Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 3865-3881 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

EVALUATION OF THE GMRES METHOD APPLIED ON THE STABILIZED DYNAMIC DIFFUSION METHOD

Isaac P. Santos[a], **Lucia Catabriga**[a] **and Regina C. Almeida** [b]

[a]*Laboratório de Computação de Alto Desempenho, Universidade Federal do Espírito Santo, isaac,luciac@lcad.inf.ufes,br, http://www.lcad.inf.ufes.br*

[b]*Department of Computational Mechanics, Laboratório Nacional de Computação Científica - LNCC/MCT, rcca@lncc.br, http://www.lncc.br*

Keywords: Dynamic diffusion, Advection-diffusion equations, GMRES(k).

Abstract. In this work, the computational performance of the Dynamic Diffusion method is addressed when the GMRES method is used to solve the resulting linear system. The DD method, introduced by Arruda, Almeida and Dutra do Carmo (Dynamic Diffusion Formulations for Advection Dominated Transport Problems, to appear), is a two-scale model for transport problem, obtained by adding to the Galerkin formulation a nonlinear dissipative operator acting isotropically in all scales. The amount of the artificial diffusion is determined by the solution of the resolved scale at the element level yielding a self adaptive free parameter method. The discrete problem is solved by using the well known element-by-element and edge-based storage local data schemes to optimize the matrix-vector product in the GMRES algorithm. Comparisons between these two storage schemes are addressed for a variety of numerical experiments covering advection dominated regimes. Our experiments have shown that the edge-based storage scheme leads to less CPU time and, since the resulting matrix is not well conditioned for some problems, the GMRES algorithm might fail for some dimensions of restart vectors.

1 INTRODUCTION

The aim of this work is to address and analyze the computational performance of the Dynamic Diffusion method when the GMRES method is used to solve the resulting linear system. The DD is a finite element two-scale model for transport problems, obtained by adding to the Galerkin formulation a nonlinear dissipative operator acting isotropically in all scales. It was developed based on the general concept of eddy viscosity methods in which a dissipation mechanism is introduced either on all scales or on the subgrid scale. The methods developed in Guermond (2001); Kaya and Rivière (2005) are, for example, subgrid eddy viscosity methods since they introduce eddy viscosity models only onto the small scales. They are similar in spirit to the spectral viscosity technique introduced in Tadmor (1989) to approximate nonlinear conservation equations by means of spectral methods (Ern and Guermond (2004)). However, like most stabilized methods for transport problems, the methods developed in Guermond (2001); Kaya and Rivière (2005) require tunable parameters whose selection is a tricky task for actual problems.

The artificial diffusion introduced by the DD method, on the other hand, is dynamically determined by imposing some restrictions on the resolved scale solution at the macro element level in the same spirit of the methods presented in Santos and Almeida (2007); Arruda et al. (2010). These two nonlinear subgrid methods were attempts to avoid user-defined coefficients by means of subgrid diffusion operators which are nonlinear local functionals of the resolved scale solution. Following the multiscale concept of splitting the variable of interest into a resolved coarse scale and an unresolved subgrid scale, the definition of the subgrid diffusion relies on the assumption that the velocity field may also be decomposed into these two scales and the subgrid velocity field is then used to determine the amount of diffusion that is able to dissipate the kinetic energy at the smallest scales. These free parameter methods present good stability and convergence properties for singular perturbed transport problems, although oscillations still remain in some situations, mainly when the velocity field is not constant and when external layers are present. To overcome these drawbacks, the DD method was developed by Arruda, Almeida and Dutra do Carmo (to appear). The underline idea still is to control the resolved scale solution so that the spurious modes are confined to the subgrid scales. This is done by adding to the Galerkin formulation an artificial diffusion nonlinear operator that isotropically acts on all scales. The consistency, stability and convergence properties of the resulting methodology relies on the definition of the artificial diffusion. Like in Santos and Almeida (2007); Arruda et al. (2010), the artificial diffusion is designed through the definition of the subgrid velocity field. In order to reduce the computational cost typical of two-scale methods, the small scale space is defined using bubble functions whose degrees of freedom are condensed onto the resolved scale degrees of freedom. Moreover, the final discrete setting is implemented e evaluated into local data structure framework, by using element-by-element and edge-based storages (Coutinho et al. (2001); Catabriga and Coutinho (2002)). They optimize the matrix-vector products of the GMRES algorithm (Saad (1995)) that is ultimately used to solve the resulting algebraic linear systems. These two storage strategies are compared for a variety of numerical experiments covering the advection dominated regime.

The remainder of this work is organized as follows. Section 2 briefly addresses the transport problem and introduces the DD method. The element-based and edge-based structures are promptly described in section 3. Numerical experiments are conducted in Section 4 to show the behavior of the proposed methodologies for a variety of transport problems focusing in advection dominated regime. Section 5 concludes this paper.

2 THE DYNAMIC DIFFUSION STABILIZATION METHOD

Consider the steady scalar advection-diffusion-reaction problem whose solution u satisfies

$$\begin{aligned} -\epsilon\Delta u + \boldsymbol{\beta}\cdot\boldsymbol{\nabla} u + \sigma u &= f \quad \text{in } \Omega; & (1)\\ u &= g \quad \text{on } \Gamma^D; & (2)\\ \epsilon\boldsymbol{\nabla} u\cdot\boldsymbol{n} &= q \quad \text{on } \Gamma^N, & (3)\end{aligned}$$

where $\Omega \subset \mathbb{R}^2$ (the extension to $\mathbb{R}^n$ is straightforward) is an open bounded domain with a Lipschitz boundary Γ. As usual, Γ is divided into two parts, denoted by Γ^D e Γ^N, such that $\Gamma^D \cap \Gamma^N = \emptyset$. Γ^D and Γ^N are the part of the boundary on which Dirichlet boundary conditions and Neumann boundary conditions are prescribed, respectively. $\boldsymbol{\beta}$ is the divergence free velocity field, σ is the reaction coefficient, $0 < \epsilon \ll 1$ is the (constant) diffusion coefficient, f is the source term, and $\boldsymbol{n}$ is the unit outward normal vector to the boundary Γ. The outflow part of Γ is defined as $\Gamma_+ = \{x \in \Gamma : \boldsymbol{\beta}(x)\cdot\boldsymbol{n} < 0\}$. It is assumed that $\boldsymbol{\beta} \in W^{1,\infty}(\Omega)$, $\sigma \in L^\infty(\Omega)$, $g \in H^{1/2}(\Gamma^D)$, $q \in H^{-1/2}(\Gamma^N)$ and $f \in L^2(\Omega)$.

Let $\mathcal{T}_h$ be a triangulation of the domain Ω with elements T. A finite element formulation to (1)-(3) can be written as:

$$\begin{aligned} &\text{Find } u_h \in U_h(\Omega) \text{ such that} & (4)\\ &a(u_h, v_h) = F(v_h), \quad \forall v_h \in V_h(\Omega), & (5)\end{aligned}$$

where U_h and V_h are finite dimensional subspaces of $U = H^1(\Omega)$ and $V = H^1_0(\Omega)$, respectively, h is the characteristic element length and

$$\begin{aligned} a(u_h, v_h) &= \int_\Omega (\epsilon\boldsymbol{\nabla} u_h\cdot\boldsymbol{\nabla} v_h + \boldsymbol{\beta}\cdot\boldsymbol{\nabla} u_h v_h + \sigma u_h v_h)d\Omega, & (6)\\ F(v_h) &= \int_\Omega f v_h d\Omega + \int_{\Gamma^N} q v_h d\Gamma. & (7)\end{aligned}$$

One obtains the standard Galerkin formulation by using the same function space for both U_h and V_h (except on Γ^D).

The standard Galerkin finite element method is not optimal for solving advection dominated problems (Donea and Huerta (2003)). Stable solutions can be obtained using stabilized or multiscale methods, as SUPG, GLS, RFB, VMS and subgrid viscosity techniques (Brooks and Hughes (1982); Hughes et al. (1989); Brezzi et al. (2003); Hughes et al. (2004); Guermond (2001)). Here, we use a two-scale method originally by Arruda, Almeida and Dutra do Carmo (to appear).

Consider the following decomposition of the approximation space

$$U_h = U_L \oplus U_B \ ,$$

where

$$U_L = \{u \in H^1_0(\Omega) \text{ such that } u|_T \in \mathbb{P}_1, \forall T \in \mathcal{T}_h\},$$

with $\mathbb{P}_1$ denoting the space of linear polynomials and the *bubble* space (see Figure 1)

$$U_B = \{v \text{ such that } v|_T \in H^1_0(T), \forall T \in \mathcal{T}_h\}.$$

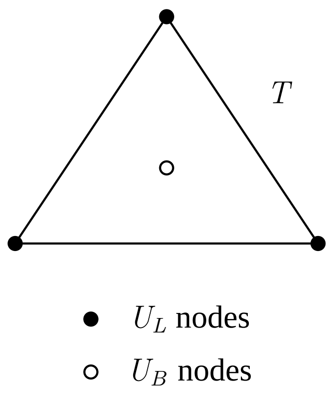


Figure 1: Schematic representation of the *bubble* space

The Dynamic Diffusion method, proposed by Arruda, Almeida and Dutra do Carmo (to appear), is given by

$$\text{Find } u_h = u_L + u_b \in U_h \text{ with } u_L \in U_L, u_b \in U_B \text{ such that}$$
$$a(u_h, v_h) + \sum_T \int_T \xi(u_L) \nabla u_h \cdot \nabla v_h d\Omega = F(v_h), \quad \forall v_h \in U_h, \tag{8}$$

where

$$\xi(u_L) = \begin{cases} \frac{1}{2}\mu(h)\frac{|R(u_L)|}{|\nabla u_L|}, \text{ if } |\nabla u_L| > tol; \\ 0, \quad \text{otherwise,} \end{cases} \tag{9}$$

with $\mu(h)$ stands for the subgrid characteristic length and

$$R(u_L) = \boldsymbol{\beta} \cdot \nabla u_L + \sigma u_L - f \text{ on } T.$$

When $|\nabla u_L|$ is small, there is no need of extra stabilization term since the Galerkin solution is stable enough. The amount of artificial diffusion term provided in (8) dynamically adapts to this situation, yielding a self adaptive method. The tolerance tol in (9) must only be small enough to avoid division by zero. Here we set $tol = 10^{-6}$. This choice may lead to some convergence difficulties, which are reported in this work. A smaller tolerance, such as $tol = 10^{-10}$, overcomes the convergence difficulties.

The method is solved by using an iterative procedure for which the initial solution is obtained using $\xi(u_L) = \sqrt{measT}$. The iterative process is defined by: Given u_h^{n-1}, we find u_h^n satisfying

$$a(u_h^n, v_h) + \int_T \sum_T \xi(u_L^{n-1}) \nabla u_h^n \cdot \nabla v_h d\Omega = (f, v_h), \qquad \forall v_h \in U_h, \tag{10}$$

with $\xi(u_L^{n-1}) = c_b^n \mu(h)$, where

$$c_b^n = \begin{cases} \frac{1}{2}\frac{|R(u_L^{n-1})|}{|\nabla u_L^{n-1}|}, \text{ if } |\nabla u_L^{n-1}| > tol; \\ 0, \quad \text{otherwise,} \end{cases} \tag{11}$$

and

$$\mu(h) = \begin{cases} 2\sqrt{measT}, \text{ if } T \cap \Gamma_+ \neq \emptyset; \\ \sqrt{measT}, \quad \text{otherwise.} \end{cases} \tag{12}$$

To improve convergence, the following average rule to determine c_b^n is used (Santos and Almeida (2007)):

$$c_b^n = \frac{1}{2}(c_b^n + c_b^{n-1}).$$

Since u_L is linear on T, the Green's theorem yields

$$\int_T \xi(u_L^{n-1})\boldsymbol{\nabla} u_h^n \cdot \boldsymbol{\nabla} v_h d\Omega = \int_T \xi(u_L^{n-1})\boldsymbol{\nabla} u_L^n \cdot \boldsymbol{\nabla} v_L d\Omega + \int_T \xi(u_L^{n-1})\boldsymbol{\nabla} u_b^n \cdot \boldsymbol{\nabla} v_b d\Omega.$$

Testing (8) using $v_L \in U_L$ and $v_b \in U_B$ and using the previous equality we obtain

$$a(u_L^n, v_L) + \sum_T \int_T \xi(u_L^{n-1})\boldsymbol{\nabla} u_L^n \cdot \boldsymbol{\nabla} v_L d\Omega + a(u_b^n, v_L) = F(v_L), \forall v_L \in U_L; \tag{13}$$

$$a(u_L^n, v_b) + a(u_b^n, v_b) + \sum_T \int_T \xi(u_L^{n-1})\boldsymbol{\nabla} u_b^n \cdot \boldsymbol{\nabla} v_b d\Omega = F(v_b), \forall v_b \in U_B. \tag{14}$$

The local equation system associated with each element T may be partitioned as

$$\begin{bmatrix} A_{LL} & A_{LB} \\ A_{BL} & A_{BB} \end{bmatrix} \begin{bmatrix} u_{L;T} \\ u_{b;T} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_L \\ \mathcal{F}_B \end{bmatrix} \tag{15}$$

where the local matrices are given by

$$A_{LL} : \int_T ((\epsilon + \xi(u_L^{n-1}))\boldsymbol{\nabla} u_L \cdot \boldsymbol{\nabla} v_L + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} u_L v_L + \sigma u_L v_L) d\Omega \tag{16}$$

$$A_{LB} : \int_T (\epsilon \boldsymbol{\nabla} u_L \cdot \boldsymbol{\nabla} v_b + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} u_L v_b + \sigma u_L v_b) d\Omega \tag{17}$$

$$A_{BL} : \int_T (\epsilon \boldsymbol{\nabla} u_b \cdot \boldsymbol{\nabla} v_L + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} u_b v_L + \sigma u_b v_L) d\Omega \tag{18}$$

$$A_{BB} : \int_T ((\epsilon + \xi(u_L^{n-1}))\boldsymbol{\nabla} u_b \cdot \boldsymbol{\nabla} v_b + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} u_b v_b + \sigma u_b v_b) d\Omega \tag{19}$$

$$\mathcal{F}_L : \int_T f v_L d\Omega \tag{20}$$

$$\mathcal{F}_B : \int_T f v_b d\Omega \tag{21}$$

Performing a static condensation of the unknowns u_b at each element, the local problem becomes

$$\mathcal{A}_T \, u_{L;T} = \mathcal{F}_T,$$

where $\mathcal{A}_T = A_{LL} - A_{LB}A_{BB}^{-1}A_{BL}$ and $\mathcal{F}_T = \mathcal{F}_L - A_{LB}A_{BB}^{-1}\mathcal{F}_B$.

After assembling the contributions of all $T \in \mathcal{T}_h$, the following new global linear system is obtained

$$\mathcal{A}\mathcal{U} = \mathcal{F}, \tag{22}$$

where

$$\mathcal{A} = \mathop{\mathbf{A}}_{T=1}^{nel} \mathcal{A}_T; \qquad \mathcal{F} = \mathop{\mathbf{A}}_{T=1}^{nel} \mathcal{F}_T; \qquad \mathcal{U} = \mathop{\mathbf{A}}_{T=1}^{nel} u_{L;T}. \tag{23}$$

In this expression, nel is the total number of elements of the mesh $\mathcal{T}_h$.

3 ELEMENT-BASED AND EDGE-BASED STRUCTURES

Element-based and Edge-based structures have been extensively used in finite element implementations, resulting in considerable improvements comparing to standard implementations. The success of this solution strategy requires an efficient implementation of matrix-vector products and the choice of a suitable pre-conditioner. Generally, the edge-based data structure reduces processing time and requires around one half of the storage area to hold the coefficient matrix when compared to an enhanced element-based implementation (Coutinho et al. (2001); Catabriga and Coutinho (2002)). We perform an implementation of the DD method using element-based and edge-based implementations.

Figure 2 shows a set of two adjacent elements, e and f, and the associated edge s. The conventional finite element data structure associated with each triangle is its connectivity (i.e., the mesh nodes I, J and K for element e, and the mesh nodes I, L and J for element f). In the edge-based data structure, each edge s is associated with the adjacent elements e and f and, thus, with the nodes I, J, K and L as shown in Figure 2. So, each element matrix can

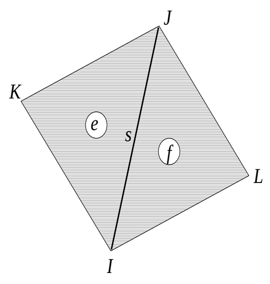


Figure 2: Elements adjacent to edge *s*, formed by nodes *I* e *J*.

be disassembled into its contributions, i.e., three edges, s, $s+1$ and $s+2$, with, respectively, connectivities IJ, JK and KI, that is,

$$\underbrace{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix}}_{\text{element } e} = \underbrace{\begin{bmatrix} \times & \times & 0 \\ \times & \times & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{edge } s} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}}_{\text{edge } s+1} + \underbrace{\begin{bmatrix} \times & 0 & \times \\ 0 & 0 & 0 \\ \times & 0 & \times \end{bmatrix}}_{\text{edge } s+2}, \tag{24}$$

where $\bullet$ and $\times$ are matrix coefficients defined from (22). Thus, all the contributions belonging to edge s will be present in the adjacent elements e and f.

The resulting edge matrix is the sum of the corresponding sub-element matrices containing all the contributions of nodes I and J,

$$\underbrace{\begin{bmatrix} \circ & \circ \\ \circ & \circ \end{bmatrix}}_{\text{edge } s} = \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{\text{element } e} + \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{\text{element } f}. \tag{25}$$

Considering a conventional element wise description of a given finite element mesh, the topological information can be manipulated to generate a new edge-based mesh description.

Thus, the assembled global matrices given in equation (23) may be written now as,

$$\mathcal{A} = \underset{S=1}{\overset{nedges}{\mathbf{A}}} \mathcal{A}_S; \qquad \mathcal{F} = \underset{S=1}{\overset{nedges}{\mathbf{A}}} \mathcal{F}_S; \qquad \mathcal{U} = \underset{S=1}{\overset{nedges}{\mathbf{A}}} u_{L;S}. \tag{26}$$

where $nedges$ is the total number of edges of the mesh $\mathcal{T}_H$. The edge matrix $\mathcal{A}_S$ is obtained from the contributions of all the element matrices $\mathcal{A}_T$ (Eq. (23)) that share the edge S.

In the element by element (EBE) implementation strategy, the coefficients of the global matrix are stored in each macro element matrix as defined by (23). The global matrix $\mathcal{A}$ is stored in a compact form of size $nel \times 9$. On the other hand, in the edge-based (EDS) strategy the coefficients of the global matrix, defined by (26), are also stored in a compact form of size $nedges \times 4$.

4 NUMERICAL EXPERIMENTS

In this section we present numerical results in order to evaluate the computational performance of the Dynamic Diffusion method when the GMRES method is used to solve the resulting linear system. In the following, $Kmax$ is the number of Krylov vectors to restart the GMRES method, $Iter_{DD}$ is the maximum number of iterations for the DD method, $Iter_{GMRES}$ is the maximum number of GMRES iterations and CPU_{time} is the computational time. We also use tol_{DD} and tol_{GMRES} to indicate the threshold tolerances of the DD and GMRES iterative processes.

4.1 Internal and external layers

This example simulates a two-dimensional advection dominated advection-diffusion problem with $\epsilon = 10^{-12}$, $f = \sigma = 0$ and $\boldsymbol{\beta} = (0.5, 1)$ (example 4.1-a) and $\boldsymbol{\beta} = (1, 1)$ (example 4.1-b). The Dirichlet boundary condition are

$$u(0, y) = u(1, y) = u(x, 1) = 0$$

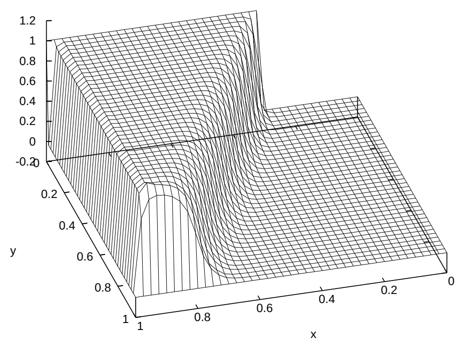
and

$$u(x, 0) = \begin{cases} 1, & x \le 0.3\,; \\ 0, & x > 0.3. \end{cases}$$

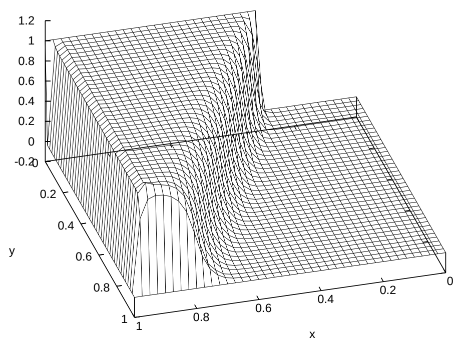
These conditions yield a solution with an internal layer in the direction of the velocity field starting at $(0.3, 0.)$ and an exponential external layer at the outflow.

Figure 3 shows the EBE/EDS-based approximate solution for $\boldsymbol{\beta} = (0.5, 1)$, using a mesh with 40×40 cells. Both solutions present some smearing behavior at both internal and external layers. Figure 4 depicts the maximum (*max*) and minimum (*min*) values of the main diagonal of global matrix at each non linear iteration of the iterative process solution, using GMRES(60) and a 80×80 mesh. The behavior of these values during the iterative procedure may help to identify an eventual ill-conditioning of the matrix. Since the errors associated with computations become significantly magnified due to ill-conditioning and the evaluation of the condition of a matrix is usually computationally expensive, an increase of the difference between *max* and *min* during the iterative procedure may indicate that the matrix is not well conditioned. Hence, this heuristic measure may be useful in understanding unexpected solution behaviors. In this case, Figure 4 does not indicate ill-conditioning of the global matrix.

Table 1 presents the computational performance of the element-based (EBE) and edge-based (EDS) data structures, respectively, using a 80×80 mesh. The CPU_{time} spent for solving the problem by using the edge-based approach is approximately 22% smaller than the element-based one.



(a)



(b)

Figure 3: Example 4.1-a) - EBE (a) and EDS (b) solutions - 40×40 mesh, GMRES(60), $tol_{GMRES} = 10^{-7}$, $tol_{DD} = 10^{-2}$ – Internal and external layers for $\boldsymbol{\beta} = (0.5, 1)$.

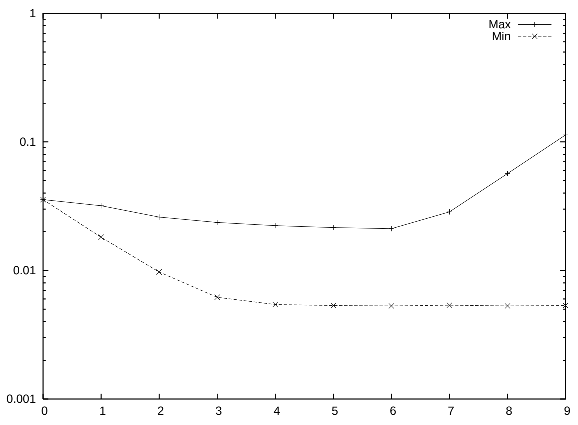


Figure 4: Example 4.1-a) $\log max$ and $\log min$ *versus* number of iterations – 80×80 mesh – Internal and external layers for $\boldsymbol{\beta} = (0.5, 1)$.

80×80 mesh						
	Element			Edge		
Kmax	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}
10	9	2984	07.784	9	3005	07.597
30	9	4074	10.545	9	4105	09.079
60	9	5155	14.461	9	5243	11.949
80	9	5448	16.426	9	5567	13.525
100	9	5011	16.504	9	5016	13.915

Table 1: Computational Efforts - EBE and EDS structures – Internal and external layers for $\beta = (0.5, 1)$.

Figure 5 shows the EBE/EDS-based approximate solution with $\boldsymbol{\beta} = (1, 1)$ obtained by using a mesh with 40 divisions in each direction. The internal layer is well represented while a negligible oscillation appears in the neighborhood of the outflow. Figure 6 depicts the maximum and minimum values of the main diagonal of global matrix, using GMRES(80) and a 80×80 mesh. We may notice that the difference between these values increases after the sixth iteration indicating an ill-conditioning of the global matrix, which is expected to affect the convergence of the non linear iterative process. This indeed happens so that the DD iterative process does not converge for $Kmax = 10, 30, 60, 80$, yielding nonphysical solution by the end of the process. By increasing $Kmax$, the DD iterative process recovers its convergence property but that can be lost again by refining the mesh. To overcome this difficulty, a pre-conditioning strategy may be introduced to prevent ill-conditioned matrices. We also notice that the lack of convergence is related to the choice of tol (equation (9)), which has to be close to the machine epsilon.

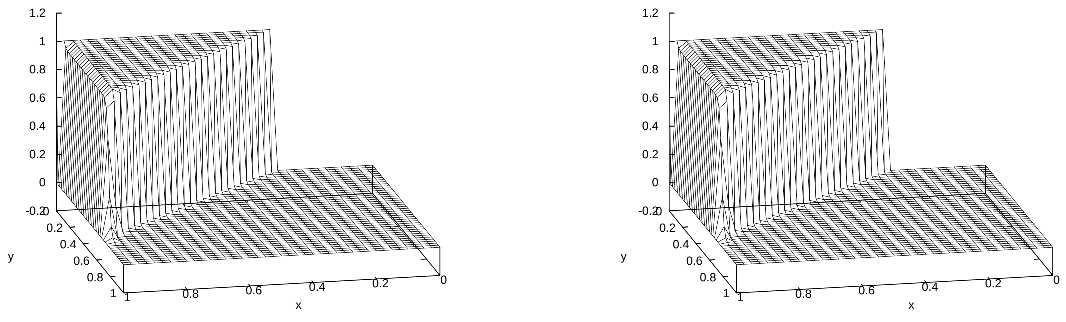


Figure 5: Example 4.1-b) - EBE (a) and EDS (b) solutions - 40×40 mesh, GMRES(100), $tol_{GMRES} = 10^{-7}$, $tol_{DD} = 10^{-2}$ – Internal and external layers for $\beta = (1, 1)$.

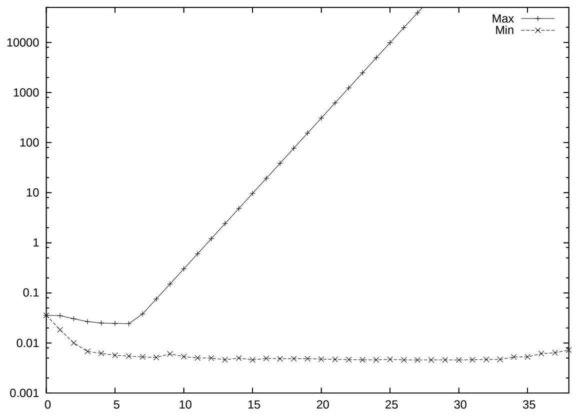


Figure 6: Example 4.1-b) $\log max$ and $\log min$ *versus* number of iterations – 80×80 mesh – Internal and external layers for $\beta = (1, 1)$.

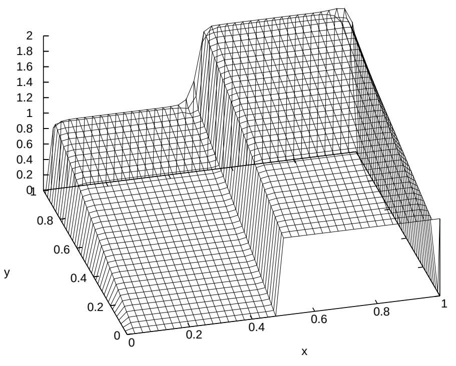
4.2 Internal and external layers with source term

This example simulates a two-dimensional advection dominated advection-diffusion problem with $\epsilon = 10^{-6}$, $\boldsymbol{\beta} = (1, 0)$, $\sigma = 0$ and a constant source term $f = 1$. The Dirichlet boundary conditions are given by

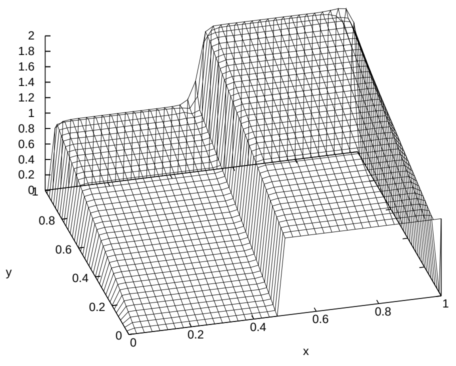
$$u(0, y) = u(1, y) = u(x, 1) = 0 \quad \text{and} \quad u(x, 0) = \begin{cases} 0, & x < 0.5\,; \\ 1, & x \geq 0.5. \end{cases}$$

The exact solution of this problem possesses parabolic layers at $x = 0$ and $x = 1$ and an exponential layer at $y = 1$. Also, an internal layer emanates from the discontinuity of the inflow boundary data at $x = 0.5$.

The element-based and edge-based solutions are shown in Figure 7. The approximate solutions are practically free from numerical oscillations, except in the neighborhood of the exponential layer, where a non-monotone behavior is presented. According to Figure 8 the global matrix seems well-conditioned. Table 2 present the computational performance considering both data structures with a tolerance of the GMRES method of 10^{-7}. The non linear iterative process converges for all values of $Kmax$, except for $Kmax = 10$. As expected, the edge-based approach is approximately 40% faster than the element-based one.



(a)



(b)

Figure 7: Example 4.2 - Element (a) and Edge (b) solutions - 40×40 mesh, GMRES(60), $tol_{GMRES} = 10^{-7}$, $tol_{DD} = 10^{-2}$ – Internal and external layers with source term.

4.3 Variable flow field with internal and external layers

In this problem, the domain is given by $\Omega = (-1, 1) \times (0, 1)$ and the velocity field is

$$\boldsymbol{\beta} = (2y(1 - x^2), -2x(1 - y^2)).$$

The inflow and outflow boundary are the intervals

$$\{(x, 0)| - 1 \leq x \leq 0\} \text{ and } \{(x, 0)|0 < x < 1\},$$

respectively. At the outflow the following natural boundary condition is prescribed

$$\frac{\partial u(x, 0)}{\partial \boldsymbol{n}} = 0, \qquad 0 < x < 1.$$

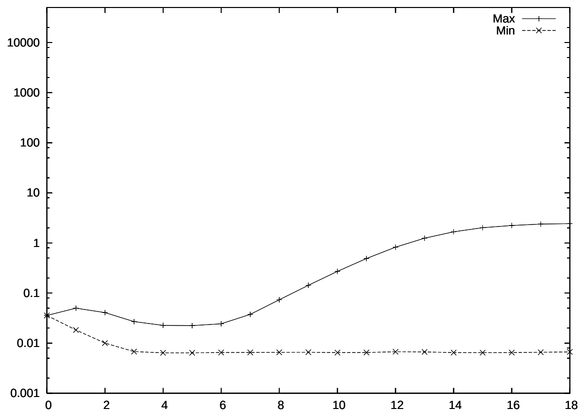


Figure 8: Example 4.2 - $\log max$ and $\log min$ *versus* iterations number, using a 80×80 mesh and GMRES(60) – Internal and external layers with source term.

80×80 mesh						
	Element			Edge		
Kmax	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}
10	50	40412	92.539	50	40415	42.026
30	18	25068	57.907	18	25081	30.513
60	18	21479	56.971	18	21510	33.758
80	18	19347	56.238	18	19269	35.563
100	18	18735	58.624	18	18733	39.031
150	18	13656	51.464	18	13656	38.095

Table 2: Computational Efforts – Internal and external layers with source term - $\beta = (1, 0)$, $tol_{GMRES} = 10^{-7}$.

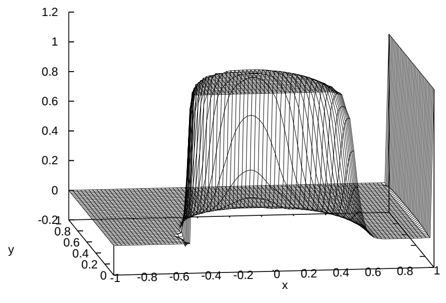
Dirichlet boundary conditions are set on the remainder of boundary Γ. At the inflow, there is a discontinuity given by

$$u(x,0) = \begin{cases} 0, & -1 \leq x < -0.5; \\ 1, & -0.5 \leq x \leq 0.0. \end{cases}$$

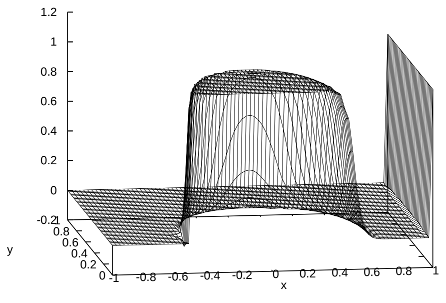
On the remaining boundary, we set $u = 0$ at $x = -1$, $u = 0$ at $y = 1$ and $u = 1$ at $x = 1$. The inflow discontinuity is convected by the circular flow towards the outflow boundary. Moreover, an external layer appears at $x = 1$. Figure 9 shows the solutions obtained using the element-based and edge-based data structures, respectively. The approximate solutions are well represented. In this example, in the neighborhood of the external layer in $x = 1$, we used the following new subgrid characteristic length

$$\mu(h) = \frac{|\boldsymbol{\beta}|}{|\boldsymbol{b}_T|}, \quad \boldsymbol{b} = (\boldsymbol{\nabla}_{N_T})\,\boldsymbol{\beta},$$

where the tensor $\boldsymbol{\nabla}_{N_T}$ is the Jacobian of the coordinate system and N_T denotes the element local coordinates. Figure 10 depicts the maximum and minimum values of the main diagonal of global matrix, obtained by using GMRES(80) and a 160×80 mesh. In this case, although the difference between these values increases with the number of iterations, such difference is not as large as that one observed in Example 4.1 when $\boldsymbol{\beta} = (1,1)$ is used. Table 3 presents the computational performance considering both data structures, using a 80×40 mesh. The iterative process converges in 13 iterations. The edge-based approach is approximately 30% faster than the element-based one. Table 4 shows the results obtained by using a 160×80 mesh. The convergence of the iterative process is obtained for all tested values of $Kmax$. As expected, the edge-based approach is again approximately 30% faster than the element-based one.



(a)



(b)

Figure 9: Example 4.3 - Element (a) and Edge (b) solutions - 80×40 mesh, GMRES(60), $tol_{GMRES} = 10^{-7}$, $tol_{DD} = 10^{-2}$ – Variable flow field with internal and external layers.

4.4 The Burgers equation

In this example we consider the Burgers equation given by

$$-\epsilon \Delta u + u \frac{\partial u}{\partial y} = 0$$

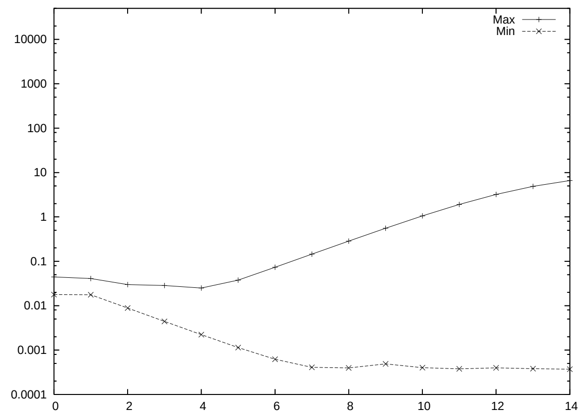


Figure 10: Example 4.3 - $\log max$ and $\log min$ *versus* number of iterations, using a 160×80 mesh and GMRES(60) – Variable flow field with internal and external layers.

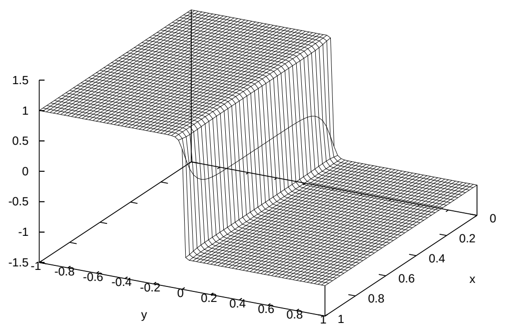
80×40 mesh						
	Element			Edge		
Kmax	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}
10	13	8998	11.044	13	8997	5.413
30	13	16455	18.954	13	16458	9.672
60	13	16971	21.606	13	16957	12.339
80	13	16955	23.774	13	17166	14.211
100	13	16695	25.131	13	16389	15.366
150	13	15198	27.190	13	15157	18.954

Table 3: Computational Efforts – Variable flow field with internal and external layers, $tol_{GMRES} = 10^{-7}$.

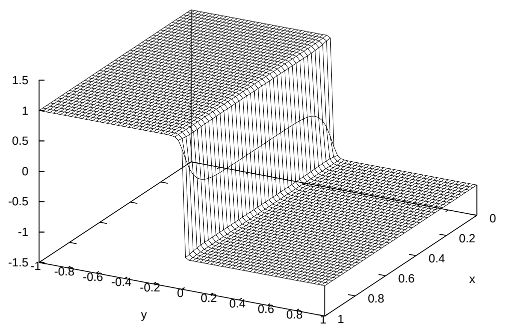
160×80 mesh						
	Element			Edge		
Kmax	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DD}$	$Iter_{GMRES}$	CPU_{time}
10	41	36713	169.306	38	34064	76.689
30	15	28784	137.046	17	34638	87.781
60	14	37069	194.859	14	37217	119.667
80	14	39355	221.426	14	39287	138.871
100	14	39688	240.489	14	40112	162.739
150	14	40009	290.768	14	39873	204.859

Table 4: Computational Efforts – Variable flow field with internal and external layers, $tol_{GMRES} = 10^{-7}$.

and defined in $\Omega = (0,1) \times (-1,1)$, with $\epsilon = 10^{-4}$. The Dirichlet boundary conditions are prescribed as follows: $u = 1$ at $y = -1$ and $u = -1$ at $y = 1$. Homogeneous Neumann boundary conditions are prescribed at $x = 0$ and $x = 1$. The initial solution of the iterative process is characterized by two fronts at $y = -0.2$ and $y = 0.2$, which eventually collapse at the end of the iterative procedure. The approximate solutions are shown in Figure 11. We can observe that the discontinuity is accurately represented in both solutions. According to Figure 12, there is an indication that the global matrix is not ill-conditioned. The computational performance of the DD method was evaluated considering the element-based and edge-based data structures. The numerical results are presented in Table 5 and Table 6. Using a 40×80 mesh, the iterative process converges in 38 iterations, while using a 80×160 mesh the convergence is obtained in 39 iterations. This amounts to a computational gain of about 20% when the edge-based structured is used.



(a)



(b)

Figure 11: Example 4.4 - Element (a) and Edge (b) solutions - 80×40 mesh, GMRES(60), $tol_{GMRES} = 10^{-7}$, $tol_{DV} = 10^{-2}$ – The Burgers equation.

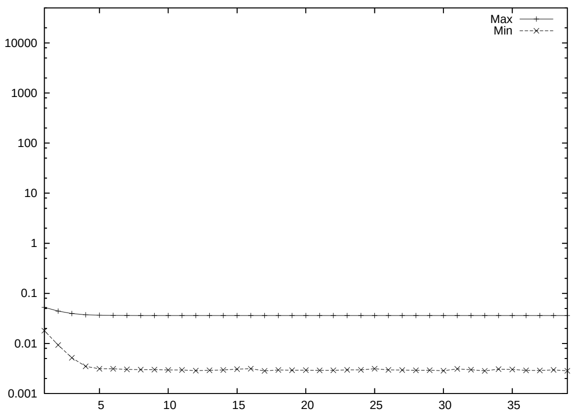


Figure 12: Example 4.4 - $\log max$ and $\log min$ *versus* number of iterations, using a 80×160 mesh and GMRES(60) – The Burgers equation.

40×80 mesh						
	Element			Edge		
Kmax	$Iter_{DV}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DV}$	$Iter_{GMRES}$	CPU_{time}
10	38	8544	12.105	38	8776	7.269
30	38	10300	14.258	38	10274	8.767
60	38	11193	16.848	38	11234	11.029
80	38	7262	12.792	38	7199	9.266
100	38	6297	12.292	38	6379	9.250
150	38	6143	13.525	38	6262	10.795

Table 5: Computational Efforts – The Burgers Equation, $tol_{GMRES} = 10^{-7}$.

80×160 mesh						
	Element			Edge		
Kmax	$Iter_{DV}$	$Iter_{GMRES}$	CPU_{time}	$Iter_{DV}$	$Iter_{GMRES}$	CPU_{time}
10	39	13028	69.856	39	12951	48.126
30	39	17398	91.213	39	17573	61.573
60	39	20992	121.960	39	21165	84.177
80	39	24959	154.252	39	24983	106.236
100	39	23555	158.901	39	5637	18.627
150	39	5117	23.197	39	5250	20.670

Table 6: Computational Efforts – The Burgers Equation, $tol_{GMRES} = 10^{-7}$.

5 CONCLUSIONS

In this work, the computational performance of the Dynamic Diffusion method was addressed when the GMRES method is used to solve the resulting linear system. The discrete problem was solved by using the well known element-by-element and edge-based storage local data schemes to optimize the matrix-vector product in the GMRES algorithm. We addressed comparisons between these two storage schemes for a variety of numerical experiments covering advection dominated regimes. Our experiments have shown that in almost all cases the edge-based storage scheme leads to less CPU time and, since the resulting matrix is not well conditioned for some problems, the GMRES algorithm might fail for some dimensions of restart vectors.

ACKNOWLEDGMENTS

This research is partially supported by the Brazilian Government, through the Agency CNPq, contracts 620185/2008-2, 309172/2009-8 and 305730/2007-0.

REFERENCES

Arruda N.C.B., Almeida R.C., and do Carmo E.G.D. Discontinuous subgrid formulations for transport problems. *Computer Methods in Applied Mechanics and Engineering*, to appear, 2010.

Brezzi F., Marini L.D., and Sangalli G. Link-cutting bubbles for the stabilization of convection-diffusion-reaction problems. *Mathematical Models and Methods in Applied Science*, 13:445–461, 2003.

Brooks A.N. and Hughes T.J.R. Streamline upwind Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.

Catabriga L. and Coutinho A. Implicit SUPG solution of Euler equations using edge-based data structures. *Computer Methods in Applied Mechanics and Engineering*, 191:3477–3490, 2002.

Coutinho A., Martins M., Alves J.L.D., Landau L., and Moraes A. Edge-based finite element techniques for nonlinear solid mechanics problems. *International Journal for Numerical Methods in Engineering*, 50:2053–2068, 2001.

Donea J. and Huerta A. *Finite Element Methods for Flow Problems*. John Wiley and Sons, Ltd, 2003.

Ern A. and Guermond J.L. *Theory and Practice of Finite Elements*. Springer-Verlag, New York, 2004.

Guermond J.L. Subgrid stabilization of galerkin approximations of linear monotone operators. *IMA Journal of Numerical Analysis*, 21:165–197, 2001.

Hughes T.J.R., Franca L.P., and Hulbert G.M. A new finite element formulation for computational fluid dynamics: Viii. the galerkin-least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73:173–189, 1989.

Hughes T.J.R., Scovazzi G., and Franca L.P. *Multiscale and Stabilized Methods, Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd., 2004.

Kaya S. and Rivière B. A discontinuous subgrid eddy viscosity method for the time-dependent Navier-Stokes equations. *SIAM J. Numer. Anal.*, 43(4):1572–1595 (electronic), 2005.

Saad Y. *Iterative methods for sparse linear systems*. PWS Publishing Company., 1995.

Santos I.P. and Almeida R.C. A nonlinear subgrid method for advection-diffusion problems.

Computer Methods in Applied Mechanics and Engineering, 196:4771–4778, 2007.

Tadmor E. Convergence of spectral methods for nonlinear conservation laws. *SIAM Journal on Numerical Analysis*, 26(1):30–44, 1989. ISSN 00361429.