COMPARISON BETWEEN GMRES AND LCD ITERATIVE METHODS IN THE FINITE ELEMENT AND FINITE DIFFERENCE SOLUTION OF CONVECTION-DIFFUSION EQUATIONS

Abstract. *In this work we evaluate the performance of the left conjugate direction method (LCD) for the solution of non-symmetric systems of linear equations arising from the finite element and the finite difference discretizations of the convection-diffusion equation. We solve the steady convection-diffusion equation using the SUPG finite element formulation and the nonlinear convection-diffusion by centered finite differences. In the latter case the resulting nonlinear system of equations is solved by the inexact Newton method. We extend the original LCD algorithm to accomodate restarts, using only one matrix-vector product per step. Our discussion considers comparison studies between the computational efficiency of the GMRES and LCD methods and some issues related to the choice of the forcing term in the inexact Newton method.*

Keywords: *left conjugate direction method, residual minimal generalized method, inexact Newton method.*

1. INTRODUCTION

Numerical strategies for flow problems in science and engineering often requires repeated solution of nonlinear systems of equations involving millions of unknowns. After some form of linearization, these systems are usually solved by generalizations of the conjugate gradient method, GMRES or QMR and its variants (Saad, 1996). The success of this solution strategy requires an efficient implementation of matrix-vector products and the choice of a suitable preconditioner. Within the finite element method, implementations of global matrix-vector products are performed by element level products followed by global assembly, which forms the core of element-by-element strategies as introduced in the finite element simulation of compressible flow by Shakib et al. (1989). In the other hand, when centered finite differences are used, we arrive to a matrix involving only 5 nonzero diagonals.

Yuan et al. (2003) introduced a new algorithm for solving nonsymmetric, nonsingular linear systems, the Left Conjugate Direction (LCD for short) method. This method is based on the concept of left and right conjugate vectors for nonsymmetric and nonsingular matrices and possesses several theoretical advantages: (i) it has a finite termination property; (ii) breakdown for general matrices can be avoided and (iii) there is a connection between LCD and LU decomposition. Initial experiments in Yuan et al. (2003) using a MATLAB implementation have shown that LCD has attractive convergence rates when compared to Bi-CGSTAB, QMR and GMRES algorithms.

Catabriga et al. (2004) evaluated the performance of LCD in the solution of nonsymmetric systems of linear equations arising from the implicit semi-discrete SUPG finite element formulation for advection-diffusion and inviscid compressible flows described in Catabriga and Coutinho (2002). They extended the original algorithm to accommodate restarts and typical finite element preconditioners very much in the same manner Shakib et al. (1989) did for GMRES. Comparisons with other Krylov space methods with or without preconditioning unfortunately do not favour LCD. Although requiring usually less iterations, CPU times and memory are larger than GMRES, Bi-CGSTAB and TFQMR. The main reason is the need to compute two matrix-vector products per iteration, one with the coefficient matrix and the other with its transposed matrix. Those are the dominant costs, although thanks to the element-by-element data structure both are performed with the same efficiency.

Valentim et al. (2004) study the solution of nonlinear systems using an inexact Newton method where the approximate solution of the resulting linear system at each iteration is obtained by LCD or GMRES. A spatial discretization based on central finite difference formulation of the heat equation and the convection-diffusion equation was considered. They showed comparison studies between the computational efficiency of the two linear solvers and some issues related to the choice of the forcing term in the inexact Newton method. The results show that the LCD is faster than GMRES in most of the cases.

Recently, Dai and Yuan (2004) proposed a new technique to overcome the breakdown problem appearing in the semi-conjugate direction method and a memory limitation scheme similar to the limited-memory BFGS method to minimize memory requirements of the original algorithm. In this work, we introduce restarts on the LCD algorithm given by Dai and Yuan (2004) and compare it with the restarted LCD algorithm given by Catabriga et al. (2004) and the restarted GMRES method for the solution of the linear and nonlinear problems discretized by finite element and finite difference methods. For the nonlinear problems we study the choice of the forcing term of the inexact Newton method.

The remainder of this work is organized as follows. In the next section we review the stabilized finite element formulations for linear convection equation and the finite difference discretization of the nonlinear convection equation. Section 3 makes a brief review of the inex-

act Newton method and shows the forcing term evaluation strategies. In section 4 we describe the LCD algorithm, with particular emphasis on the introduction of restarts. Next section shows several numerical experiments, where we compare the performance of LCD with GMRES in the convection flow problems. Finally the paper ends with a summary of our main conclusions.

2. GOVERNING EQUATIONS AND DISCRETE FORMULATIONS

2.1 Linear Convection-diffusion equation

Let us consider the following time-dependent advection-diffusion equation in conservative form defined in a domain Ω with boundary Γ:

$$\boldsymbol{\beta}.\nabla u - \nabla.(\boldsymbol{\kappa}\nabla u) = f. \tag{1}$$

where u represents the quantity being transported (e.g. temperature, concentration), $\boldsymbol{\beta}$ is the flow velocity and $\boldsymbol{\kappa}$ is the volumetric diffusivity given as,

$$\boldsymbol{\kappa} = \begin{bmatrix} \kappa_x & 0 \\ 0 & \kappa_y \end{bmatrix}. \tag{2}$$

The essential and natural boundary conditions appended to equation (1) are:

$$\begin{aligned} u &= g \quad \text{on } \Gamma_g, \\ \boldsymbol{n}.\boldsymbol{\kappa}\nabla u &= h \quad \text{on } \Gamma_h, \end{aligned} \tag{3}$$

where g and h are given functions of $\boldsymbol{x} = (x, y)$ and t, $\boldsymbol{n}$ is the unit outward normal vector at the boundary, Γ_g and Γ_h are the complementary subsets of Γ where boundary conditions are prescribed.

Consider a finite element discretization of Ω into elements Ω_e, $e = 1, \ldots, n_{el}$, where n_{el} is the number of elements. We consider piecewise linear basis spanning finite-dimensional trial solution and test function spaces $\mathcal{S}^h$ and $\mathcal{V}^h$. The stabilized finite element formulation of equation (1) can then be written as follows. Find $u^h \in \mathcal{S}^h$ such that $\forall\ w^h \in \mathcal{V}^h$:

$$\begin{aligned} \int_\Omega \left(w^h \boldsymbol{\beta}^h.\nabla u^h - \nabla w^h.\boldsymbol{\kappa}\nabla u^h \right) d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{SUPG} \boldsymbol{\beta}^h.\nabla w^h \left(\boldsymbol{\beta}^h.\nabla u^h \right) d\Omega \quad = \\ \int_\Omega w^h f d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{SUPG} \boldsymbol{\beta}^h.\nabla w^h f d\Omega, \end{aligned} \tag{4}$$

where τ_{SUPG} is the $SUPG$ stabilization parameter which may computed as suggested in Brooks and Hughes (1982) and Franca et al. (1992). Let the standard finite element approximation be given as follows:

$$u^h(\boldsymbol{x}) \cong \sum_{i=1}^{nnodes} N_i(\boldsymbol{x}) u_i, \tag{5}$$

where $nnodes$ is the number of the nodes, N_i is a shape function corresponding to node i and u_i are the nodal values of u. Then, substituting (5) into (4) we arrive at a system of ordinary differential equations,

$$\boldsymbol{K}\boldsymbol{v} = \boldsymbol{F}, \tag{6}$$

where $\boldsymbol{v} = \{u_1, u_2, \ldots, u_{nnodes}\}^t$ is the vector of nodal values of u, $\boldsymbol{K}$ is called the "stiffness" matrix and $\boldsymbol{F}$ is called the "load" vector.

2.2 Nonlinear Convection-diffusion equation

Let us consider the nonlinear convection-diffusion equation defined in a square domain $\Omega = (0, l_x) \times (0, l_y)$ with boundary Γ:

$$\begin{aligned} Cu\nabla.u - \nabla^2 u &= f, \\ u &= g \quad \text{on } \Gamma, \end{aligned} \tag{7}$$

where $u(x, y)$ represents the the quantity being transported, $f(x, y)$, $g(x, y)$ and constant C are known. Consider a discretization of Ω into a uniform grid with $n + 2$ points in the x direction and $m + 2$ points in the y direction, i.e,

$$x_i = i \times h_x, \quad i = 0, \ldots, n+1 \qquad y_j = j \times h_y, \quad j = 0, \ldots, m+1 \tag{8}$$

where $h_x = \frac{l_x}{n+1}$ and $h_y = \frac{l_y}{m+1}$. Since the values at the boundaries are known, we have $N = n \times m$ unknowns points in Ω. We consider the approximation of the first and second order derivatives by the centered finite differences and we arrive at a nonlinear system of equations,

$$F(u_1, u_2, \ldots, u_N) = \begin{bmatrix} f_1(u_1, u_2, \ldots, u_N) \\ \vdots \\ f_N(u_1, u_2, \ldots, u_N) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9}$$

where $F : \mathbb{R}^N \rightarrow \mathbb{R}^N$ function, $u_1, u_2, \ldots, u_N$ are the unknowns and each f_k depends only the unknowns u_{k-n}, u_{k-1}, u_k and u_{k+n} for $k = 1, 2, \ldots, N$.

3. THE INEXACT NEWTON METHOD

The nonlinear system (9) can be solved by Newton's method. It is a iterative method for nonlinear equations that approximate the function F at a given point $\boldsymbol{u} = (u_1, u_3, \ldots, u_N)^t$ by a linear function. The Jacobian matrix J represents the variation of the function F with respect of $\boldsymbol{u}$. Each iteration of the Newton's method is given by

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{s}^k, \tag{10}$$

where $\boldsymbol{s}^k$ is calculated by the solution of the linear system:

$$J(\boldsymbol{u}^k)\boldsymbol{s}^k = -F(\boldsymbol{u}^k). \tag{11}$$

We may terminate the iteration when the relative nonlinear residual $\|F(\boldsymbol{u}^k)\|/\|F(\boldsymbol{u}^0)\|$ is small. However, if there is error in evaluation of F or the initial iterate is near a solution, a termination decision based on the relative residual may be made too late in the iteration or it may not terminate at all. Kelley (1995) suggested to stop the iteration if

$$\|F(\boldsymbol{u}^k)\| \leq \tau_r \|F(\boldsymbol{u}^0)\| + \tau_a. \tag{12}$$

where the relative error tolerance τ_r and absolute error tolerance τ_a are input to the algorithm. When the iterative method is used to solve the system (11) the Newton's method is known as inexact Newton method (Pernice and Walker, 1998). In this method the Newton equation is relaxed to an inexact Newton condition:

$$\|F(\boldsymbol{u}^k) + J(\boldsymbol{u}^k)\boldsymbol{s}^k\| \leq \eta_k \|F(\boldsymbol{u}^k)\|. \tag{13}$$

for some $\eta_k \in [0, 1)$. The parameter η_k is often called a forcing term, since its roles is to force the residual (11) to be suitably small (Shadid et al., 1997). Inexact Newton methods are

especially well suited for large-scale problems and have been used very successfully in many applications. In this work, we use two schemes for choosing the forcing term, one suggested by Papadrakakis (1993) and other suggested by Kelley (1995). The calculation of the linear system tolerances suggested by Papadrakakis (1993) is given by:

$$\eta_k = min\{\eta_{max}, \left(\frac{\|F(\boldsymbol{u}^k)\|}{\|F(\boldsymbol{u}^0)\|}\right)^r\}, \tag{14}$$

where η_{max} and $0 < r < 1$ are known parameters. In this work we consider $\eta_{max} = 0.9999$ and $r = 0.5$. On the other hand, Kelley (1995) describes that a measure of the degree to which the nonlinear iteration approximates the solution is,

$$\eta_k^A = \gamma \frac{\|F(\boldsymbol{u}^k)\|^2}{\|F(\boldsymbol{u}^{k-1})\|^2}, \tag{15}$$

where $\gamma \in (0, 1]$ is a parameter. If η_k^A uniformly bounded away from 1, then setting $\eta_k = \eta_k^A$ for $k > 0$ would guarantee superlinear convergence. In this way, the most information possible would be extracted from the inner iteration. In order to specify the choice at the first iteration and bound the sequence away from 1 we set

$$\eta_k^B = \begin{cases} \eta_{max}, & k = 0 \\ min(\eta_{max}, \eta_k^A), & k > 0 \end{cases}. \tag{16}$$

The parameter η_{max} is an upper limit on the sequence $\eta_k, k = 1, 2, \ldots$. The initial values suggested by Kelley (1995) are $\gamma = 0.9$ and $\eta_{max} = 0.9999$. It may happen that η_k^B is small for one or more iterations while $\boldsymbol{u}^k$ is still far from the solution. In this case, Kelley (1995) suggested that if η_{k-1} is sufficiently large we do not let η_k decrease by much more than a factor of η_{k-1},i.e,

$$\eta_k^C = \begin{cases} \eta_{max}, & k = 0 \\ min(\eta_{max}, \eta_k^A), & k > 0 \text{ e } \gamma\eta_{k-1}^2 < 0.1 \\ min(\eta_{max}, max(\eta_k^A, \gamma\eta_{k-1}^2)) & k > 0 \text{ e } \gamma\eta_{k-1}^2 > 0.1 \end{cases}, \tag{17}$$

where the constant 0.1 is somewhat arbitrary.

4. THE LEFT CONJUGATE DIRECTION ALGORITHM

The finite element discretization of the linear convection-diffusion equation and the finite difference discretization of the nonlinear convection-diffusion equation described on the previous section, respectively, leads to a linear problem given in (6) and a nonlinear problem given in (9). In any case we have to solve a system of linear equations of the form,

$$Ax = b, \tag{18}$$

where A is a $N \times N$ nonsymmetric sparse matrix, x is the vector of nodal unknowns and b is the out-of-balance force or residual vector.

The LCD method was recently introduced by Yuan et al. (2003). In this method vectors $p_1, p_2, \ldots, p_N \in \mathbb{R}^N$ are called left conjugate gradient vectors of an $N \times N$ real nonsingular matrix A if

$$\begin{aligned} p_i^T A p_j &= 0 \quad \text{for } i < j, \\ p_i^T A p_j &\neq 0 \quad \text{for } i = j. \end{aligned} \tag{19}$$

Suppose that the solution of the system (18) is x^*, and $\{p_1, p_2, \ldots, p_N\}$ are left conjugate gradient vectors of A. Then it follows that

$$x^* = x_0 + \sum_{i=1}^{N} \alpha_i p_i, \tag{20}$$

for every fixed vector x_0. If r denotes the residual vector then

$$r = r_0 - \sum_{i=1}^{N} \alpha_i A p_i, \tag{21}$$

where r_0 is the initial residual vector. To determine α_i, since $p_1, p_2, \ldots, p_N$ are linearly independent, then take r orthogonal to all p_i, that is

$$p_i^T r = 0 \quad \forall i = 1, \ldots, N. \tag{22}$$

From (22) we obtain

$$\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T A p_i}. \tag{23}$$

We also can write

$$r_i = b - Ax_i = r_{i-1} - \alpha_i A p_i, \tag{24}$$

$$x_i = x_0 + \sum_{k=1}^{i} \alpha_k p_k = x_{i-1} + \alpha_i p_i. \tag{25}$$

From (23), (24) and (25) we can implement the left conjugate direction method if we know the set of linearly independent vectors $p_1, p_2, \ldots, p_N$ such that they are left conjugate gradient vectors of A. There is still a recurrence relation among $p_1, p_2, \ldots, p_k$ and r_k to compute the left conjugate gradient vector p_{k+1}, given in Yuan et al. (2003):

$$\begin{aligned} q_0 &= r_k, \\ \beta_i &= -\frac{p_i^T A q_{i-1}}{p_i^T A p_i}, \\ q_i &= q_{i-1} + \beta_i p_i \quad \text{for } i = 1, \ldots, k, \\ p_{k+1} &= q_k. \end{aligned} \tag{26}$$

In this case we need to know the first vector p_1 such that $p_1^T A p_1 \neq 0$. Putting all together, Yuan et al. (2003) described the complete left conjugate direction method as follows:

Algorithm 4.1

1. *Input* x_0, A, p_1 *such that* $p_1^T A p_1 \neq 0$ *and* b;
2. $r_0 = b - Ax_0$;
3. *For* $k = 1, \ldots, N$ *do*
   - 3.1 $q_k = A^T p_k$,
     $\alpha_k = \frac{p_k^T r_{k-1}}{q_k^T p_k}$,
     $x_k = x_{k-1} + \alpha_k p_k$,
     $r_k = r_{k-1} - \alpha_k A p_k$;
   - 3.2 $p_{k+1} = r_k$,
     $\beta_i = -\frac{q_i^T p_{k+1}}{q_i^T p_i}$,
     $p_{k+1} = p_{k+1} + \beta_i p_i$ *for* $i = 1, \ldots, k$.

In Algorithm 4.1 we need to store N vectors p_k and N vectors q_k, furthermore we need two matrix vector product per iteration to obtain the solution x_N. Dai and Yuan (2004) proposed news ideas for the LCD methods where one matrix vector product is need. The new algorithm can be written as follows.

Algorithm 4.2

1. *Input* x_1, A, p_1 *such that* $p_1^T A p_1 \neq 0$ *and* b;
2. $r_1 = b - Ax_0$;
3. $q_1 = Ap_1$;
4. *For* $k = 1, \ldots, N$ *do*

3.1 $\alpha_k = \frac{p_k^T r_k}{p_k^T q_k}$,
$x_{k+1} = x_k + \alpha_k p_k$,
$r_{k+1} = r_k - \alpha_k q_k$;
3.2 $p_{k+1} = r_k$,
$q_{k+1} = Ap_{k+1}$,
For $i = 1, \ldots, k$ *do*
$\beta_i = -\frac{p_i^T q_{k+1}}{p_i^T q_i}$,
$p_{k+1} = p_{k+1} + \beta_i p_i$,
$q_{k+1} = q_{k+1} + \beta_i q_i$.

Catabriga et al. (2004) introduced an algorithm similar to Algorithm 4.1, but with restart as in the GMRES algorithm implemented by Shakib et al. (1989). In this paper we consider the same restart for the Algorithm 4.2 and the new algorithm is given below:

Algorithm 4.3 - LCD(k)

1. *Given* x_1, A, b, l_{max}, k *and* ϵ_{tol}
2. $r_1 = b - Ax_1$
3. $\epsilon = \epsilon_{tol}\|r\|$
4. *Choose* p_1 *such that* $p_1^T A p_1 \neq 0$
5. *For* $l = 1, \ldots, l_{max}$ *do*

5.1$q_l = Ap_l$
5.2. *For* $i = 1, \ldots, k$ *do*
5.2.1. $\alpha_i = \frac{p_i^T r_i}{p_i^T q_i}$
$x_{i+1} = x_i + \alpha_i p_i$
$r_{i+1} = r_i - \alpha_i q_i$
5.2.2. *if* $\|r_i\| < \epsilon$ *then exit loop and* l, x_i *is the solution.*
5.2.3. $p_{i+1} = r_i$
$q_{i+1} = Ap_{i+1}$
For $j = 1, \ldots, i$ *do*
$\beta_j = -\frac{p_j^T q_{i+1}}{p_j^T q_j}$
$p_{i+1} = p_{i+1} + \beta_j p_j$
$q_{i+1} = q_{i+1} + \beta_j q_j$
5.3. *choose the new* p_1 *such that* $p_1^T A p_1 \neq 0$

where l_{max} is the maximum number of iterations, ϵ_{tol} is the user supplied tolerance and k is number of left conjugate directions considered in the restart. We need to store $2k$ N-dimensional vectors $\{p_1, \ldots, p_k\}$ and $\{q_1, \ldots, q_k\}$. For each iteration l we need only one matrix-vector products as in the GMRES algorithm. To start LCD(k), we have to choose p_1 and a new p_1^l for each

l iteration. Catabriga et al. (2004) reported numerical experiments about this choice. The best results were $p_1 = r_1$ and $p_1^l = p_{k+1}^{l-1}$ and in this work we adopt this choice.

5. NUMERICAL RESULTS

In this section we evaluate the LCD algorithm implemented by Yuan et al. (2003) and Dai and Yuan (2004). All results were implemented using restarts. The LCD algorithm given by Yuan et al. (2003) is denoted by LCD$_A$ and it given by Dai and Yuan (2004) is denoted by LCD$_B$.

5.1 Pure convection problem

We consider a pure convection of a scalar on a square domain, where convection is skew to the mesh and the diffusivity is negligible. Figure 1 shows the problem set up. The domain is the unit square $\Omega = [0, 1] \times [0, 1]$ and the boundary conditions are

$$\begin{aligned} &u = 0.0 \quad \text{along} \quad y = 0.0, \\ &u = 0.0 \quad \text{along} \quad x = 0.0 \text{ and } 0.0 < y < 0.25, \\ &u = 1.0 \quad \text{along} \quad x = 0.0 \text{ and } 0.25 < y < 1.0. \end{aligned} \tag{27}$$

The diffusivity is $\kappa_x = \kappa_y = 1 \times 10^{-7}$, the flow direction is 45° from the x-axis, $\|\beta\| = 1$

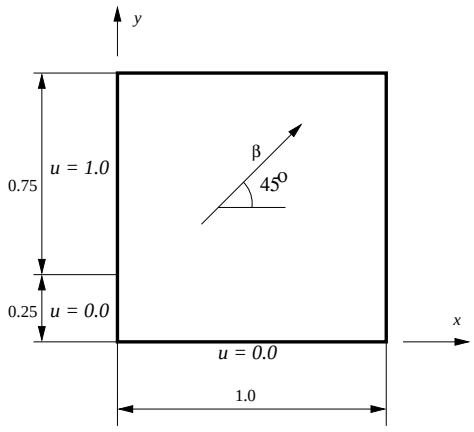
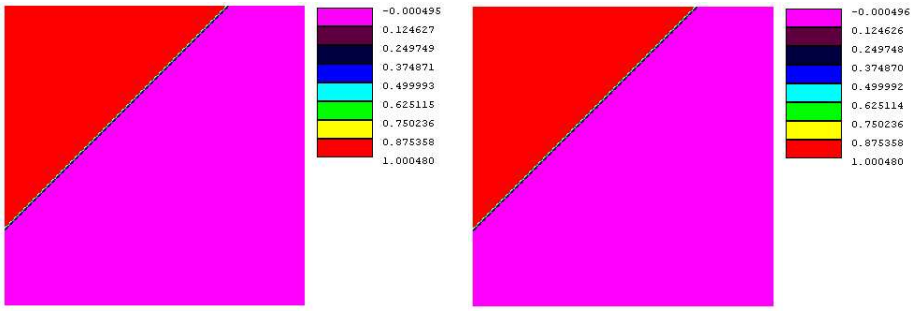


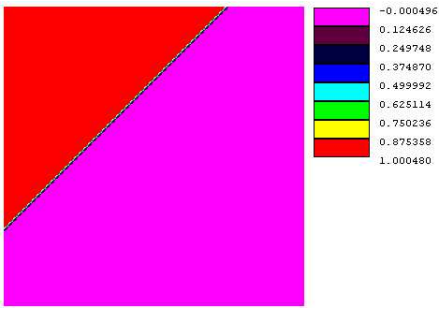
Figure 1: Problem set up - Pure convection of a scalar on a square domain.

and the stabilization parameter is computed as in Brooks and Hughes (1982). The domain is discretized by grids of triangular elements with 64×64, 128×128, 256×256 and 512×512 cells. Each cell is subdivided into four triangles.

Figures 2(a), 2(b) and 2(c) show, respectively, the solution with the LCD(5)$_A$, LCD(5)$_B$ and the GMRES(5) solution for the mesh with 64×64 cells. We observed that the solutions are virtually identical. The solution for other numbers of left conjugate gradient vectors is the same. Figure 3 compares the relative residual evolution for LCD(5)$_A$, LCD(5)$_B$ and GMRES(5) for the four meshes defined before. Iterations are halted when the relative residual reaches 10^{-10}. Although relative residual in LCD(5)$_A$ and LCD(5)$_B$ decrease more slowly than in GMRES(5) in the beginning of the process, the total number of LCD(5)$_A$ and LCD(5)$_B$ iterations are the same and it is smaller than the number of GMRES(5) iterations. A similar behavior is observed for all meshes. Figure 4 shows the performance of LCD$_A$, LCD$_B$ and GMRES methods for k=1, 5, 10, 20, 40 and without restart considering the mesh of 128×128 cells. Results for the other meshes are similar. Table 1 shows the number of iterations (*Niter*) and CPU times for the GMRES and LCD methods using a relative residual tolerance of 10^{-10}. In this Table *Neq* is

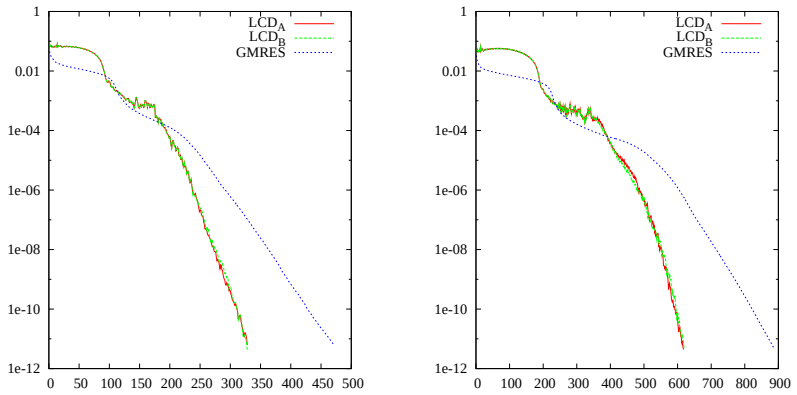


(a) LCD(5)$_A$ solution. (b) LCD(5)$_B$ solution.



(c) GMRES(5) solution.

Figure 2: Mesh with 64×64 cells - Pure convection of a scalar on a square domain.

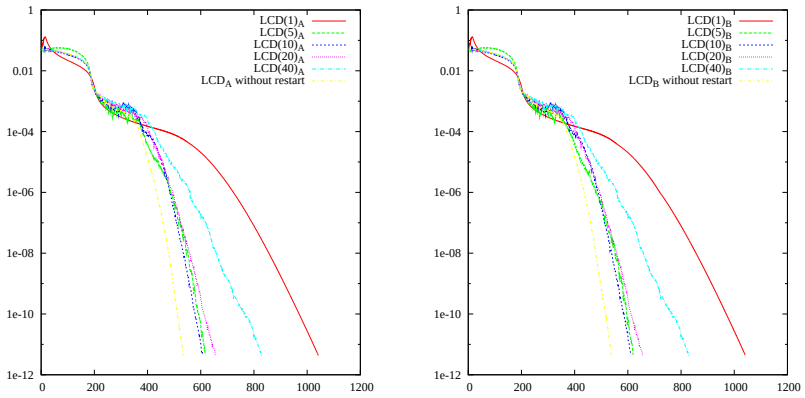


(a) Mesh 64×64. (b) Mesh 128×128.

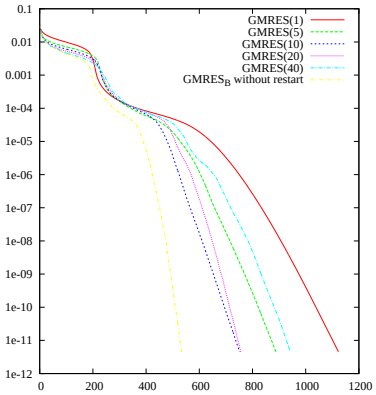
(c) Mesh 256×256. (d) Mesh 512×512.

Figure 3: Relative residual evolution for LCD(5) and GMRES(5) - Pure convection of a scalar on a square domain.

the number of the unknowns. We can observe that the LCD$_A$ and LCD$_B$ method converge with less iterations than GMRES method in most of the cases, however they are slower than GMRES method in the most of cases.



(a) LCD$_A$ (b) LCD$_B$



(c) GMRES

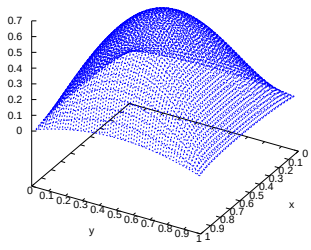
Figure 4: Relative residual evolution for LCD(k)$_A$, LCD(k)$_B$ and GMRES(k) - mesh of 128×128 cells - Pure convection of a scalar on a square domain.

5.2 Nonlinear convection-diffusion problem

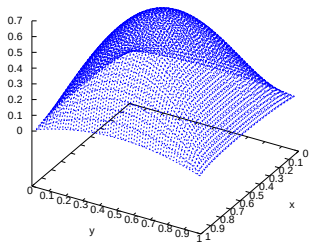
We consider the equation (7) with homogeneous Dirichlet boundary conditions on the unit square $(0,1) \times (0,1)$. Function f has been constructed so that the exact solution was the discretization of $u(x,y) = 10xy(1-x)(1-y)e^{x^{4.5}}$. We set $C = 20$, $\boldsymbol{u}^0 = 0$, $\tau_a = 10^{-9}$ and $\tau_r = 10^{-12}$. In this problem, we use the global Gauss-Seidel preconditioner for all cases. Figures 5(a), 5(b) and 5(c) show, respectively, the solution with the LCD(10)$_A$, LCD(10)$_B$ and GMRES(10) solutions when we consider the domain discretized on a 64×64 cells. We observed that the solutions are virtually identical. The solution for other numbers of left conjugate gradient vectors is the same. From now on, we consider the domain discretized on a 512×512 cells. Figures 6(a), 6(b) and 6(c) show, respectively, plots of the nonlinear residual norm obtained using LCD(10)$_A$, LCD(10)$_B$ and GMRES(10) considering a fixed value for the forcing term and the forcing term computed by Papadrakakis and Kelley criteria. Figures 7(a), 7(b) and 7(c) show, respectively, the forcing behavior using LCD(10)$_A$, LCD(10)$_B$ and GMRES(10) computed by Paradrakakis and Kelley criteria. Figures 8(a), 8(b) and 8(c), respectively, show the number of linear iteration for each nonlinear iteration using LCD(10)$_A$, LCD(10)$_B$ and GMRES(10) considering a fixed value for the forcing term and the forcing term computed by Pa-

Table 1: Computational costs - Pure convection of a scalar on a square domain

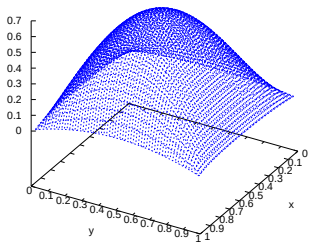
1 vector							
Mesh		GMRES(1)		LCD(1)$_A$		LCD(1)$_B$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	714	7	654	7	654	7
128×128	32768	1123	55	1042	53	1041	54
256×256	131072	1918	385	1784	364	1784	371
5 Vectors							
Mesh		GMRES(5)		LCD(5)$_A$		LCD(5)$_B$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	471	3	328	3	328	2
128×128	32768	888	31	618	33	620	25
256×256	131072	1661	238	1163	244	1182	186
512×512	524288	3104	1726	2384	2086	2378	1533
10 Vectors							
Mesh		GMRES(10)		LCD(10)$_A$		LCD(10)$_B$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	399	2	356	4	356	2
128×128	32768	751	28	608	35	611	28
256×256	131072	1479	220	1091	258	1105	204
20 Vectors							
Mesh		GMRES(20)		LCD(20)$_A$		LCD(20)$_A$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	448	3	401	5	401	3
128×128	32768	756	34	655	44	655	39
256×256	131072	1383	255	1123	307	1150	285
40 Vectors							
Mesh		GMRES(40)		LCD(40)$_A$		LCD(40)$_B$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	595	5	478	7	478	6
128×128	32768	942	60	829	71	829	75
256×256	131072	1552	405	1265	439	1269	467
without restart							
Mesh		GMRES		LCD$_A$		LCD$_B$	
Cells	Neq	Niter	Time(sec)	Niter	Time(sec)	Niter	Time(sec)
64×64	8192	261	6	262	11	262	13
128×128	32768	533	320	534	307	538	481



(a) LCD(10)$_A$ solution.

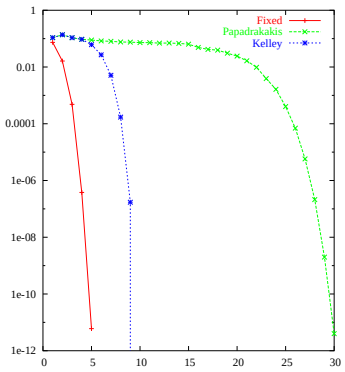


(b) LCD(10)$_B$ solution.

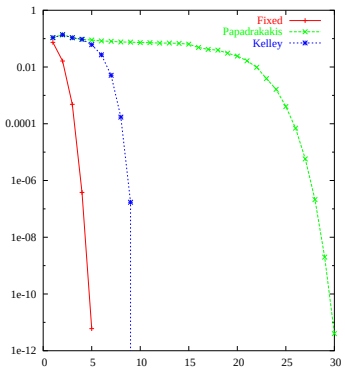


(c) GMRES(10) solution.

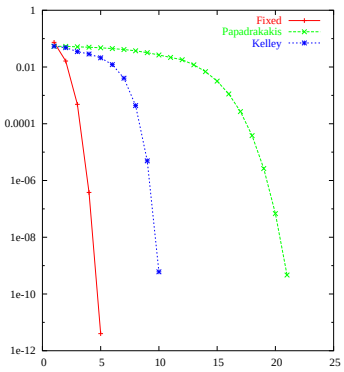
Figure 5: The solution of the convection-diffusion problem - Mesh with 64×64 cells



(a) LCD(10)$_A$ solution.

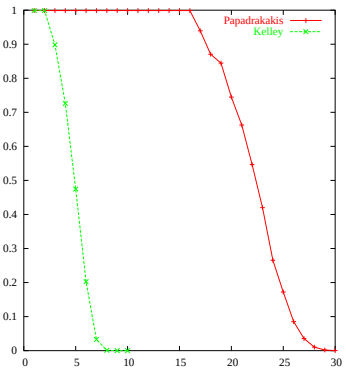


(b) LCD(10)$_B$ solution.

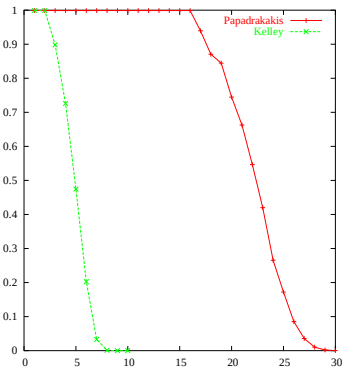


(c) GMRES(10) solution.

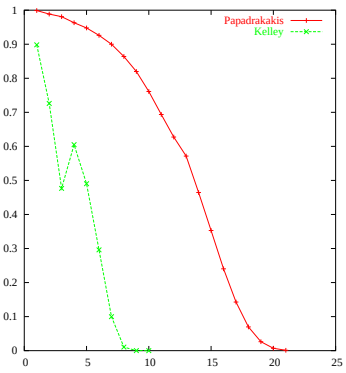
Figure 6: Convergence histories using different choices of the forcing terms (Nonlinear iterations $\times$ $\|F(u^k)\|$) - Mesh with 512×512 cells - Nonlinear convection-diffusion problem.



(a) LCD(10)$_A$ solution.



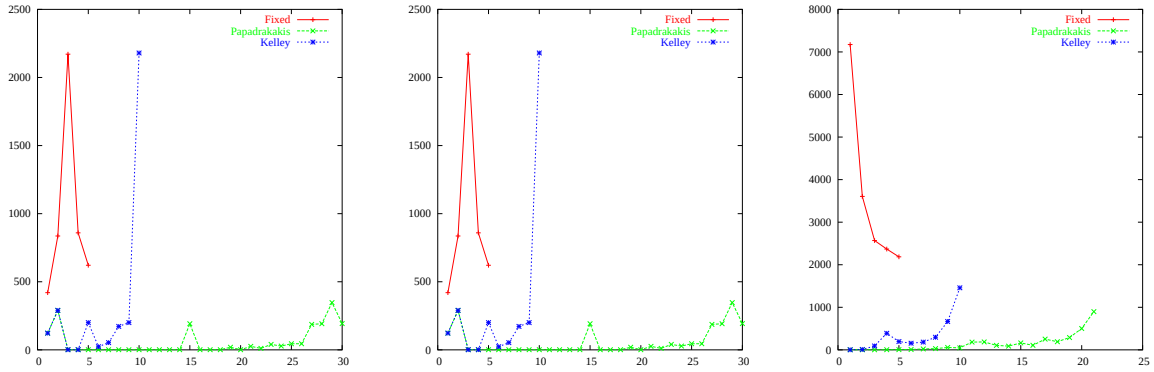
(b) LCD(10)$_B$ solution.



(c) GMRES(10) solution.

Figure 7: Forcing term behavior using different choices of the forcing terms (Nonlinear iterations $\times$ forcing term) - Mesh with 512×512 cells - Nonlinear convection-diffusion problem.

padrakakis and Kelley criteria. We can observe that the fixed forcing term needs less nonlinear iterations for convergence than the other two criteria, but it needs more linear iterations for each nonlinear iteration. Table 2 shows the number of linear iterations (Niter) and CPU times for



(a) LCD(10)$_A$ solution. (b) LCD(10)$_B$ solution. (c) GMRES(10) solution.

Figure 8: Number of linear iteration behaviour using different choices of the forcing terms - Mesh with 512×512 cells - Nonlinear convection-diffusion problem.

the LCD$_A$, LCD$_B$ and GMRES methods using three types of forcing term (fixed, Papadrakakis and Kelley). We can observe that the inexact methods decrease the number of linear iterations when compared with the fixed tolerance criterion. The Papadrakakis criterion needs less inner linear iterations for all cases. The LCD$_A$ algorithm is faster than LCD$_B$ for all number of restart vectors tested. Moreover, LCD tends to be slower than GMRES as the number of basis vectors increase.

6. CONCLUSIONS

In this work we compared the performance of LCD and GMRES algorithm in the finite element and finite difference solution, respectively, for a linear convection-diffusion and a nonlinear convection-diffusion problems. The nonlinear problem solution have been carried out by the inexact Newton method. We studied two choices for the forcing term of the inexact Newton method. We implemented two different algorithms for the LCD method. One considers two matrix-vector products per iteration (LCD$_A$) and the other considers only one matrix-vector product, but needs to compute more inner products, if we consider k vectors to restart (LCD$_B$).

For the finite element experiments we can observe that the LCD$_B$ algorithm is faster than GMRES and LCD$_A$ only when we consider small number of basis vectors. For the finite difference experiments, the LCD$_A$ is faster than LCD$_B$ for all cases. LCD$_A$ using 10 vectors to restart and Papadrakakis criterion gave the smallest CPU time. However, GMRES is faster than LCD in the most cases. On the criteria to choose the forcing term for the inexact Newton method, we conclude that the Papadrakakis criterion was the best option.

Acknowledgements

We would like to thank the support of CNPq/MCT and the Federal University do Espírito Santo under the PIBIC/PIVIC program. Partial support of the CAPES-University of Texas at Austin International Cooperation Program is gratefully acknowledged.

Table 2: Computational costs - Mesh with 512×512 cells - Nonlinear convection-diffusion problem

5 vectors						
η_k	GMRES(5)		LCD(5)$_A$		LCD(5)$_B$	
	Niter	Time(min)	Niter	Time(min)	Niter	Time(min)
Fixed (10^{-5})	35,834	103.18	8,835	38.73	8,833	38.36
Papadrakakis	7,063	21.41	4,214	18.35	4,214	18.55
Kelley	6,214	18.63	12,614	54.23	12,614	54,73
10 vectors						
η_k	GMRES(10)		LCD(10)$_A$		LCD(10)$_B$	
	Niter	Time(min)	Niter	Time(min)	Niter	Time(min)
Fixed (10^{-5})	17,903	66.00	4,907	25.36	4,907	28.75
Papadrakakis	3,144	11.41	1,754	9.08	1,754	10.53
Kelley	3,444	12.76	3,243	16.83	3,243	17.93
20 vectors						
η_k	GMRES(20)		LCD(20)$_A$		LCD(20)$_B$	
	Niter	Time(min)	Niter	Time(min)	Niter	Time(min)
Fixed (10^{-5})	9,098	47.15	12,809	87.71	12,809	116.33
Papadrakakis	1,721	9.10	1,414	9.93	1,414	12.11
Kelley	2,450	13.10	2,132	14.68	2,132	19.36
40 vetores						
η_k	GMRES(40)		LCD(40)$_A$		LCD(40)$_B$	
	Niter	Time(min)	Niter	Time(min)	Niter	Time(min)
Fixed (10^{-5})	4,495	39.45	2,962	29.96	2,962	43.20
Papadrakakis	1,319	10.45	1,481	14.03	1,481	22.06
Kelley	1,650	18.83	1,678	22.10	1,678	25.70

REFERENCES

Brooks, A. & Hughes, T., 1982. Streamline Upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. and Engrg.*, vol. 32, pp. 199–259.

Catabriga, L. & Coutinho, A. L. G. A., 2002. Implicit SUPG solution of Euler equations using edge-based data structures. *Comput. Methods Appl. Mech. and Engrg.*, vol. 191, pp. 3477–3490.

Catabriga, L., Coutinho, A. L. G. A., & Franca, L. P., 2004. Evaluating the lcd algorithm for solving linear systems of equations aring from implicit supg formulation of compressible flows. *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 1513–1534.

Dai, Y. & Yuan, J. Y., 2004. Study on semi-conjugate direction methods for non-symmetric systems. *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 1383–1399.

Franca, L., Frey, S., & Hughes, T., 1992. Stabilized finite element methods: Application to the advective-diffusive model. *Comput. Methods Appl. Mech. and Engrg.*, vol. 95, pp. 253–276.

Kelley, C., 1995. *Iterative Methods for Linear and Nonlinear Equations*. Siam, Philadelphia, US.

Papadrakakis, M., 1993. *Solving Large-Scale Problems in Mechanics: The Development and Application of Computational Solution Procedures*. John Wiley & sons, London, England.

Pernice, M. & Walker, H. F., 1998. Nitsol: A newton iterative solver for nonlinear system. *SIAM J. Sci. Comput.*, vol. 19, pp. 302–318.

Saad, Y., 1996. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston.

Shadid, J. N., Tuminaro, R. S., & Walker, H. F., 1997. An inexact newton method dor fully-couped solution of the navier-stokes equations with heat and mass transport. UC-405 SAN97-0132, Sandia National Laboratories.

Shakib, F., Hughes, T. J. R., & Johan, Z., 1989. A multi-element group preconditioned gmres algorithm for nonsymmetric systems arising in finite element analysis. *Comput. Methods Appl. Mech. and Engrg.*, vol. 65, pp. 415–456.

Valentim, E. C., Pessoa, L. M., Melotti, B. Z., Valli, A. M., & Catabriga, L., 2004. Comparison between gmres and lcd methods in the implementation of an inexact newton method (in portuguese). In *10th Brazilian Congress of Thermal Engineering and Sciences CD-ROM*, pp. 1–12, Rio de Janeiro, Brazil (submitted).

Yuan, J., Golub, G., Plemmons, R., & Cecílio, W. A. G., 2003. Semiconjugate direction methods for real positive definite systems. *BIT (http://www-sccm.stanford.edu/urap/pub-tech.html/SCCM-02-02)*, vol. (accepted).