (LBB) condition when incompressibility, or near incompressibility, is encountered, allowing equal order interpolations for both velocity and pressure fields and enhancing pressure stability of the CBS scheme. As an alternative, it is possible to consider pressure gradient into the momentum equations as a source term, but as discussed in [27] this approach is not recommended.

The main advantage of this scheme is the equal order interpolation for pressure or density and velocity which reduces the complexity of the program and the CPU time. Moreover, we use linear elements with direct integration in order to further reduce the computational requirement.

In the CBS scheme, the semi-implicit version with global time-stepping is used for incompressible problems where the pressure is treated implicitly. Instead, the fully-explicit version is used for the compressible flow example. For the simulation details we refer to [12,28] for the incompressible Navier–Stokes equations while to [16] for the compressible case.

## 3. The extrapolation algorithm

The CBS is an iterative algorithm which provides an approximation of the solution throughout the construction of a vector sequence which, under suitable hypothesis, converges to the true solution. In some real applications cases, the complexity of the problem needs a long number of iteration steps before arising to an acceptable approximation of the solution. As a consequence, waste computational time is needed, and this may not be acceptable in the development of engineering projects where a low time to market of new products is one of the most important targets to be satisfied. Hence, in all such cases it is of strong interest to investigate the behaviour of an acceleration technique in order to speed up the rate of convergence and, thus, allowing lower simulation times.

### 3.1. The MPE method

For the purpose of this work, although for a more mathematical reviews we refer to [23], an acceleration technique is a map $\mathscr{T}$ that transforms a given convergent sequence $\{x_n\}$ into another sequence $\{y_n\} = \mathscr{T}(\{x_n\})$ which converges to the same limit of the input sequence. Clearly, the interest relies on those maps $\mathscr{T}$ which accelerate as much as possible the rate of convergence of $\{y_n\}$. The choice of one of such map is a difficult task. Moreover it is not clear, a priori, if the application of a specific map to a given problem provides reliable results. As consequence, besides the theoretical study of new acceleration maps [29], it is also of interest to investigate the behaviour of some well known acceleration techniques applied to a specific problem. In this work, the effectiveness of the acceleration technique known as Minimal Polynomial Extrapolation (MPE) is investigated when applied to the fluid simulation environment. For some other applications, see [17,20,30].

To introduce the MPE, let $\{x_n\}, n \in \mathbb{N}$, be a sequence of vectors in $\mathbb{R}^N$ with limit $x$ and $k$ some positive, fixed, integer. Setting $u_n = x_{n+1} - x_n, n \in \mathbb{N}$, the sequence $\{y_n\}$ is obtained as a weight sum of the form

$$y_n = \sum_{j=0}^{k} \gamma_j x_{n+j}, \quad \text{with } \gamma_j = \frac{\zeta_j}{\sum_{i=0}^{k} \zeta_i}, \quad j = 0, 1, \ldots, k$$

and $\zeta = [\zeta_0, \zeta_1, \ldots, \zeta_{k-1}]$ solution of the linear system

$$[u_n u_{n+1}, \ldots, u_{n+k-1}]\zeta = -u_{n+k},$$

and $\xi_k = 1$.

The matrix of this linear system is $N \times k$ where, in this work, $N$ is the number of mesh nodes and $k$ is the number of collected vectors that are used to extrapolate a new value for $y_n$, and as a consequence we have $N \gg k$. The system is solved in the sense of the minimum least squares.

### 3.2. The extrapolation technique

In this section we show how the acceleration technique has been implemented in the CBS algorithm. We start by recalling that the Navier–Stokes equations is a system of partial differential equations expressed using a set of independent variables denoted by $\mathscr{U}$. For example for problem **P1** and **P2**, see Section 4, we have, respectively, $\mathscr{U} = \{T, u_1, u_2, p\}$ and $\mathscr{U} = \{u_1, u_2, p\}$. The discretization of this continuous problem using the CBS scheme produces a sequence of sets $\mathbf{U}^{(n)}, n = 0, 1, 2, \ldots$ of approximations of the variables in $\mathscr{U}$. Referring again to the problem **P1**, we have $\mathbf{U}^{(n)} = \{\mathbf{T}^{(n)}, \mathbf{u}_1^{(n)}, \mathbf{u}_2^{(n)}, \mathbf{p}^{(n)}\}$ where, for example, $\mathbf{T}^{(n)}$ is the vector of the temperature nodal values at the step $n$ computed by the CBS algorithm. Moreover, we assume that the sequence of sets $\mathbf{U}^{(n)}$ converges, in some norm, to the stationary solution $\mathbf{U}$, i.e.,

$$\lim_{n \to +\infty} \mathbf{U}^{(n)} = \mathbf{U} = \{\mathbf{T}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{p}\}.$$

The purpose of this work is to accelerate the previous sequence $\mathbf{U}^{(n)}$ with $n \in \mathbb{N}$. Since, we have introduced only the acceleration of a vector sequence, we define an extrapolation of a set sequence as the one obtained applying the extrapolation procedure to each component sequence of the set. That is, we have to collect an appropriate number $N_e$ of sets from the sequence, denoted by $\mathbf{U}^{(n)}$ with $n = \bar{n} + 1, \ldots, \bar{n} + N_e$, and then to perform the extrapolation step using the MPE method on each sequence (variable).

There are two main problems in this operation. First, to pick a correct value for $N_e$ and, second, when to start to collect these $N_e$ consecutive sets, i.e., how to choose $\bar{n}$. Since there are no general guidelines for evolution problems that converge to stationary solution, we adopt the following strategies.

1. We select an appropriate subset $\widetilde{\mathbf{U}}$ of $\mathbf{U}$ which represents the variables used to check where extrapolation procedure should be done. We remark that, regardless the choice of $\widetilde{\mathbf{U}}$, all the solution variables are extrapolated.
2. We skip the first $N_s$ sets of the sequence, that is $\mathbf{U}^{(n)}, n = 1, \ldots, N_s$, since they are, reasonable, the most far away from the stationary solution and, hence, it make not sense to perform some extrapolation step.
3. We collect $N_e$ consecutive sets characterized by the fact that the maximum error for each variables in $\widetilde{\mathbf{U}}$ between two consecutive iterations occurs at the same mesh nodes. This heuristic like to mimic the behaviour of iterative solutions near stationary points where the shape of the solutions change small between two consecutive iterations. As a consequence the error between consecutive iterations are proportional and hence it is reasonably that it appears at the same nodes of the mesh. This way to check the extrapolation point is very less, memory and time, consuming.
4. From the previous collected sets we perform extrapolation only on the last $N_{ef}$ sets of $N_e$. This has two reasons. First, we need to maintain the number of extrapolation vectors and thus the CPU time low in solving of the MPE algorithm. Second, we need to check assessment of the extrapolation criteria inside a more long sequence in order to avoid to collect less significant vectors for the extrapolation.
5. Finally, since the extrapolated solution, typically, is not a feasible solution for the evolution problem (i.e. it does not satisfy the equations), we give some iterations, say $N_r$, to relax towards a feasible one before looking back again to the sequence $\mathbf{U}^{(n)}$ for finding another starting point $\bar{n}$ before collecting further $N_e$ sets. To avoid breakdown of the overall accelerated algorithm, it is necessary that the underlying CBS scheme is robust and stable.

It follows that there are four extrapolation parameters $(N_{ef}, N_s, N_r, N_{eb})$, where $N_{eb} = N_e - N_{ef}$, to set up which give much flexibility to the algorithm but, as a drawback, require a training session in order to find a good choice. This time is not taken into account.

For all the simulations we have adopted the stopping criteria based on the maximum absolute error [4,31],

$$e = \frac{\|\mathbf{U}^{(i)} - \mathbf{U}^{(j)}\|_\infty}{\Delta t},$$

where $\Delta t$ is the time step of the CBS algorithm. The error is computed between the iterations $i$ and $j$, for all the scaled solution variables in $\mathbf{U}$, i.e. $\max_{\mathbf{v}\in\mathbf{U}}\|\mathbf{v}\|_\infty$. In this work the convergence criteria is checked every 10 iterations.

All these ideas are shown in the algorithm reported in Fig. 1. In this code we denoted by $\epsilon$ the stopping tolerance, $N_{maxiter}$ the maximum number of iterations. Looking closer to the code, $n_r$ is a local variables that are used to perform the previous step 5. Furthermore, in this implementation we restart the extrapolation procedure if the extrapolation criteria breakdown before $N_e$ sets are collected.

## 4. Numerical results

The performance of the algorithm is investigated using the following benchmark problems

- incompressible Navier–Stokes equations:
  (**P**1): non-isothermal incompressible problem (natural convection);
  (**P**2): isothermal incompressible problem (forced convection);
- compressible inviscid Navier–Stokes equations:
  (**P**3): NACA0012 airfoil problem.

All simulations have been run on a Linux system with an Intel Pentium 4 CPU 2.00 GHz with a Cache size of 512 KB and with 1 GB of RAM. We have used optimized ATLAS library and the GFortran compiler. The post-processing has been done in Matlab. The CPU times are measured in seconds.

The simulation parameters are reported in Tables 1, 3 and 5 and have the following structure. The first row of each table refers to the results of the simulation obtained without acceleration, i.e., with a standard implementation of the CBS scheme. We refer to this simulation as `sim0`. The next six rows of each table contain the results of the simulations where an acceleration procedure is applied. The corresponding variables $\widetilde{\mathbf{U}}$ used to verify the collecting check are reported in the second column and the extrapolation parameters $(N_{ef}, N_s, N_r, N_{eb})$ are given in the third column. The six rows contain two groups of simulations which have the same extrapolation criteria but different extrapolation parameters. In the first group belong the simulations `sim1`, `sim2` and `sim3` while in the second group belong the simulations `sim4`, `sim5`, and `sim6`. Moreover, each simulation in the first group have the same counterpart in the second group (in the same order) which differs from the extrapolation criteria but has the same extrapolation parameters. For example, simulations `sim1` and `sim4` differ only from the extrapolation criteria but have the same extrapolation parameters. The variables that appear in the extrapolation criteria column of Tables 1, 3 and 5 and that refer to the first simulation group are also the independent solution variables $\mathbf{U}$ computed by the CBS scheme for the considered problem.

**Acceleration procedure**

**Initializations**
choose $\widetilde{\mathbf{U}}$
set $N_{ef},\ N_s,\ N_r,\ N_{eb},\ \epsilon$
guess $\mathbf{U}^{(0)}$,
$k \leftarrow 0$
**repeat**
update solution $\mathbf{U}^{(k+1)}$ from $\mathbf{U}^{(k)}$ using CBS scheme
compute the error $e$
$k \leftarrow k+1$
**until** $k \leq N_s$)

$n_r \leftarrow N_r + 1$
$n \leftarrow 0$
**while** (e $\geq \epsilon$ & $k \leq N_{maxiter}$)
update solution $\mathbf{U}^{(k+1)}$ from $\mathbf{U}^{(k)}$ using CBS scheme
**if** (the extrapolation criteria is fulfilled for $\widetilde{\mathbf{U}}^{(k+1)}$ & $n_r > N_r$ )
$n \leftarrow n+1$
$\mathbf{V}_n \leftarrow \mathbf{U}^{(k+1)}$
**if** $n = N_e$
extrapolate $\mathbf{V}$ using $\mathbf{V}_{N_{eb}}, \ldots, \mathbf{V}_{N_e}$
$\mathbf{U}^{(k+1)} \leftarrow \mathbf{V}$
$n \leftarrow 0$
$n_r \leftarrow 1$
**end**
**else**
**if** the extrapolation criteria was fulfilled at the previous iteration
$n \leftarrow 0$
$n_r \leftarrow 1$
**end**
**end**
compute the error $e$
$k \leftarrow k+1$
$n_r \leftarrow n_r + 1$
**end**

**Fig. 1.** The acceleraed CBS algorithm.

**Table 1**
Extrapolation parameters used in the natural convection problem.

| Simulation | Description | |
|---|---|---|
| | Extrap. criteria | Parameters |
| sim0 | No extrapolation | |
| sim1 | ($T$, $u_1$, $u_2$, $p$) | (100, 0, 100, 100) |
| sim2 | ($T$, $u_1$, $u_2$, $p$) | ( 50, 0, 100, 150) |
| sim3 | ($T$, $u_1$, $u_2$, $p$) | (100, 0, 200, 100) |
| sim4 | ($T$, $u_1$, $u_2$) | (100, 0, 100, 100) |
| sim5 | ($T$, $u_1$, $u_2$) | ( 50, 0, 100, 150) |
| sim6 | ($T$, $u_1$, $u_2$) | (100, 0, 200, 100) |

The total computational times are reported in Tables 2, Table 4 and 6 at the different stopping tolerances $\epsilon$ equal to $10^{-6}, 10^{-7}$ and $10^{-8}$. Near each simulation time we have reported a normalization time in percent rounded at the unit which is refered to the time of the simulation without extrapolation sim0 at the same stopping tolerance.

Figs. 4, 7 and 9 provide the relative error which is computed by the formula

$$e_v = \frac{1}{\Delta t}\frac{\|v^{(n)} - v^*\|_2}{\|v^*\|_2}$$

where $v^{(n)}$ is the solution at the $n$-th iterations and $v^*$ is a reference solution computed with a high accuracy tolerance ($10^{-10}$) using the standard CBS code without acceleration. We use this formula since the algorithm procedure is time dependent, and

**Table 2**
Simulation times obtained from the natural convection problem.

| Simulation | Stopping tolerance | | | | | |
|---|---|---|---|---|---|---|
| | 1e−06 | | 1e−07 | | 1e−08 | |
| sim0 | 4061 | (100%) | 4199 | (100%) | 4358 | (100%) |
| sim1 | 2025 | (50%) | 2112 | (50%) | 2244 | (51%) |
| sim2 | 1728 | (43%) | 1817 | (43%) | 1859 | (43%) |
| sim3 | 1708 | (42%) | 1780 | (42%) | 1861 | (43%) |
| sim4 | 1758 | (43%) | 1836 | (44%) | 1897 | (44%) |
| sim5 | 1786 | (44%) | 1888 | (45%) | 2089 | (48%) |
| sim6 | 2266 | (56%) | 2343 | (56%) | 2572 | (59%) |

**Table 3**
Extrapolation parameters used in the forced convection problem.

| Simulation | Description | |
|---|---|---|
| | Extrap. criteria | Parameters |
| sim0 | No extrapolation | |
| sim1 | $(u_1, u_2, p)$ | (100, 10000, 1000, 100) |
| sim2 | $(u_1, u_2, p)$ | ( 50, 10000, 500, 50) |
| sim3 | $(u_1, u_2, p)$ | ( 25, 5000, 250, 25) |
| sim4 | $(u_1, u_2)$ | (100, 10000, 1000, 100) |
| sim5 | $(u_1, u_2)$ | ( 50, 10000, 500, 50) |
| sim6 | $(u_1, u_2)$ | ( 25, 5000, 250, 25) |

**Table 4**
Simulation times obtained from the forced convection problem.

| Simulation | Stopping tolerance | | | | | |
|---|---|---|---|---|---|---|
| | 1e−06 | | 1e−07 | | 1e−08 | |
| sim0 | 1302 | (100%) | 1785 | (100%) | 2189 | (100%) |
| sim1 | 842 | (65%) | 1051 | (59%) | 1329 | (61%) |
| sim2 | 853 | (66%) | 1061 | (59%) | 1312 | (60%) |
| sim3 | 633 | (49%) | 1076 | (60%) | 1478 | (68%) |
| sim4 | 837 | (64%) | 904 | (51%) | 1117 | (51%) |
| sim5 | 839 | (65%) | 1188 | (67%) | 1227 | (56%) |
| sim6 | 675 | (52%) | 741 | (42%) | 1587 | (73%) |

**Table 5**
Extrapolation parameters used in the NACA0012 problem.

| Simulation | Description | |
|---|---|---|
| | Extrap. criteria | Parameters |
| sim0 | No extrapolation | |
| sim1 | $(E, u_1, u_2, \rho)$ | (100, 20000, 15000, 0) |
| sim2 | $(E, u_1, u_2, \rho)$ | ( 50, 20000, 15000, 50) |
| sim3 | $(E, u_1, u_2, \rho)$ | (100, 10000, 15000,0) |
| sim4 | $(E, u_1, u_2)$ | (100, 20000, 15000,0) |
| sim5 | $(E, u_1, u_2)$ | ( 50, 20000, 15000, 50) |
| sim6 | $(E, u_1, u_2)$ | (100, 10000, 15000, 0) |

the difference of two solutions is proportional to the time step $\Delta t$ and to the residual of the solution itself. In particular, in the figures we compare the relative error results for the solution obtained without extrapolation and the best extrapolated solution obtained from the simulations at each ten iterations. Moreover, in Figs. 4, 7 and 9 we have enhanced with vertical dashed lines where the extrapolation is performed.

The optimal choice of the extrapolation vectors is still an open question since depends on both the type and the parameters of the considered problem. Hence, there are no general rules to perform this choice and, as a consequence, the experiments are done using different values for the extrapolation parameters.

As a general remark, we can observe from the computation results that it is more difficult to extrapolate pressure or density instead of the velocity field or temperature. Indeed, for example in Fig. 4 we can note that the third extrapolation step

performs well for all the variables except the pressure, i.e. the solution for the pressure after the extrapolation is worst than the corresponding one before the extrapolation. This phenomenon still remains in the other Figs. 7 and 9.

### 4.1. Buoyancy-driven convection problem (**P1** problem)

This benchmark problem shows the case of natural convection where the Navier–Stokes equations are coupled with the temperature equation. In this problem a local temperature difference induces a local density difference within the fluid and this produce the fluid motion.

A schematic diagram of the geometry of the problem, as well as the applied boundary conditions, are shown in the left of Fig. 2. The two vertical walls of the square domain $\Omega = (0,1) \times (0,1)$ are kept at the constant temperatures $T = 1$ and $T = 0$, respectively for the left and the right one; the other two horizontal walls are adiabatic, i.e., $\partial T/\partial \mathbf{n} = 0$ where $\mathbf{n}$ denotes the outward normal unit vector. Moreover, no-slip velocity conditions are applied on all the walls and the pressure in the left bottom corner is fixed at zero for each time $t$. The Rayleigh and the Prandtl numbers are, respectively, $Ra = 10^4$ and $Pr = 0.71$.

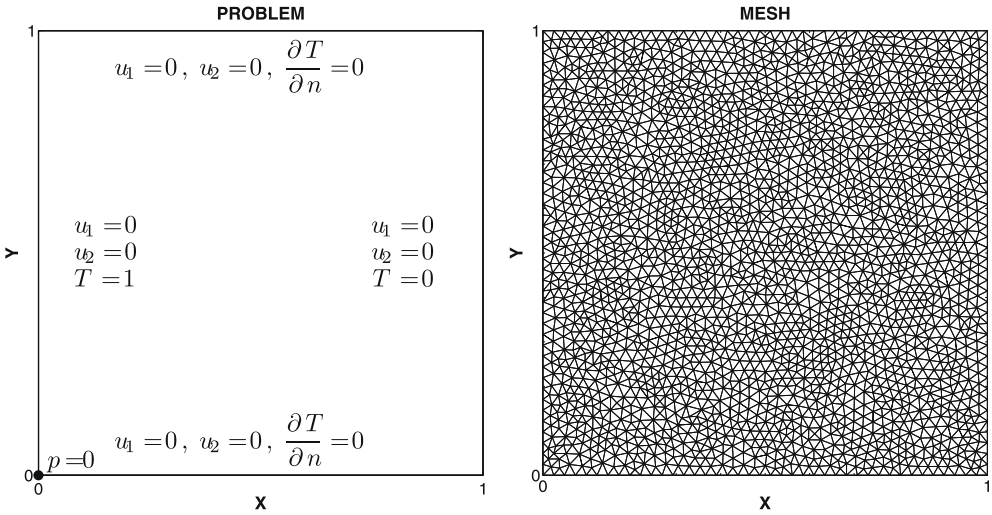


**Fig. 2.** Natural convection problem: problem definition (left); mesh (right).

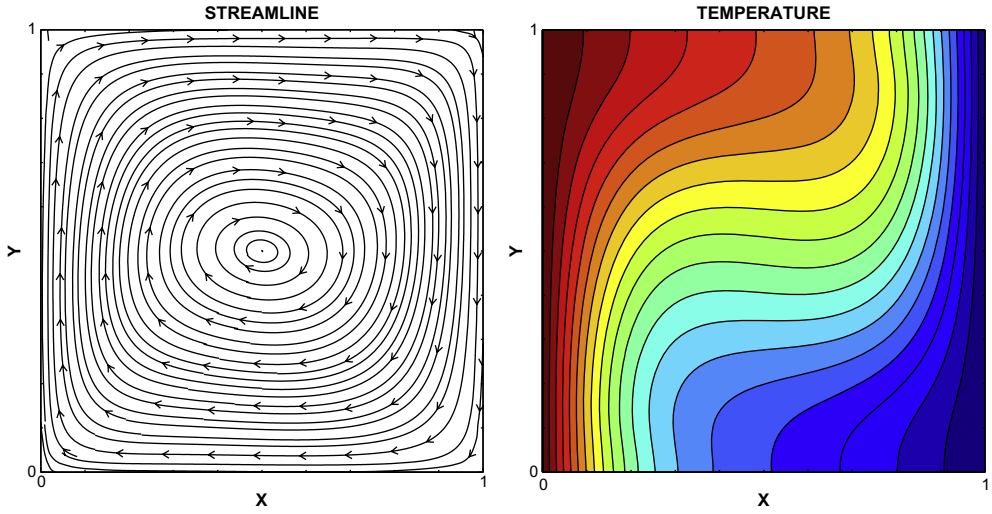


**Fig. 3.** Natural convection problem: streamline (left); temperature (right).

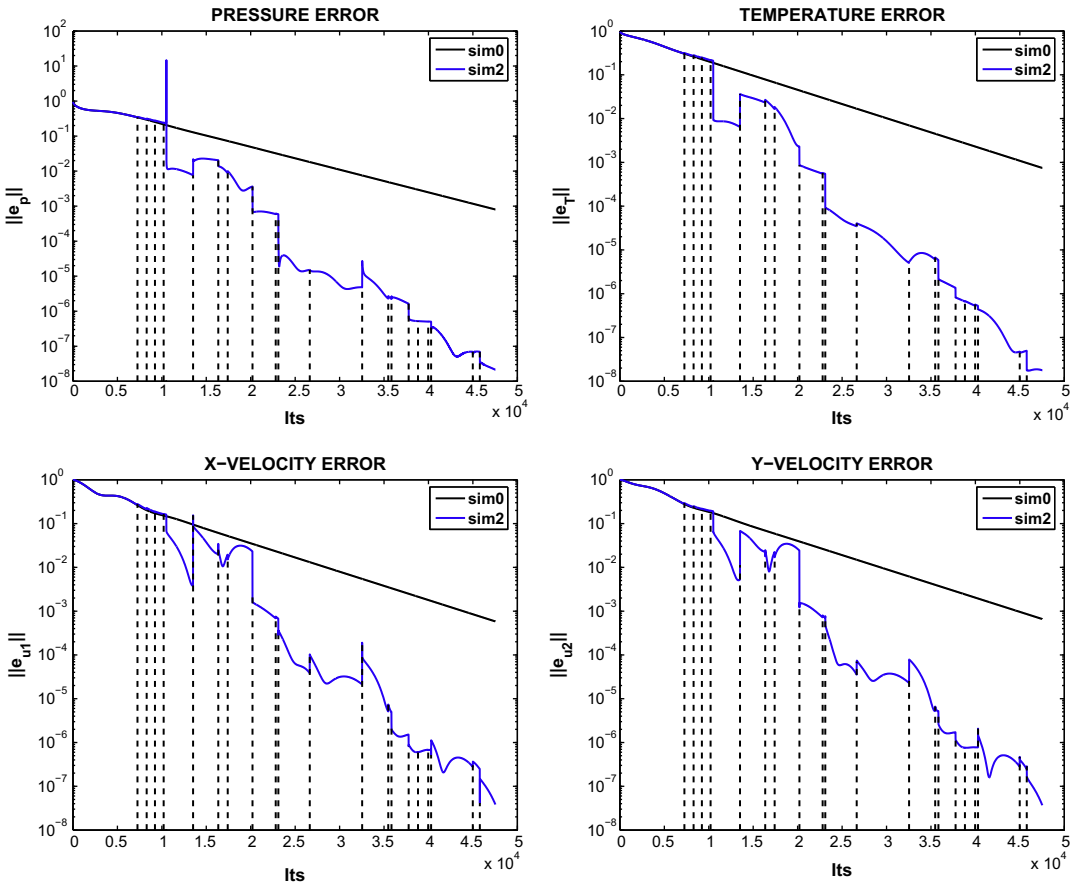


**Fig. 4.** Errors from `sim2` for the natural convection problem using $\epsilon = 10^{-8}$: pressure (top left); temperature (top right); $x$-velocity (bottom left); $y$-velocity (bottom right).

The domain is discretized using an unstructured mesh, shown on the right of Fig. 2, with 2375 nodes and 4584 triangles. The CBS code is used in its semi-implicit form with a global time-stepping $\Delta t = 10^{-5}$ and zero initial conditions for all the variables.

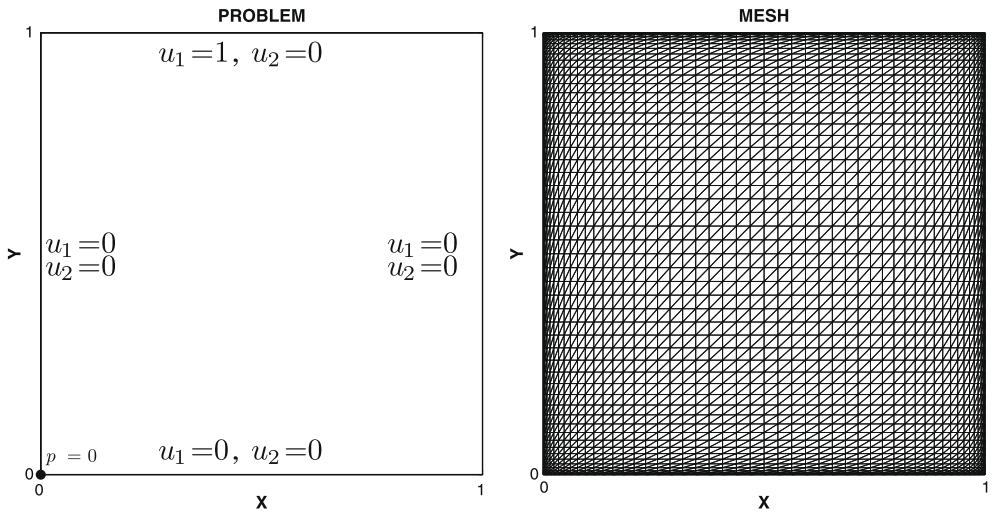


**Fig. 5.** Forced convection problem: problem definition (left); mesh (right).

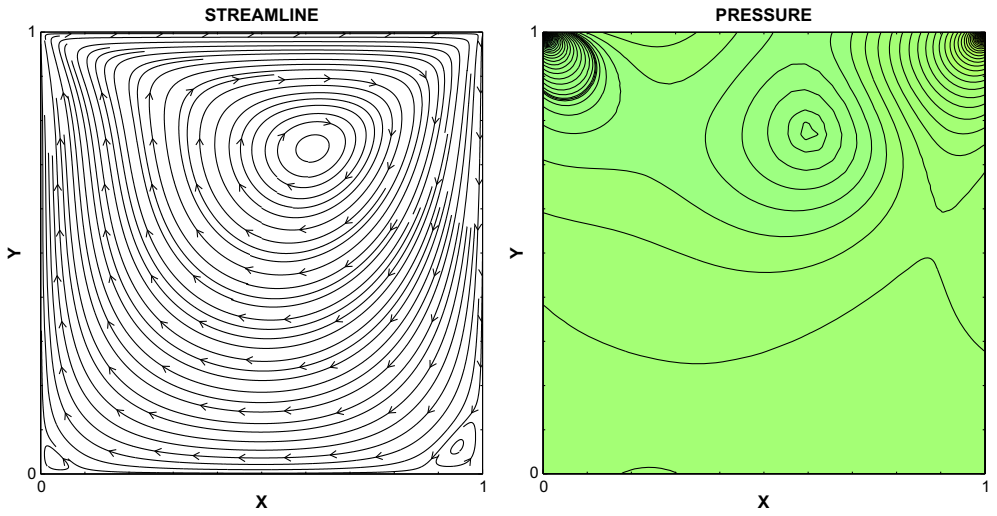


**Fig. 6.** Forced convection problem: streamline (left); pressure (right).

For example in Fig. 3 we show the streamline and temperature solution of the problem. For further details, see [12,28,31] and the references therein.

The extrapolation parameters are shown in Table 1 and the corresponding CPU times in Table 2. For both Tables 2 and Fig. 4 appear that the extrapolation procedure allows a reduction of 55% of the original CPU time.

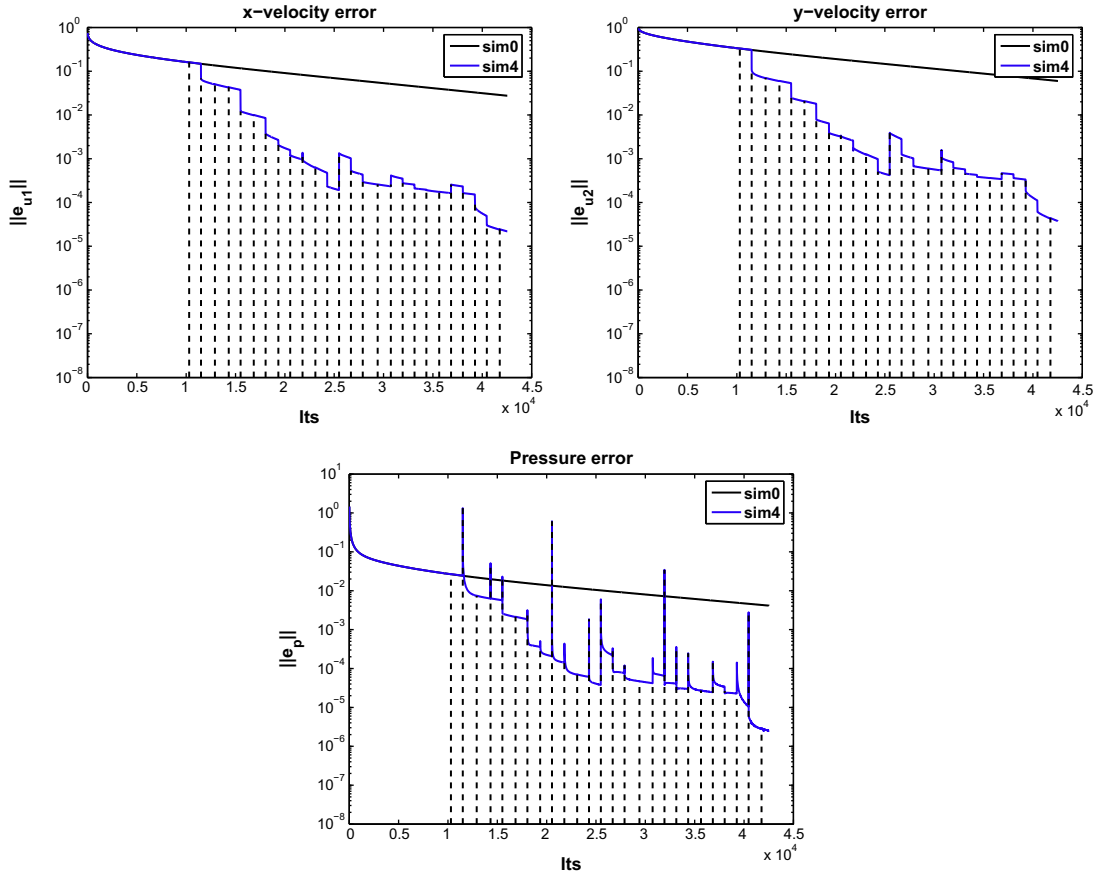


**Fig. 7.** Errors from `sim4` for the forced convection problem using $\epsilon = 10^{-8}$: $x$-velocity (top left); $y$-velocity (top right); pressure (bottom).

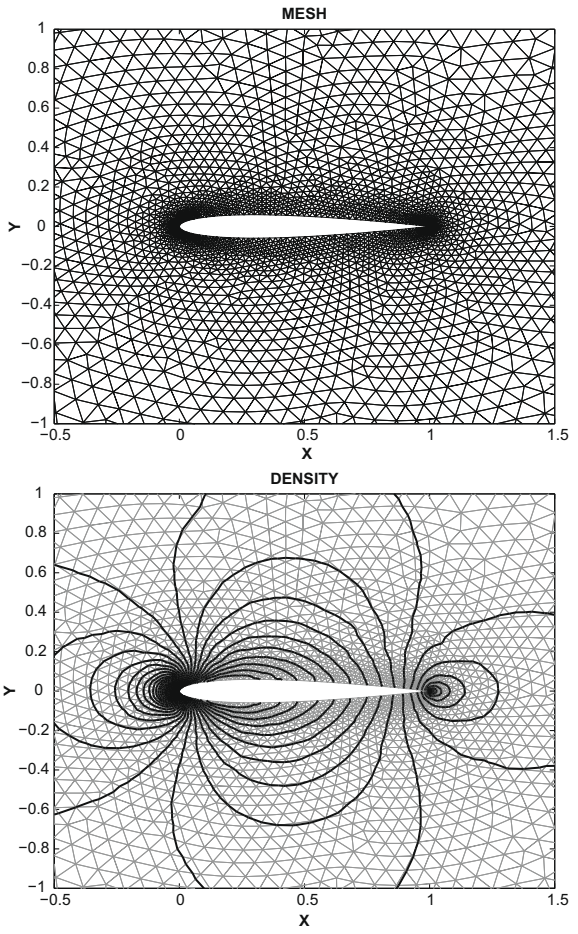


**Fig. 8.** NACA0012 problem: computational domain (top); density contours (bottom).

### 4.2. Lid-driven cavity problem (***P2*** problem)

This problem considers a square cavity domain $\Omega = (0,1) \times (0,1)$ with no-slip velocity conditions on the bottom and on the side walls, and with the top-lid moving at a given horizontal velocity $u_x = 1, u_y = 0$ as shown on the left of Fig. 5.

Fig. 6 shows the pressure, horizontal and vertical velocity contours (streamline) at the Reynolds number of 100.

The computational domain has been discretized using a structured triangular mesh with 2601 nodes and 5000 elements refined near the walls of the geometry as reported on the right of Fig. 5. The global time stepping of the semi-implicit version of the CBS scheme is equal to $10^{-4}$.

In this case the extrapolation procedure performs less good with respect to the previous example since we have a reduction of the computational time of 40%. This is probably due to the difficult in the extrapolation of the pressure as shown in Fig. 7.

### 4.3. NACA0012 airfoil problem (***P3*** problem)

In this section the performance of the CBS algorithm with the MPE extrapolation technique has been investigated on an inviscid compressible flow over a NACA0012 airfoil test problem at Mach number 0.25.

The used algorithm is the same reported in [1] in its fully-explicit version with time step $\Delta t = 10^{-4}$.

The computational domain is circular with a diameter of $25L$ centered in (0,0), where $L$ is the chord length of the NACA0012 airfoil. The leading edge of the airfoil is located at (0,0).

The left part of the circle is used to impose inlet boundary conditions while the right part of the circle is used to impose exit boundary conditions. The mesh shown in the left of Fig. 8 is a close detail of the unstructured mesh with 7351 elements and 3753 nodes. Furthermore, on the right of Fig. 8 we have reported only the density contours as the solution variable.

The results presented in Table 6 and Fig. 8 show that the extrapolation procedure allows a reduction of 45% of the original CPU time. For these simulations we observed that the two extrapolation criteria produce the same results.

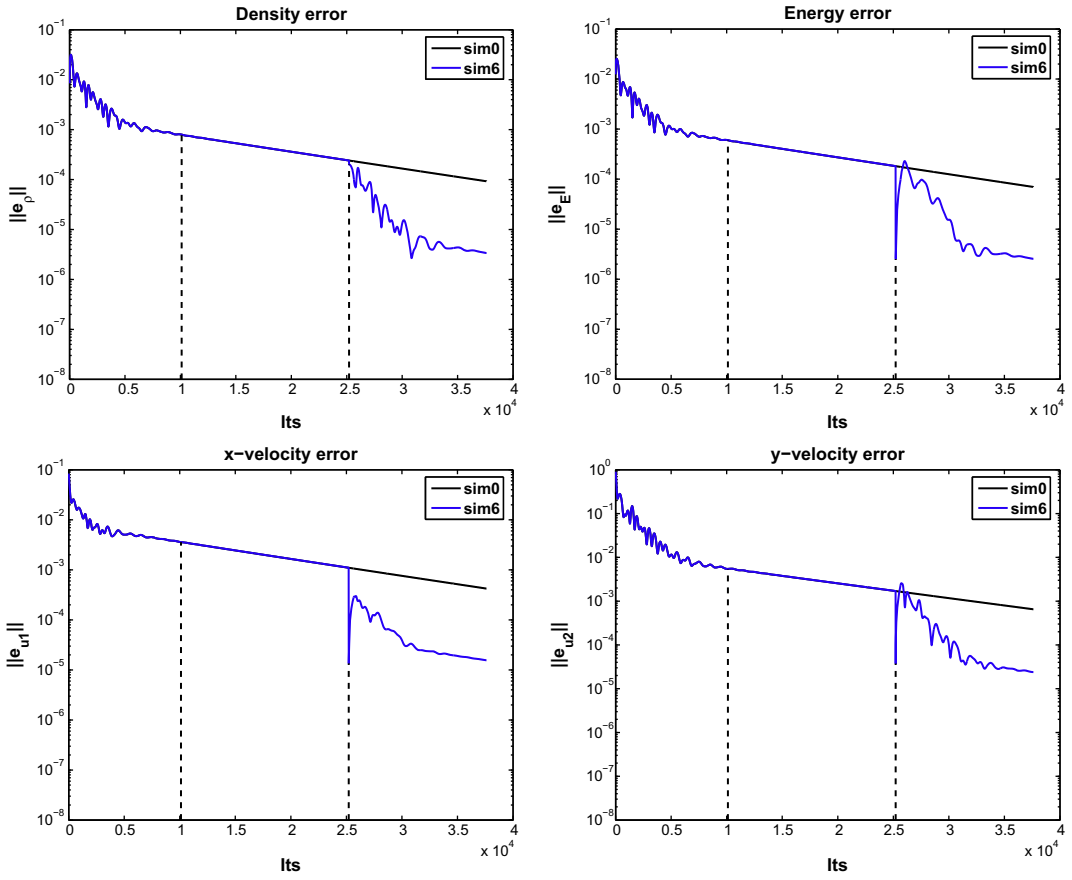


**Fig. 9.** Errors from `sim6` for the NACA0012 problem using $\epsilon = 10^{-8}$: density (top left); energy (top right); *x*-velocity (bottom left); *y*-velocity (bottom right).

**Table 6**
Simulation times obtained from the NACA0012 problem.

| Simulation | Stopping tolerance | | | | | |
|---|---|---|---|---|---|---|
| | 1e−06 | | 1e−07 | | 1e−08 | |
| sim0 | 282 | (100%) | 872 | (100%) | 1530 | (100%) |
| sim1 | 280 | (99%) | 903 | (104%) | 971 | (63%) |
| sim2 | 280 | (99%) | 902 | (103%) | 970 | (63%) |
| sim3 | 284 | (100%) | 711 | (81%) | 808 | (53%) |
| sim4 | 280 | (99%) | 913 | (105%) | 983 | (64%) |
| sim5 | 280 | (99%) | 912 | (105%) | 981 | (64%) |
| sim6 | 283 | (100%) | 716 | (82%) | 816 | (53%) |

## 5. Conclusions

In this paper we have applied the MPE extrapolation method to the CBS algorithm obtaining a new algorithm. The resulting algorithm is tested on three benchmark problems coming from Navier–Stokes equations. The preliminary numerical results demonstrate that the new approach is an attractive field of interest for the applications of the extrapolation techniques since the presented results show a remarkable reduction of the computational cost of the simulations.

## Acknowledgements

Authors are pleasure to acknowledge professors M. Redivo Zaglia, S. De Marchi and M. Mo-ran-di Cec-chi for their constant suggestions and remarks. The authors wish also to thank anonymous reviewers for their remarks that improved the overall quality of the paper.

## References


[1] O.C. Zienkiewicz, R. Codina, A general algorithm for compressible and incompressible flow. I. The split, characteristic-based scheme, Int. J. Numer. Meth. Fluids 20 (8-9) (1995) 869–885.
[2] O.C. Zienkiewicz, P. Nithiarasu, R. Codina, M. Vázquez, P. Ortiz, The characteristic-based-split procedure: an efficient and accurate algorithm for fluid problems, Int. J. Numer. Meth. Fluids 31 (1) (1999) 359–392.
[3] P. Nithiarasu, R. Codina, O.C. Zienkiewicz, The characteristic-based split (CBS) scheme – a unified approach to fluid dynamics, Int. J. Numer. Meth. Eng. 66 (10) (2006) 1514–1546.
[4] O.C. Zienkiewicz, R.L. Taylor, P. Nithiarasu, The Finite Element Method for Fluid Dynamics, sixth ed., Elsevier Butterworth Heinemann, London, 2005. vol. 3.
[5] P. Ortiz, O.C. Zienkiewicz, J. Szmelter, Hydrodynamics and transport in estuaries and rivers by the CBS finite element method, Int. J. Numer. Meth. Eng. 66 (10) (2006) 1569–1586.
[6] M. Morandi-Cecchi, M. Venturin, Characteristic-based split (CBS) algorithm finite element modelling for shallow waters in the Venice lagoon, Int. J. Numer. Meth. Eng. 66 (10) (2006) 1641–1657.
[7] J. Rojek, E. Oñate, R.L. Taylor, CBS-based stabilization in explicit solid dynamics, Int. J. Numer. Meth. Eng. 66 (10) (2006) 1547–1568.
[8] P. Nithiarasu, A matrix free fractional step method for static and dynamic incompressible solid mechanics, Int. J. Comput. Meth. Eng. Sci Mech. 7 (2006) 369–380.
[9] P. Nithiarasu, An efficient artificial compressibility (AC) scheme based on the characteristic based split (CBS) method for incompressible flows, Int. J. Numer. Meth. Fluids 56 (13) (2003) 1815–1845.
[10] P. Nithiarasu, J.S. Mathur, N.P. Weatherill, K. Morgan, Three-dimensional incompressible flow calculations using the characteristic based split (CBS) scheme, Int. J. Numer. Meth. Fluids 44 (11) (2004) 1207–1229.
[11] P. Nithiarasu, C.-B. Liu, Steady and unsteady flow calculations in a double driven cavity using the artificial compressibility (AC)-based characteristic based split (CBS) scheme, Int. J. Numer. Meth. Fluids 63 (3) (2005) 380–397.
[12] N. Massarotti, P. Nithiarasu, O.C. Zienkiewicz, Characteristic-based-split (CBS) algorithm for incompressible flow problems with heat transfer, Int. J. Numer. Meth. Heat Fluid Flow 8 (8) (1998) 969–990.
[13] N. Massarotti, P. Nithiarasu, O.C. Zienkiewicz, Natural convection in a porous medium–fluid interface problems a finite element analysis by using the characteristic based split (CBS) algorithm, Int. J. Numer. Meth. Heat Fluid Flow 11 (2001) 473–490.
[14] P. Nithiarasu, C.-B. Liu, An artificial compressibility based characteristic based split (CBS) scheme for steady and unsteady turbulent incompressible flows, Comput. Methods Appl. Mech. Eng. 195 (23–24) (2006) 2961–2982.
[15] P. Nithiarasu, A fully explicit characteristic based split (CBS) scheme for viscoelastic flow calculations, Int. J. Numer. Meth. Eng. 60 (2004) 949–978.
[16] P. Nithiarasu, C.G. Thomas, Influences of element size and variable smoothing on inviscid compressible flow solution, Int. J. Numer. Meth. Heat Fluid Flow 15 (5) (2005) 420–428.
[17] A. Sidi, Vector extrapolation methods with application to solution of large system of equations and to PageRank computations, Comput. Math. Appl. 56 (2008) 1–24.
[18] K. Jbilou, H. Sadok, Vector extrapolation methods. Applications and numerical comparison, J. Comput. Appl. Math. 122 (2000) 149–165.
[19] N. Rajeevan, Vector-extrapolated fast maximum likelihood estimation algorithms for emission tomography, IEEE Trans. Med. Imaging 11 (1) (1992) 9–20.
[20] R. Bertelle, M.R. Russo, An approach to the Gummel map by vector extrapolation methods, Numer. Algor. 45 (2007) 331–343.
[21] L. Dascal, G. Rosman, R., Kimmel, Efficient Beltrami Filtering of Color Images Via Vector Extrapolation, LNCS 4485/2008, SSVM07, 2007, pp. 92–103.
[22] K. Shimano, S. Yonezu, Y. Enomoto, Acceleration of unsteady incompressible flow calculation using extrapolation methods (fluids engineering) J, Trans. Japan Soc. Mech. Eng. B 74 (745) (2008) 1896–1902 (in Japanese).
[23] C. Brezinski, M. Redivo-Zaglia, Extrapolation Methods Theory and Practice, North-Holland, Amsterdam, 1991.
[24] D.A. Smith, W.F. Ford, A. Sidi, Extrapolation methods for vector sequences, SIAM Rev. 29 (1987) 199–233.
[25] K. Jbilou, H. Sadok, Some results about vector extrapolation methods and related fixed-point iterations, J. Comput. Appl. Math. 36 (1991) 385–398.
[26] C. Hirsch, Numerical Computation of Internal and External Flows, Fundamentals of Numerical Discretization, John Wiley & Sons, Chichester, England, 1995. vol. 1.
[27] P. Nithiarasu, O.C. Zienkiewicz, Analysis of an explicit and matrix free fractional step method for incompressible flows, Comput. Meth. Appl. Mech. Eng. 195 (2006) 5537–5551.
[28] M. Massarotti, F. Arpino, R.W. Lewis, P. Nithiarasu, Explicit and semi-implicit CBS procedures for incompressible viscous flows, Int. J. Numer. Meth. Eng. 66 (2006) 1618–1640.
[29] C. Brezinski, M. Redivo Zaglia, New vector sequence transformations, Linear Algebra Appl. 389 (2004) 189–213.
[30] C. Brezinski, M. Redivo-Zaglia, The PageRank vector: properties, computation, approximation, and acceleration, SIAM J. Matrix Anal. Appl. 28 (2006) 551–575.
[31] R.W. Lewis, P. Nithiarasu, K.N. Seetharamu, Fundamentals of the Finite Element Method for Heat and Fluid Flow, John Wiley & Sons, Chichester, England, 2004.