Commun. Comput. Phys.
doi: 10.4208/cicp.041209.160910a

Vol. **9**, No. 4, pp. 1020-1039
April 2011

A Comparison of Semi-Lagrangian and Lagrange-Galerkin hp-FEM Methods in Convection-Diffusion Problems

Pedro Galán del Sastre[1,*] and Rodolfo Bermejo[2]

[1] *Departamento de Matemática Aplicada al Urbanismo, a la Edificación y al Medio Ambiente, E.T.S.A.M., Universidad Politécnica de Madrid, Avda. Juan de Herrera 4, 28040 Madrid, Spain.*
[2] *Departamento de Matemática Aplicada, E.T.S.I.I., Universidad Politécnica de Madrid, C/ José Gutiérrez Abascal 2, 28006 Madrid, Spain.*

Received 4 December 2009; Accepted (in revised version) 16 September 2010

Communicated by Jie Shen

Available online 10 November 2010

Abstract. We perform a comparison in terms of accuracy and CPU time between second order BDF semi-Lagrangian and Lagrange-Galerkin schemes in combination with high order finite element method. The numerical results show that for polynomials of degree 2 semi-Lagrangian schemes are faster than Lagrange-Galerkin schemes for the same number of degrees of freedom, however, for the same level of accuracy both methods are about the same in terms of CPU time. For polynomials of degree larger than 2, Lagrange-Galerkin schemes behave better than semi-Lagrangian schemes in terms of both accuracy and CPU time; specially, for polynomials of degree 8 or larger. Also, we have performed tests on the parallelization of these schemes and the speed-up obtained is quasi-optimal even with more than 100 processors.

AMS subject classifications: 65M60, 65L60, 65N30, 65M25, 76M10, 76M25

Key words: Navier-Stokes equations, convection-diffusion equations, semi-Lagrangian, Lagrange-Galerkin, second order backward difference formula, hp-finite element method.

1 Introduction

This work is devoted to the study of convection dominated-diffusion problems and in particular the Navier-Stokes equations. Many efforts have been made by the numerical analysis community, and the Eulerian-Lagrangian approach has proven to produce very

*Corresponding author. *Email addresses:* pedro.galan@upm.es (P. Galán del Sastre), rbermejo@etsii.upm.es (R. Bermejo)

good results in many different situations. To fix ideas, we shall consider two models: (i) the prototype model of convection-diffusion equations

$$\begin{cases} \dfrac{\partial w}{\partial t}+\vec{u}\cdot\nabla w-\nu\Delta w=f, & \text{in } \Omega\times(0,T), \\ w|_{\partial\Omega}=g, \\ w(0)=w_0, \end{cases} \tag{1.1}$$

where $\Omega\subset\mathbb{R}^2$ is an open bounded subset, $\vec{u}$ is the prescribed velocity vector, f denotes the forcing term, ν is the diffusion coefficient whereas w_0 and g denote the initial and Dirichlet boundary conditions respectively; (ii) the time-dependent incompressible Navier-Stokes equations

$$\begin{cases} \dfrac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}-\nu\Delta\vec{u}=-\nabla p+\vec{f}, & \text{in } \Omega\times(0,T), \\ \operatorname{div}\vec{u}=0, & \text{in } \Omega, \\ \vec{u}|_{\partial\Omega}=\vec{g}, \\ \vec{u}(0)=\vec{u}_0, \end{cases} \tag{1.2}$$

where $\Omega\subset\mathbb{R}^2$ is an open bounded subset, $\vec{u}$ is the velocity of the fluid, p is the pressure, $\vec{f}$ is the forcing term, ν is the diffusion coefficient, $\vec{u}_0\in L^2(\Omega)$ is the prescribed initial condition and $\vec{g}$ is the boundary conditions. For the sake of simplicity, we shall consider homogeneous Dirichlet boundary conditions in the description of the numerical schemes.

In this article we propose a comparative study in terms of accuracy and CPU time of two of the most popular methods when using the Eulerian-Lagrangian approach, namely the Lagrange-Galerkin (or Characteristic Galerkin) and the semi-Lagrangian methods, because so far it is not clear which one of them should be used.

Both methods discretize the material derivative

$$\frac{Dw}{Dt}=\frac{\partial w}{\partial t}+\vec{u}\cdot\nabla w,$$

(in the Navier-Stokes equation, $w=\vec{u}$) along the characteristic curves $X(x,t_{n+1};t)$ defined as follows:

$$\begin{cases} X'(x,t_{n+1};t)=\vec{u}\big(X(x,t_{n+1};t),t\big), \\ X(x,t_{n+1};t_{n+1})=x, \end{cases} \tag{1.3}$$

where $t_{n-l}\le t\le t_{n+1}$, $l\in\mathbb{N}\cup\{0\}$, $x\in\Omega$, $t_j=j\Delta t$, and Δt the time step in the numerical scheme. Note that the characteristics curves are discretized backwards in time. This is one of the main advantages of the Eulerian-Lagrangian approach versus the Lagrangian one, which allows the mesh to follow the trajectories of the flow and so, the mesh is subject to large deformations. Another advantage of Eulerian-Lagrangian methods is that they have a large stability region, so that in the applications the time step can be chosen taking into account only accuracy considerations.

Using the definition of the characteristics curves (1.3), the material derivative can be written as follows:

$$\frac{Dw}{Dt}\big(X(x,t_{n+1};t),t\big)=\frac{d}{dt}\big(w(X(x,t_{n+1};t),t)\big),$$

so that it is possible to discretize the material derivative using any classical formula. In this work we propose an implicit second order BDF formula for time discretization, first proposed by [7] for convection-diffusion equations, and latter on used for Navier-Stokes equation by [2] and [19] among others. Thus for each $x\in\Omega$,

$$\frac{Dw}{Dt}\big(X(x,t_{n+1};t_{n+1}),t_{n+1}\big)\simeq\frac{3w^{n+1}(x)-4w^{n*}(x)+w^{(n-1)**}(x)}{2\Delta t},\tag{1.4}$$

where the following notation has been used for each n:

$$\begin{aligned}&w^n(x)=w(x,t_n),\\&w^{n*}(x)=w(X(x,t_{n+1};t_n),t_n),\\&w^{n**}(x)=w(X(x,t_{n+2};t_n),t_n),\end{aligned}$$

where $X(x,t_{n+1};t_n)$ and $X(x,t_{n+2};t_n)$ are the position at time t_n of the points that moving with the flow velocity will be at x at time instants t_{n+1} and t_{n+2} respectively; usually these points are known as the feet of the characteristics or departure points.

As we describe below, semi-Lagrangian schemes calculate $w^{n*}(x)$ and $w^{n**}(x)$ by interpolation for $x=x_i$, x_i being the mesh points; whereas Lagrange-Galerkin schemes calculate the functions w^{n*} and w^{n**} by L^2-projection into the finite element space. Hence, apparently semi-Lagrangian schemes are easier to implement and faster than Lagrange-Galerkin schemes, but Lagrange-Galerkin schemes have proven to be less dissipative and more accurate than semi-Lagrangian methods. In this work, an exhaustive study of both methods will be carried out in the *hp* finite element framework to ascertain which one of them can be better considering the type (modal or nodal) and the degree of polynomials to be used.

The paper is organized as follows. The numerical schemes are presented in Section 2, firstly the discretization in time for both, the convection-diffusion model and the Navier-Stokes equations, and secondly the spatial discretization using the *hp* finite element method. We finish the section by describing the semi-Lagrangian and the Lagrange-Galerkin method as well as a *p*-adaptive version of the Lagrange-Galerkin method. In Section 3 the numerical examples are presented with the Gaussian bell test and the cavity problem. Finally, some notes about parallelization referred to the convective step are given.

2 Numerical schemes

This section is devoted to the description of the numerical schemes to discretize Eqs. (1.1) and (1.2). We use the following notation related to the L^2-norm throughout this paper.

Let $u,v\in L^2(\Omega)$, then

$$(u,v)=\int_\Omega uv \quad \text{and} \quad \|u\|^2=\int_\Omega |u|^2,$$

and analogously, if $\vec{u},\vec{v}\in L^2(\Omega)^2$,

$$(\vec{u},\vec{v})=\int_\Omega \vec{u}\cdot\vec{v} \quad \text{and} \quad \|\vec{u}\|^2=\int_\Omega |\vec{u}|^2.$$

Let $\Delta t>0$ be the time step, and $t_n=n\Delta t$. Then when Eulerian-Lagrangian approach is used to discretize (1.1) combined with (1.4), we have to solve for each n,

$$\begin{cases} \dfrac{3w^{n+1}-4w^{n*}+w^{(n-1)**}}{2\Delta t}-\nu\Delta w^{n+1}=f, & \text{in } \Omega, \\ w\,|_{\partial\Omega}^{n+1}=0, \end{cases}$$

or in variational formulation,

$$\frac{3}{2}(w^{n+1},v)+\nu\Delta t(\nabla w^{n+1},\nabla v)=\left(2w^{n*}-\frac{1}{2}w^{(n-1)**},v\right)+\Delta t(f,v), \tag{2.1}$$

for all $v\in H_0^1(\Omega)$.

Following the idea used in the discretization of problem (1.1), the numerical scheme for the Navier-Stokes equations would be

$$\begin{cases} \dfrac{3\vec{u}^{n+1}-4\vec{u}^{n*}+\vec{u}^{(n-1)**}}{2\Delta t}-\nu\Delta\vec{u}^{n+1}=-\nabla p^{n+1}+\vec{f}, & \text{in } \Omega, \\ \operatorname{div}\vec{u}^{n+1}=0, & \text{in } \Omega, \\ \vec{u}\,|_{\partial\Omega}^{n+1}=0. \end{cases} \tag{2.2}$$

Then, for each n, a Stokes problem must be solved. In the literature there are some possibilities to solve the Stokes problem in (2.2), using, for instance, the conjugate gradient algorithm [4] or the Uzawa one [3]. Nevertheless, some of the most popular are the splitting schemes because they decouple the velocity and the pressure which results in a very efficient scheme from the computational point of view. One of the main disadvantages of the splitting schemes consists in the error committed when decoupling the velocity and the pressure. But in the past years there have been proposed several splitting schemes that lead to a priori error estimates of $\mathcal{O}(\Delta t^2)$, see [11] or [10] for instance. In this work we split the pressure and the velocity based on the scheme (2.2) via extrapolation of the term u^{n+1} in the first step of the scheme, and using the so called rotational formulation [12]. Specifically, in variational formulation we have the following equations:

1. Set $\vec{u}_e^{n+1}=2\vec{u}^n-\vec{u}^{n-1}$, and find $p^{n+1}\in H^1(\Omega)$, such that

$$\begin{aligned}(\nabla p^{n+1},\nabla v)=&(\vec{f},\nabla v)-\frac{1}{2\Delta t}(3\vec{u}_e^{n+1}-4\vec{u}^{n*}+\vec{u}^{(n-1)**},\nabla v)\\ &-\nu(\nabla\times\nabla\times\vec{u}_e^{n+1},\nabla v),\end{aligned} \tag{2.3}$$

for all $v\in H^1(\Omega)$.

2. Find $\vec{u}^{n+1}\in H_0^1(\Omega)^2$, such that

$$\frac{3}{2}(\vec{u}^{n+1},\vec{v})+\nu\Delta t(\nabla\vec{u}^{n+1},\nabla\vec{v})=\left(2\vec{u}^{n*}-\frac{1}{2}\vec{u}^{(n-1)**},\vec{v}\right)+\Delta t(-\nabla p^{n+1}+\vec{f},\vec{v}),\qquad(2.4)$$

for all $\vec{v}\in H_0^1(\Omega)^2$.

In fact, this splitting is equivalent to the one proposed in [13] and [12] when solving the Navier-Stokes equations with spectral/hp element method and finite element method respectively and also in [19] where the authors apply the semi-Lagrangian method combined with this splitting.

Now, we follow with the spatial discretization where we propose the hp-finite element method.

2.1 The hp-finite element method generalities

Suppose that $\Omega\subset\mathbb{R}^2$ is an open bounded subset with a sufficiently smooth boundary and D_h is a partition of $\bar{\Omega}$ such that

$$D_h=\{Q_j\}_{j=1}^{NE}\subset\bar{\Omega},$$

where Q_j is a quadrilateral element and NE denotes the total number of elements. We assume that there exists a real constant $\sigma>0$ such that the next hypotheses hold for any D_h:

1. $\bar{\Omega}=\bigcup_{j=1}^{NE}Q_j$.
2. Any face of Q_j is either a subset of $\partial\Omega$ or any other face of other Q_i, with $i\neq j$.
3. Let $h_j=\text{diam}Q_j$, and $\rho_j=\sup\{\text{diam}S\colon S \text{ a ball contained in } Q_j\}$. Then $h_j/\rho_j<\sigma$, for all $1\leq j\leq NE$.

Hereafter, $h=\max_{1\leq j\leq NE}h_j$.

We define the finite element subspaces V_h and V_{h0} associated to the partition D_h by

$$V_h=\left\{v_h\in C(\bar{\Omega})\colon\ v_{h|Q_j}\in P_m(Q_j),\ \text{for all } 1\leq j\leq NE\right\},$$
$$V_{h0}=V_h\cap H_0^1(\Omega),$$

where $P_m(Q_j)$ is defined as follows for a fixed $m\in\mathbb{N}$. Let $\hat{Q}=[-1,1]^2$ be the reference element, and let $T_j:\hat{Q}\to Q_j$ be the continuous bijective transformation between $\hat{Q}$ and Q_j. Then,

$$P_m(Q_j)=\left\{p\in C(Q_j)\colon\ p=\hat{p}\circ T_j^{-1},\ \ \hat{p}\in P_m([-1,1])\otimes P_m([-1,1])\right\},$$

where $P_m([-1,1])$ is the set of polynomials of degree $\leq m$ in the interval $[-1,1]$.

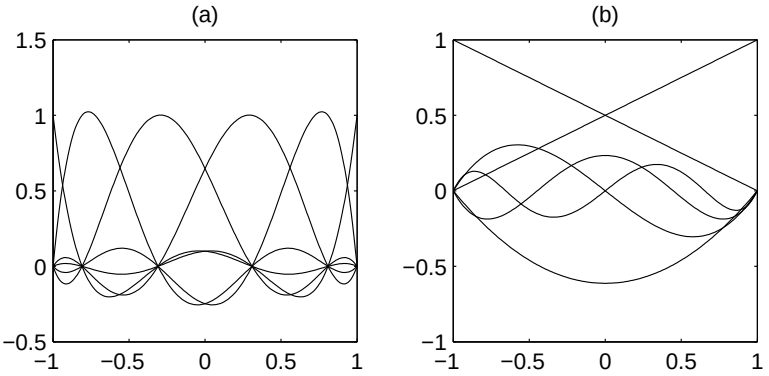

Figure 1: Basis of $P_m([-1,1])$ for $m=5$ using nodal polynomials, (a), and modal polynomials, (b).

Depending on the implementation of the algorithm used in the hp finite element method, there are two choices for the basis of $P_m([-1,1])$, usually known as modal and nodal expansion.

In the nodal expansion, the basis of $P_m([-1,1])$ consists of $m+1$ Lagrange polynomials associated to $m+1$ nodal points. Clearly, the choice of the nodal points is crucial in terms of the oscillations of the polynomials as well as the condition number of both the mass and stiffness matrices. Perhaps the most popular nodal expansion is the so called "spectral elements" that use the zeros of the Gauss-Lobatto polynomials as nodal points, for instance, see [14] as a general reference on spectral methods and [9] as a particular reference on the application of spectral finite elements combined with a first order LG scheme to calculate a numerical solution of the convection-diffusion equation.

On the other hand, in the modal expansion the polynomials of the basis in $P_m([-1,1])$ are not associated to nodal points, instead, a hierarchical set of polynomials is used to generate the basis, in the sense that if $B_m=\{l_i\}_{i=0}^m$ is the basis of $P_m([-1,1])$, then $B_m\subset B_{m+1}$ for all m. Again there are different choices, but the Lobatto functions is a nice possibility (obtained as the integral of the Legendre polynomials or also as a specific family of the Jacobi polynomials, see [14,18]). Using this basis, the first two polynomials, l_0 and l_1 are linear with value 0 or 1 at $x=\pm 1$, and the $(j+1)$-th polynomial in the basis, l_j, is a polynomial of degree j. One of the main advantages of the polynomials of this basis is that

$$\int_{-1}^{1} l_i' l_j' = 0, \quad \text{if } i\neq j \text{ and } i>1 \text{ or } j>1;$$

so that the stiffness matrix maintains a good condition number. This property is important when solving Poisson type problems as, for example, the one in (2.3). In Fig. 1(b) we show the modal basis for $m=5$ where we can compare with the nodal basis also for $m=5$.

Of course, depending on the problem it can be more convenient to use a nodal or modal basis in the finite element formulation. We shall inspect how they perform in both the semi-Lagrangian and Lagrange-Galerkin methods.

Hereafter we shall denote $\{\varphi_i\}_{i=1}^{NN} \subset V_h$ as the basis of V_h when using nodal polynomials whereas $\{\phi_i\}_{i=1}^{NN} \subset V_h$ will denote the basis when modal polynomials are to be used, where $NN = \dim V_h$. Note that when using $\{\varphi_i\}_{i=1}^{NN}$, there is a set of nodes $\{x_i\}_{i=1}^{NN} \subset \bar{\Omega}$ associated to D_h such that $\varphi_i(x_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Let $\{z_i\}_{i=0}^{m} \subset [-1,1]$ be the zeros of the Gauss-Lobatto polynomials of degree m, then for all $1 \le k \le NN$ and for any $1 \le j \le NE$ and $0 \le i,i' \le m$, one defines the nodes $x_k = T_j(z_i, z_{i'})$. On the other hand, no nodes are associated to the basis $\{\phi_i\}_{i=1}^{NN}$.

Once the finite element subspaces are defined, the full discretization of (1.1) becomes: for each n, find $w_h^{n+1} \in V_{h0}$, such that

$$\frac{3}{2}(w_h^{n+1}, v_h) + \nu\Delta t(\nabla w_h^{n+1}, \nabla v_h) = \left(2w_h^{n*} - \frac{1}{2}w_h^{(n-1)**}, v_h\right) + \Delta t(f, v_h), \tag{2.5}$$

for all $v_h \in V_{h0}$. Analogously, the Navier-Stokes equations (1.2) are discretized in two steps using the splitting (2.3)-(2.4) as follows. For each n:

1. Set $\vec{u}_{he}^{n+1} = 2\vec{u}_h^n - \vec{u}_h^{n-1}$, and find $p_h^{n+1} \in V_h$, such that

$$\begin{aligned}(\nabla p_h^{n+1}, \nabla v_h) = (\vec{f}, \nabla v_h) - \frac{1}{2\Delta t}(3\vec{u}_{he}^{n+1} - 4\vec{u}_h^{n*} + \vec{u}_h^{(n-1)**}, \nabla v_h) \\ - \nu(\nabla \times \nabla \times \vec{u}_{he}^{n+1}, \nabla v_h),\end{aligned} \tag{2.6}$$

   for all $v_h \in V_h$.

2. Find $\vec{u}_h^{n+1} \in V_h \times V_h$, such that

$$\frac{3}{2}(\vec{u}_h^{n+1}, \vec{v}_h) + \nu\Delta t(\nabla \vec{u}_h^{n+1}, \nabla \vec{v}_h) = \left(2\vec{u}_h^{n*} - \frac{1}{2}\vec{u}_h^{(n-1)**}, \vec{v}_h\right) + \Delta t(-\nabla p_h^{n+1} + \vec{f}, \vec{v}_h), \tag{2.7}$$

   for all $\vec{v}_h \in V_{h0} \times V_{h0}$.

In Eqs. (2.5)-(2.7), we have to compute an L^2-inner product involving any of $w_h^{n*}, w_h^{(n-1)**}, \vec{u}_h^{n*}$ or $\vec{u}_h^{(n-1)**}$ with $v_h \in V_h$, $\vec{v}_h \in V_h \times V_h$ or ∇v_h, with $v_h \in V_h$. We must note that in general $w_h^{n*}, w_h^{(n-1)**} \notin V_h$ and $\vec{u}_h^{n*}, \vec{u}_h^{(n-1)**} \notin V_h \times V_h$, so that the computations of these inner products are not trivial. One way to overcome this difficulty is substituting w_h^{n*} by either $P_h w_h^{n*}$ or $I_h w_h^{n*}$ (and analogously for $w_h^{(n-1)**}, \vec{u}_h^{n*}$ and $\vec{u}_h^{(n-1)**}$), where $P_h : L^2(\Omega) \to V_h$ is the L^2-projector operator onto V_h, i.e., for each $w \in L^2(\Omega)$,

$$\begin{cases} P_h w \in V_h, \\ (P_h w, v_h) = (w, v_h), \quad \text{for all } v_h \in V_h, \end{cases}$$

and $I_h : C(\bar{\Omega}) \to V_h$ is the interpolant operator, i.e., for each $w \in C(\bar{\Omega})$,

$$\begin{cases} I_h w \in V_h, \\ I_h w(x_i) = w(x_i), \quad \text{for all } x_i, \text{ where } \{x_i\}_{i=1}^{NN} \text{ is a set of nodes in } D_h. \end{cases}$$

When the L^2-projection is used, the resulting scheme is the so called Lagrange-Galerkin, whereas it is called semi-Lagrangian when the interpolant operator is applied. In the next two subsections, we discuss the advantages and disadvantages of both methods.

2.2 The semi-Lagrangian method

We shall describe in this section the computation of $I_h w_h^{n*}, I_h w_h^{(n-1)**}, I_h \vec{u}_h^{n*}$ and $I_h \vec{u}_h^{(n-1)**}$. The set of nodes $\{x_i\}_{i=1}^{NN}$ in D_h, associated to the interpolant operator I_h, are the nodes of the set of nodal basis $\{\varphi_i\}_{i=1}^{NN}$. Note that the modal basis $\{\phi_i\}_{i=1}^{NN}$ are not associated with any set of nodal points, however, when using this modal basis with semi-Lagrangian methods we propose that the set of nodes associated to the interpolant operator is the same as the one of the nodal basis. Thus, the function $I_h w_h^{n*} \in V_h$ (analogously $I_h w_h^{(n-1)**}, I_h \vec{u}_h^{n*}$ and $I_h \vec{u}_h^{(n-1)**}$) can be univocally determined once

$$w_h^{n*}(x_i) = w_h^n\big(X(x_i,t_{n+1};t_n)\big)$$

has been calculated for all $1 \le i \le NN$. Then, we first have to compute $X(x_i,t_{n+1};t_n)$ as the solution of (1.3) for each node x_i. Usually the ODE system (1.3) can not be solved analytically, and in particular, in the case of the Navier-Stokes equations where the velocity is not known at time t_{n+1}. So that, the ODE (1.3) is usually solved numerically for each node x_i using an ODE solver. Some of the most popular ODE solvers when implementing the semi-Lagrangian scheme are the fourth order Runge-Kutta and the second order fix point method [17]. These solvers have proven to give accurate results when combined with the semi-Lagrangian method. While the fourth order Runge-Kutta gives very accurate results, the second order fix point performs also very good results (second order, while the Runge-Kutta is fourth order) while maintaining a very cheap implementation in terms of CPU time. Some other solvers, as the second order Runge-Kutta method, are also second order accurate, nevertheless computations do not maintain the same accuracy as the fix point solver [17]. In this work, we choose the forth order Runge-Kutta solver, although more expensive than the fix point solver, more accurate.

Remark 2.1. The Runge-Kutta method of order four we use is as follows. At time t_{n+1}, $n \ge 3$, given $\vec{u}^n, \vec{u}^{n-1}, \vec{u}^{n-2}$ and $\vec{u}^{n-3}$, calculate

$$\vec{K}_1 = \Delta t \vec{u}_h^{n+1}(x), \qquad \vec{K}_2 = \Delta t \vec{u}_h^{n+\frac{1}{2}}\Big(x - \frac{1}{2}\vec{K}_1\Big),$$
$$\vec{K}_3 = \Delta t \vec{u}_h^{n+\frac{1}{2}}\Big(x - \frac{1}{2}\vec{K}_2\Big), \qquad \vec{K}_4 = \Delta t \vec{u}_h^{n}(x - \vec{K}_3),$$
$$X(x,t_{n+1};t_n) = x - \frac{1}{6}\Big(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4\Big),$$

where $\vec{u}_h^{n+1}(\cdot)$ and $\vec{u}_h^{n+1/2}(\cdot)$ are extrapolated by formulas of order four. However, for $n = 1,2$ and 3 we use extrapolation formulas of lower order. Nevertheless, in many nu-

merical experiments we have observed that extrapolation formulas of order two give similar results of those obtained with extrapolation formulas of order two.

Note that when computing $I_h w_h^{n*} \in V_h$ we have to compute the coefficients $\{\alpha_i\}_{i=1}^{NN}$ or $\{\beta_i\}_{i=1}^{NN}$ such that

$$I_h w_h^{n*}(x) = \sum_{i=1}^{NN} \alpha_i \varphi_i(x) = \sum_{i=1}^{NN} \beta_i \phi_i(x) \tag{2.8}$$

depending on the polynomials chosen in the basis of V_h.

From the implementation view point the simplest way for the interpolation is to use nodal polynomials because the coefficients $\alpha_i = w_h^{n*}(x_i)$, which are easy to compute by interpolation of w_h^n at the points $X(t_{n+1},x_i;t_n)$.

In the case of modal polynomials we have to compute the coefficients β_i. This can be done element by element. The description of the algorithm can be summarized as follows:

(i) For all $i=1,2,\cdots,NN$, compute $w_h^{n*}(x_i)$.

(ii) If x_i is a vertex of any Q_j, then $\beta_i = w_h^{n*}(x_i)$.

(iii) For any edge of any element Q_j, let $p(x)=\sum_{i=0}^{m}\gamma_i l_i(x)$, $x\in[-1,1]$, such that $p(z_k)=w_h^{n*}(x_{i_k})$, with $(z_k,1)$, $(z_k,-1)$, $(1,z_k)$ or $(-1,z_k)\in\hat{Q}$ mapped into x_{i_k} for $k=0,1,\cdots,m$. The coefficients γ_i can be computed easily inverting an $(m-1)\times(m-1)$ matrix and using the fact that $\gamma_0=\beta_{i_0}$ and $\gamma_1=\beta_{i_1}$ computed in (ii).

(iv) Similarly to (iii), for any element Q_j, let $q(x,y)=\sum_{k,l=0}^{m}\eta_{kk'}l_k(x)l_{k'}(y)$, $-1\le x,y\le 1$, with $T_j(z_k,z_{k'})=x_{i_{kk'}}\in Q_j$. We compute $q(x,y)$ such that $q(z_k,z_{k'})=w_h^{n*}(x_{i_{kk'}})$, for $0\le k,k'\le m$. Again, $\eta_{kk'}=\beta_{i_{kk'}}$ that has already been computed in (ii) and (iii) for $k\le 1$ or $k'\le 1$. Thus, the coefficients $\eta_{kk'}$ can be calculated inverting an $(m-1)^2\times(m-1)^2$ matrix.

Nevertheless, in order to compute the coefficients β_i we still have to compute $w_h^{n*}(x_i)$ for all i and then solve one linear system for each element.

Though this procedure seems to be more expensive than the one needed in the nodal case, we must note that the computation of $w_h^{n*}(x_i)$ is much cheaper in the modal case. The point is that the interpolation of the velocity required when solving the ODE (1.3) at points located at the interior of any element, and also the interpolation of w_h^n at $X(t_{n+1},x;t_n)$ is much cheaper in the modal case because the modal basis is hierarchical with increasing polynomial degree, whereas in the nodal case all the polynomials of the basis have the same degree. When performing the interpolation we have to evaluate $(m+1)^2$ polynomials of degree m (say $m\otimes m$ in x and y directions respectively) in the nodal case whereas we have $(m+1)^2$ polynomials of degree $\le m$ in the modal case (specifically, if $\phi_i(x,y)\,|_{Q_j}=l_k(\hat{x})l_{k'}(\hat{y})$,with $(x,y)\in Q_j\subset\Omega$ and $(\hat{x},\hat{y})\in\hat{Q}$, $\deg(l_k(\hat{x})l_{k'}(\hat{y}))=\max\{1,k\}\otimes\max\{1,k'\}$). In the numerical examples we shall show that when m is large, the modal polynomial basis is much more efficient than the nodal one.

2.3 The Lagrange-Galerkin method

From the computational point of view it is not necessary to calculate explicitly the projection $P_h w_h^{n*}$ because what one needs is to compute (w_h^{n*},v_h), see (2.5), due to the fact that

$$(P_h w_h^{n*},v_h)=(w_h^{n*},v_h), \quad \text{for all } \ v_h\in V_h.$$

Note that $P_h w_h^{n*}$ is the L^2-projection of w_h^{n*} onto V_h and may be necessary for the convergence analysis of the methods [5,16].

Thus, we focus on the calculation of (w_h^{n*},v_h) that is performed by a Gauss quadrature rule of high order,

$$(w_h^{n*},v_h)=\sum_{j=1}^{NE}\int_{Q_j} w_h^{n*}v_h=\sum_{j=1}^{NE}\int_{\hat{Q}} \hat{w}_h^{n*}\hat{v}_h|J_j|,$$

where $\hat{w}_h^{n*}=w_h^{n*}\circ T_j$, $\hat{v}_h=v_h\circ T_j$ and J_j being the Jacobian determinant of the transformation T_j. Finally, we approximate

$$\int_{\hat{Q}} \hat{w}_h^{n*}\hat{v}_h|J_j|\simeq\sum_{i=1}^{NPG}\omega_i\hat{w}_h^{n*}(\hat{x}_i)\hat{v}_h(\hat{x}_i)|J_j(\hat{x}_i)|,$$

where ω_i and $\hat{x}_i\in\hat{Q}$ are the weights and points respectively of the numerical quadrature in the reference element and NPG is the number of points used in the quadrature formula. In this paper we use the Gauss-Legendre quadrature rule of order $(m+2)\times(m+2)$ points, m being the degree of polynomials in V_h. Note that the scheme requires the computation of (w_h^{n*},v_h) for all v_h in the finite element space, or equivalently, for $v_h=\varphi_k$ or $v_h=\phi_k$ for all k. The computation of $\hat{v}_h(\hat{x}_i)=v_h(T_j\hat{x}_i)$ can be done once for all. On the other hand, the computation of $\hat{w}_h^{n*}(\hat{x}_i)=w_h^{n*}(T_j\hat{x}_i)$ can be done using the same methodology explained for the semi-Lagrangian method.

In [15] it is pointed out that in Lagrange-Galerkin methods it is convenient to use high order quadrature rules to maintain the stability and convergence properties that the method has when the integrals are calculated exactly. The rules we are using calculates exactly for polynomials of degree $2m+3$ in each coordinate direction.

Note that while the implementation in the semi-Lagrangian method depends on whether the polynomial basis of V_h is nodal or modal, the Lagrange-Galerkin method has no differences in the implementation in this respect. As we pointed out before, modal polynomials are more efficient in interpolating, so that we carry out all the numerical examples using modal polynomials with the Lagrange-Galerkin method.

2.4 Some computation aspects about the semi-Lagrangian and Lagrange-Galerkin methods

In both methods the function w_h^{n*} has to be evaluated either at the nodes associated to D_h or at the points of the numerical quadrature. In the case of the semi-Lagrangian method

the number of evaluations is $\dim V_h$, whereas in the Lagrange-Galerkin method the number of evaluations is $NPG \times NE$. Therefore, the number of evaluations in the case of Lagrange-Galerkin method is larger than in the semi-Lagrangian method. Nevertheless, the Lagrange-Galerkin method has proven to be more accurate than semi-Lagrangian method, see [1] for instance, because the interpolation procedure of the latter introduces an additional artificial diffusion in the numerical calculation.

2.5 The adaptive Lagrange-Galerkin method

Modal hierarchical bases make easier the implementation of p-adaptivity, see [18] for instance. Our procedure to implement p-adaptivity in Lagrange-Galerkin method for the convection-diffusion equations (1.1) is as follows:

1. When solving w_h^{n+1} using the variational formulation (2.5), we assign an interpolation order $m_j \leq m$ to each element Q_j.
2. If $X(t_{n+1},x_i;t_n) \in Q_j$, then $w_h^{n*}(x_i)$ is computed via interpolation using only the polynomials in Q_j of degree $\leq m_j$.

Note that $X(t_{n+1},x_i;t_n)$ is computed solving the ODE (1.3), and, as we mentioned above, this also requires the interpolation of the velocity $\vec{u}$ at points located in the interior of any element of D_h. Thus, defining an order m_j' in Q_j for the velocity

$$u_h^n(x) = \sum_{p=1}^{NN} u_p^n \phi_p(x),$$

and supposing that $X(t_{n+1},x_i;t_n) \in Q_j$ for any j, then

$$u_h^n\big(X(t_{n+1},x_i;t_n)\big) = \sum_{k,k'=0}^{m_j'} u_{kk'}^n (l_k \otimes l_{k'})\big(T_j^{-1}(X(t_{n+1},x_i;t_n))\big),$$

where

$$u_{kk'}^n = u_p^n, \quad \text{if } \phi_p = (l_k \otimes l_{k'}) \circ T_j^{-1}.$$

Similarly, in the Navier-Stokes equations (1.2) the p-adaptivity can be implemented using the same procedure. Note that in this case $m_j' = m_j$.

3 Numerical results

To ascertain the behavior of the schemes considered in this paper in terms of CPU time and accuracy, we run two numerical tests, the first one for problem (1.1) and the second one for the Navier-Stokes equations. The test for the convection-diffusion equation is the so called Gaussian bell problem; we choose this test for the following reasons. First,

we know the analytical solution, so that we can calculate the error of the schemes; second, although simple, this problem with a small diffusion coefficient ν possesses all the ingredients to make the Eulerian methods have a hard time.

The test for the Navier-Stokes equations is the so called driven cavity problem that, for high Reynolds numbers, has become a benchmark problem for numerical methods used in the integration of the incompressible Navier-Stokes equations.

3.1 The Gaussian bell problem

In this problem we shall compare four different discretizations of the convective terms, namely, two semi-Lagrangian schemes, one using nodal interpolation and the other one using modal interpolation, and two Lagrange-Galerkin schemes, one without adaptivity and the other one with adaptivity. Hereafter, we denote these schemes as *nodal-SL*, *modal-SL*, *LG* and *adap-LG* respectively. We consider problem (1.1) defined on the domain

$$\Omega=\Big\{(x,y)\in\mathbb{R}^2:x^2+y^2\leq 4\Big\},$$

with velocity $\vec{u}(x,y)=(-y,x)$ and initial condition

$$w_0(x,y)=\exp\left(-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}\right),$$

where $(x_0,y_0)=(1/2,0)$ and $\sigma^2=0.0078$. The exact solution is

$$w(x,y,t)=\frac{\sigma^2}{\sigma^2+2\nu t}\exp\left(-\frac{\bar{x}^2+\bar{y}^2}{2(\sigma^2+2\nu t)}\right),$$

where

$$\bar{x}=x-x_0\cos t+y_0\sin t\quad\text{and}\quad\bar{y}=y-x_0\sin t-y_0\cos t.$$

Note that the exact solution represents a Gaussian bell rotating around the point $(0,0)$. To calculate the numerical solution we shall use the boundary conditions obtained from the exact solution. In the numerical calculations we take the parameter $\nu=6.2070\times10^{-4}$ and the time step $\Delta t=2\pi/100$, so that, we need 100 time steps to complete a revolution ($T=2\pi$). We perform numerical tests with different meshes (see Fig. 2) and degrees of polynomials. Table 1 shows the parameters NE (number of elements in the mesh), NN (number of nodes in the mesh) and IN (number of nodes in the inner circle of radius 1) that we use in the numerical experiments. We must remark two items: (i) the mesh in the unit circle is more refined than in the rest of the domain because outside this circle the values of the solution are $\leq 10^{-4}$, so that the error of the numerical solution basically depends on IN; (ii) we keep the outer region to avoid numerical noise.

In Figs. 3-5 we represent for different values of m the average CPU time (upper panel) per time step versus the number of degrees of freedom (d.o.f.) NN, and the L^2-norm error after one revolution versus the number of d.o.f IN (lower panel). The reason to

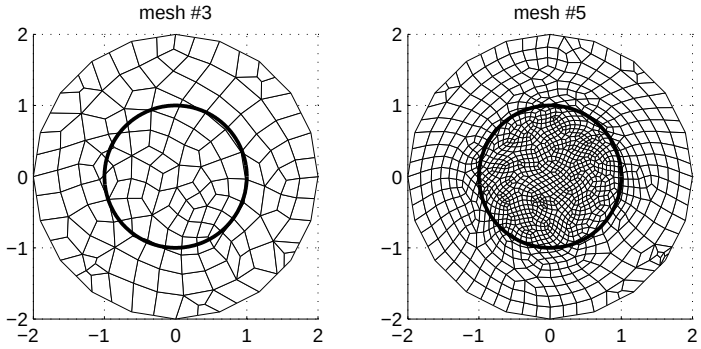

Figure 2: Mesh number 3 and 5 as an example of the refinement in the unit circle depending on the density.

Table 1: Details about the meshes used to solve the Gaussian bell.

	mesh #	1	2	3	4	5	6	7
	NE	55	74	162	435	1263	4104	14854
$m=2$	NN	249	331	691	1799	5115	16487	59495
	IN	39	63	208	805	3195	12878	52512
$m=4$	NN	937	1253	2677	7077	20333	65805	
	IN	155	235	823	3214	12792	51492	
$m=6$	NN	2065	2767	5959	15835	45655	147955	
	IN	346	532	1851	7248	28760	115847	
$m=8$	NN	3633	4873	10537	28073			
	IN	617	943	3281	12885			
$m=10$	NN	5641	7571	16411	43791			
	IN	951	1469	5134	20137			

plot the L^2-norm error in terms of IN is due to the fact, as we mention above, that the solution basically depends on the d.o.f. of the unit circle (IN), whereas the CPU time depends on the total number of d.o.f. The CPU time bars consist of two parts, the lower part corresponds to the CPU time spent in the calculation of the convective terms (the calculation of the feet of the characteristics plus the interpolations for the SL schemes or the quadrature rules for the LG schemes) and the upper part indicates the CPU time needed to solve the symmetric system of equations yielded by (2.5); the system is solved by the Incomplete Cholesky Conjugate Gradient method. Note that the upper part of the CPU time bars is much smaller than the lower part because the matrix is very well conditioned. Furthermore, we remark that when using *adap-LG* the degree of the polynomials m_j varies between 2 and m in all the experiments whereas $m_j'=2$ because the velocity $\vec{u}$ is linear in this example.

For $m=2$ (see Fig. 3) SL schemes require less CPU time than LG schemes. The reason for this behavior is that the number of departure points to be calculated in the SL schemes is smaller than the one in the LG schemes and also the calculation of the quadra-

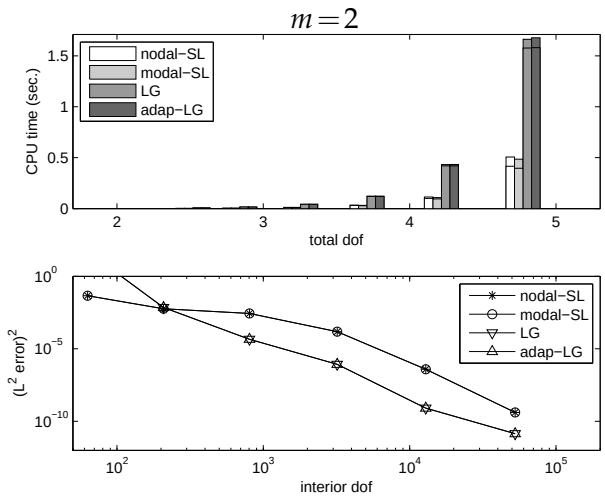

Figure 3: CPU time and L^2 error for meshes #2-#7 and $m=2$.

ture rules requires more number of operations than the pure interpolation performed in the SL schemes; however, the error of the SL schemes is larger than the error of the LG schemes, although both SL and LG schemes have, at least for this example, the same asymptotic rate of convergence. Note that the CPU time is the same for both SL methods, and similarly the CPU time is also the same for both LG methods. For $m=4$, 6, 8 and 10 (see Figs. 4 and 5) things change significantly. First, as the number of d.o.f. increases the

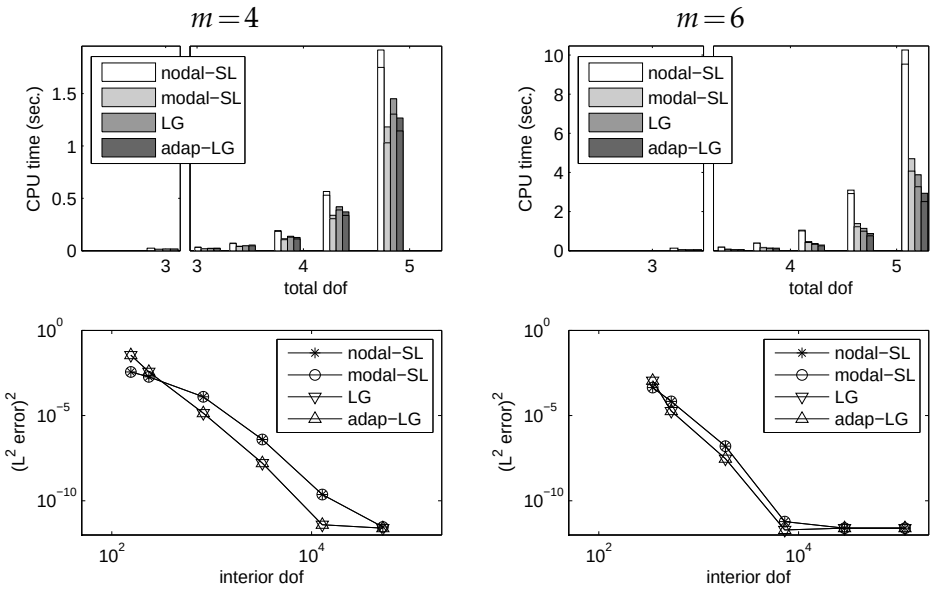

Figure 4: CPU time and L^2 error for meshes #1-#6 for $m=4$ and $m=6$.

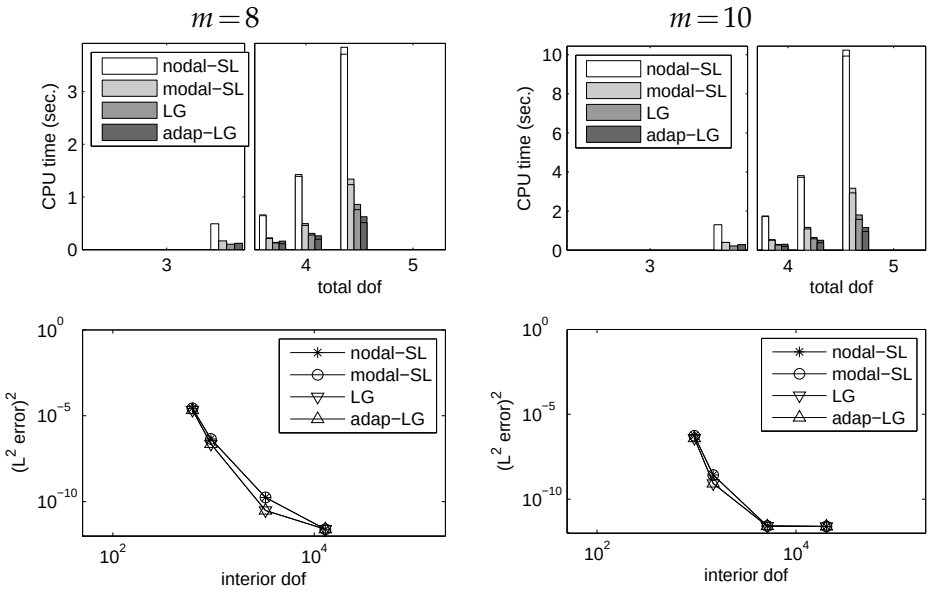

Figure 5: CPU time and L^2 error for meshes #1-#4 for $m=8$ and $m=10$.

nodal-SL scheme becomes more expensive than any other of the schemes. The *modal-SL* scheme is the least expensive of all for $m=4$, but for $m>4$, both *LG* and *adap-LG* schemes are less expensive than the *modal-SL* scheme. For $m\geq 6$, the *adap-LG* scheme is the most efficient of all of them. As for the error, both LG schemes yield smaller errors than the SL schemes, but the asymptotic rate of convergence is about the same for all of them. Note that both SL methods have the same error and the same thing happens for both LG methods.

Next we wish to know whether it is better to use $m=2$ and meshes with a large number of d.o.f. or to use m larger and less d.o.f. Also, we want to know, in both scenarios, which one of these schemes is the most efficient in terms of CPU time. To this end, for each m and each scheme, we have chosen those experiments that, with a tolerance to the square of the L^2-error that is less or equal than 10^{-9}, have the lowest number of d.o.f. The results are shown in Fig. 6, where the average CPU time per time step is represented in the upper panel, the L^2-error in the middle panel and the number of d.o.f in the lower panel, all of them versus the degrees of the polynomials used in the experiments. The lower part of the bars of the d.o.f corresponds to the inner unit circle. Firstly, we note that for large m both *LG* and *adap-LG* schemes are more efficient than the SL schemes, it is worth mentioning that the CPU time for the *nodal-SL* scheme with $m=10$ is 3.81 secs, far larger than that of the other schemes. Secondly, as far as the L^2-error is concerned, the LG schemes are also more efficient than the semi-Lagrangian schemes, for instance, for $m=2$ the *nodal-SL* scheme needs 59495 nodes, whereas the LG schemes need 16487.

From these experiments, and at least for this example, we can conclude that for a fixed tolerance LG schemes are more efficient than the semi-Lagrangian schemes.

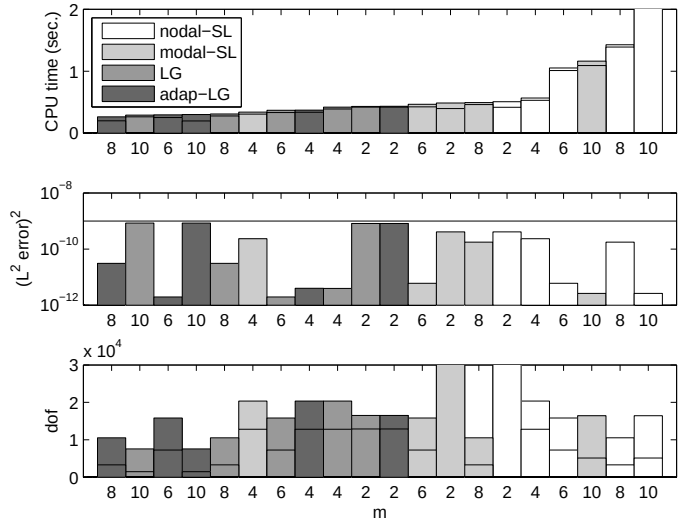

Figure 6: CPU time (upper panel), L^2 error (middle panel) and interior and total number of d.o.f. (lower panel) obtained for the four schemes and $m=2,4,6,8,10$ (in abscissa) using the mesh which gives the square of the L^2-error below the tolerance 10^{-9} with less d.o.f.

3.2 The driven cavity problem

In the Gaussian bell problem we have seen that for a fixed tolerance the LG schemes are more efficient than the semi-Lagrangian schemes; so that, we shall use the LG schemes to solve the Navier-Stokes equations for the driven cavity problem. This example shows the differences between *LG* and *adap-LG* in a Navier-Stokes problem.

We consider problem (1.2) with $\Omega=(0,1)\times(0,1)$, $\vec{u}_0=0$ and $\nu=1/\mathrm{Re}$ in this example, with *Re* the Reynolds number. We assume non-slip boundary conditions, $\vec{u}=(0,0)$, on the left, right and lower boundaries and $\vec{u}=(1,0)$ on the upper boundary. All the computations are carried out with $\Delta t=0.02$, a large time step for this problem ($CFL=20$), but accurate enough to obtain the well accepted results of Ghia et al. [8]. The mesh consists of 10×10 regular uniform quadrilaterals with $m=10$, i.e., 10201 nodes. Hence, in this example $2\leq m_j=m'_j\leq 10$ depending on the regularity of the solution in each element Q_j when using *adap-LG*.

We carry out numerical experiments with $Re=1000$, 3200, 5000 and 10000 and using *LG* and *adap-LG* schemes. In Fig. 7 we show the velocity profiles of u and v at the centerlines $x=0.5$ and $y=0.5$ respectively. In this figure we see the velocity profiles obtained by the *adap-LG* scheme at $T=500$, when the solution has reached the steady state compared with the results of the steady state computed by Ghia et al. in [8]. The velocity profiles shown at $Re=10000$ correspond to the time averaged values, but still the agreement is very good even for this large *Re*. For a throughout and careful study of this problem in a range of Reynolds number as large as 21000 see [6] where a comparison of its results with other studies are also performed.

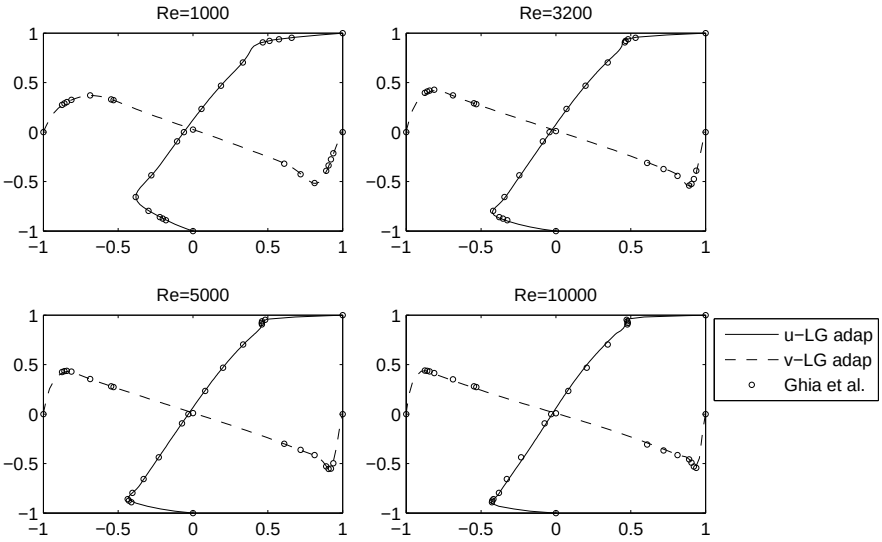

Figure 7: Velocity profiles of u and v at the horizontal and vertical centerlines.

In Fig. 8 we show the CPU time in seconds of the convective step of *LG* and *adap-LG* schemes every time step as well as the evolution of the number of nodes of the *adap-LG* with respect to the time t. Note that in this problem the CPU time required for the convective step with the *adap-LG* scheme can vary in each time step depending on the

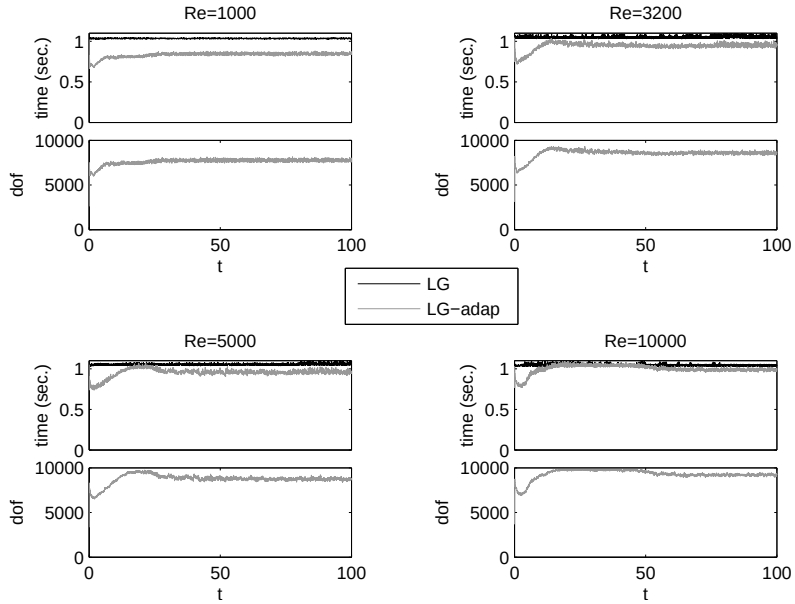

Figure 8: Evolution of the CPU time of the convective step for the *LG* and *adap-LG* scheme (upper panel) and number of nodes (lower panel) for $t \leq 100$.

solution of the cavity problem at each time instant. In contrast, the *LG* scheme spends the same CPU time in every time step as shown in Fig. 8.

In all the computations we see that the number of nodes in the *adap-LG* method undergoes a rapid increase at the beginning of the calculations, then the increase slows down, and near the steady state the number of nodes becomes practically stationary. We also see that at $Re=1000$, a low Reynolds number, the number of nodes required are about 7000 whereas for $Re=10000$, with a more complex fluid structure, the number of nodes is about 9500, close to the limit number of nodes equal to 10201. This results in a faster calculation at low Reynolds numbers, whereas at large Reynolds numbers the CPU time is about the same for both schemes. However, note that even for $Re=10000$ the convective step is slightly faster in the *adap-LG* scheme as shown in Fig. 7.

3.3 Some notes about parallelization

One of the main advantages of the semi-Lagrangian and Lagrange-Galerkin schemes when using m large is that the number of d.o.f. required to get an accurate solution is reduced considerably. However, as we can see in Figs. 3-5, the disadvantage is that the CPU time spent to compute the convective step increases due to the large amount of interpolations that the algorithm needs, but instead it decreases when inverting the matrix associated to the variational formulation (2.5), for the convection-diffusion problem, or (2.6) and (2.7) for the Navier-Stokes equations, since the number of nodes is usually much lower. Nevertheless, although the convective step spends a large percentage of the CPU time needed in each time step (mainly in the Gaussian bell problem, in the driven cavity problem is about 60%, not shown in figures), the nice point is that most of the computations are independent, that means that it is easy to parallelize the computations.

Specifically, for the parallelization of the convective step, we propose that all the processors share the mesh information and that each one of them takes care of the computations of a group of quadrilaterals. Then, if NP is the number of processors, $J_k \subset \mathbb{N}$ is such that all the computations related to the elements $\{Q_j : j \in J_k\} \subset D_h$ are carried out by the processor k, with $1 \leq k \leq NP$. Note that it is important that each processor owns all the information related to the mesh because it will have to compute $X(x_i,t_{n+1};t_n)$ for any i, where x_i are the nodes of the group of the quadrilaterals, and $X(x_i,t_{n+1};t_n)$ may not belong to such a group.

The algorithm needs only two MPI communications. At the beginning of the algorithm an MPI_Bcast is used to share the velocity $\vec{u}_h^n$ and $\vec{u}_h^{n-1}$ (as well as w^n and w^{n-1} in the convection diffusion model) to all the processors. This allows to processor k to compute $X(x_i,t_{n+1};t_{n-l})$, with $l=0$ and 1, for any $x_i \in Q_j$, $j \in J_k$; then $\vec{u}_h^{n*}(x_i)$ and $\vec{u}_h^{(n-1)**}(x_i)$ or $w_h^{n*}(x_i)$ and $w_h^{(n-1)**}(x_i)$; and finally we calculate the integrals, which are the first terms on the right hand side of (2.5), (2.6) and (2.7), in the elements Q_j for $j \in J_k$. The computation is finished once the integrals over the whole domain are done. To do so we require the last communication call, MPI_Reduce, and the convective step is over.

In Fig. 9, we show the CPU time using this parallelization with the cavity problem for

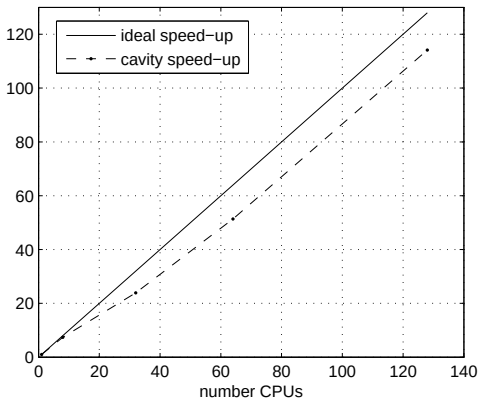

Figure 9: Speed-up of the overall CPU time of the convective step needed to reach $T=500$ for $Re=5000$.

$Re=5000$. The figure refers to the overall CPU time for the convective step, including the MPI communications, needed in the whole computation to reach $T=500$. As we can see, even for more than 100 CPUs the speed up is close to the ideal one.

4 Conclusions

We have conducted a sequence of numerical experiments employing the Navier-Stokes equations and the convection-diffusion equations of a scalar quantity to test and compare nodal and modal semi-Lagrangian schemes as well as Lagrange-Galerkin schemes in the framework of high order hp finite elements. According to the results of these experiments, both semi-Lagrangian and Lagrange-Galerkin hp finite element methods are able to produce very good results in convection-dominated problems.

Regarding the behaviour, in terms of CPU time and accuracy, of semi-Lagrangian schemes versus Lagrange-Galerkin schemes we have noticed that when the degree of the polynomials of the hp finite elements are higher than 2 Lagrange-Galerkin are better than semi-Lagrangian schemes because they are more accurate (although the asymptotic rate of convergence is the same for both schemes) and faster, specially when the degree of the polynomials are larger or equal than 8. We also have tested Lagrange-Galerkin p-adaptive schemes and the conclusion is that p-adaptive Lagrange-Galerkin schemes are more efficient than the conventional Lagrange-Galerkin ones. Additionally, we have also conducted experiments to test the efficiency of parallelization of the schemes presented in the paper. Our results show that the parallelization is very efficient, giving a speed-up quasi-optimal in the convective step even with more than 100 processors.

Acknowledgments

This research has been partially funded by grant CGL2007-66440-C04-01 from Ministerio de Educación y Ciencia de España.

References

[1] R. Bermejo and J. Carpio, A semi-Lagrangian-Galerkin projection scheme for convection equations, IMA J. Numer. Anal., 30(3) (2010), 799–831.

[2] K. Boukir, Y. Maday, B. Métivet, and E. Razafindrakoto, A high-order characteristics/finite element method for the incompressible Navier-Stokes equations, Int. J. Numer. Methods. Fluids., 25(12) (1997), 1421–1454.

[3] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, Uzawa type algorithms for nonsymmetric saddle point problems, Math. Comput., 69(230) (2000), 667–689.

[4] E. J. Dean and R. Glowinski, On some finite element methods for the numerical simulation of incompressible viscous flow, in Incompressible Computational Fluid Dynamics, pages 17–65, Cambridge University Press, 1993.

[5] J. Douglas Jr. and T. F. Russell, Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures, SIAM J. Numer. Anal., 19(5) (1982), 871–885.

[6] E. Erturk, T. C. Corke, and C. Gokcol, Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers, Int. J. Numer. Meth. Fluids., 48(7) (2005), 747–774.

[7] R. E. Ewing and T. F. Russell, Multistep Galerkin methods along characteristics for convection-diffusion problems, Advances in Computer Methods for Partial Differential Equations-IV, pages 28–36, 1981.

[8] U. Ghia, N. Ghia, and C. T. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, J. Comput. Phys., 48 (1982), 387–411.

[9] F. X. Giraldo, The Lagrange-Galerkin spectral element method on unstructured quadrilateral grids, J. Comput. Phys., 147(1) (1998), 114–146.

[10] J. L. Guermond, P. Minev, and J. Shen, An overview of projection methods for incompressible flows, Comput. Methods. Appl. Mech. Engrg., 195(44-47) (2006), 6011–6045.

[11] J. L. Guermond and J. Shen, A new class of truly consistent splitting schemes for incompressible flows, J. Comput. Phys., 192(1) (2003), 262–276.

[12] J. L. Guermond and J. Shen, Velocity-correction projection methods for incompressible flows, SIAM J. Numer. Anal., 41(1) (2003), 112–134 (electronic).

[13] G. E. Karniadakis, M. Israeli, and S. A. Orszag, High-order splitting methods for the incompressible Navier-Stokes equations, J. Comput. Phys., 97(2) (1991), 414–443.

[14] G. E. Karniadakis and S. J. Sherwin, Spectral/hp Element Methods for CFD, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 1999.

[15] K. W. Morton, A. Priestley, and E. Süli, Stability of the Lagrange-Galerkin method with nonexact integration, RAIRO Modél. Math. Anal. Numér., 22(4) (1988), 625–653.

[16] O. Pironneau, On the transport-diffusion algorithm and its applications to the Navier-Stokes equations, Numer. Math., 38(3) (1981/82), 309–332.

[17] C. Temperton and A. Staniforth, An efficient two-time-level semi-lagrangian semi-implicit integration scheme, Q.J.R. Meteorol. Soc., 113(477) (1987), 1025–1039.

[18] P. Šolín, K. Segeth, and I. Doležel, Higher-Order Finite Element Methods, Studies in Advanced Mathematics, Chapman & Hall/CRC, Boca Raton, FL, 2004.

[19] D. Xiu and G. E. Karniadakis, A semi-Lagrangian high-order method for Navier-Stokes equations, J. Comput. Phys., 172(2) (2001), 658–684.