COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng 2006; **22**:643–656
Published online 5 January 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm.842

Performance of LCD iterative method in the finite element and finite difference solution of convection–diffusion equations

L. Catabriga[1,*,†], A. M. P. Valli[1,‡], B. Z. Melotti[1,§], L. M. Pessoa[1,¶] and A. L. G. A. Coutinho[2,‖]

[1] *Department of Computer Science, Federal University of Espírito Santo, Brazil*
[2] *Center for Parallel Computations and Department of Civil Engineering—COPPE, Federal University of Rio de Janeiro, Brazil*

SUMMARY

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 finite element and finite difference discretizations of the convection–diffusion equation. We extend the LCD algorithm proposed by Dai and Yuan (*Int. J. Numer. Meth. Engng* 2004; **60**:1383–1399) to accommodate restarts. 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. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: left conjugate direction method; iterative solvers; inexact Newton method

1. INTRODUCTION

Numerical strategies for flow problems in science and engineering often requires repeated solution of non-linear systems of equations involving millions of unknowns. After some form of linearization, these systems are usually solved by Krylov subspace iterative methods [1]. Yuan *et al.* [2] introduced a new algorithm for solving non-symmetric, non-singular linear systems, the left conjugate direction (LCD for short) method. This method is based on the concept of left and right conjugate vectors for non-symmetric and non-singular matrices and

*Correspondence to: L. Catabriga, Av. Fernando Ferrari, 514, Goiabeiras, ES 29075-910, Vitória, Brazil.
†E-mail: luciac@inf.ufes.br
‡E-mail: avalli@inf.ufes.br
§E-mail: bzmelotti@inf.ufes.br
¶E-mail: lmantovaneli@hotmail.com
‖E-mail: alvaro@nacad.ufrj.br

Contract/grant sponsor: CNPq/MCT
Contract/grant sponsor: Federal University of Espírito Santo
Contract/grant sponsor: CAPES-University of Texas

Received 9 May 2005
Revised 23 August 2005
Accepted 28 October 2005

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 Reference [2] using a MATLAB implementation have shown that LCD has attractive convergence rates when compared to Bi-CGSTAB, QMR and GMRES algorithms.

Catabriga *et al.* [3] evaluated the performance of the original LCD algorithm in the solution of non-symmetric systems of linear equations arising from the implicit semi-discrete SUPG finite element formulation for inviscid compressible flows described in Reference [4]. They extended the original algorithm to accommodate restarts and typical finite element preconditioners. Comparisons with other Krylov subspace methods with or without preconditioning unfortunately did not favour the LCD method. 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.

Valentim *et al.* [5] studied the solution of non-linear 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 centred finite difference approximations of the heat equation and the convection–diffusion equation was considered. They studied 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 have shown that the LCD method is faster than the GMRES method in most of the cases.

Recently, Dai and Yuan [6] 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 new LCD algorithm given by Dai and Yuan [6] and compare it with the restarted LCD algorithm given by Catabriga *et al.* [3] and the restarted GMRES method for the solution of the linear and non-linear problems discretized by finite element and finite difference methods. For the non-linear 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 briefly review the stabilized finite element formulation for linear convection equation and the finite difference discretization of the non-linear convection equation. Section 3 introduces the inexact Newton method and shows the forcing term evaluation strategies. In Section 4, we describe the LCD algorithms, with particular emphasis on the introduction of restarts. Section 5 shows several numerical experiments, where we compare the performance of the LCD method with the GMRES method. 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 convection–diffusion equation defined in a domain Ω with boundary Γ:

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

$$u = g \quad \text{on } \Gamma_g \tag{2}$$

$$\mathbf{n}.\boldsymbol{\kappa}\nabla u = h \quad \text{on } \Gamma_h \tag{3}$$

where u represents the quantity being transported (e.g. temperature, concentration), $\boldsymbol{\beta}$ is the divergence-free flow velocity and $\boldsymbol{\kappa}$ is the volumetric diffusivity. Equations (2) and (3) are the essential and natural boundary conditions, respectively, g and h are given functions of $\mathbf{x} = (x, y)$, and $\mathbf{n}$ 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, $e = 1, \ldots, n_{\text{el}}$, where n_{el} is the number of elements. The stabilized finite element formulation of Equation (1), described in detail in Reference [7], leads to a system of linear equations

$$\mathbf{K}\mathbf{v} = \mathbf{F} \tag{4}$$

where $\mathbf{v} = \{u_1, u_2, \ldots, u_{n\text{nodes}}\}^t$ is the vector of nodal values of u, $\mathbf{K}$ is called the 'stiffness' matrix and $\mathbf{F}$ is the 'load' vector.

2.2. Non-linear convection–diffusion equation

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

$$\phi u \nabla.u - \nabla^2 u = f \tag{5}$$

$$u = g \quad \text{on } \Gamma \tag{6}$$

where u represents again the quantity being transported, functions $f(x, y)$, $g(x, y)$ and constant ϕ 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.

$$x_i = i \times h_x, \quad i = 0, \ldots, n+1 \quad y_j = j \times h_y, \quad j = 0, \ldots, m+1 \tag{7}$$

where $h_x = l_x/(n+1)$ and $h_y = l_y/(m+1)$. 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 centred finite differences, arriving to the non-linear system of equations

$$F(u_1, u_2, \ldots, u_N) = \begin{bmatrix} f_1(u_1, u_2, \ldots, u_N) \\ \vdots \\ f_N(u_1, u_2, \ldots, u_N) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8}$$

where $F : \mathbb{R}^N \to \mathbb{R}^N$ is a non-linear vector 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, u_{k+1} and u_{k+n} for $k = 1, 2, \ldots, N$.

3. THE INEXACT NEWTON METHOD

The non-linear system (8) can be solved by Newton's method. It is an iterative method for non-linear equations that approximate the function F at a given point $\mathbf{u} = (u_1, u_3, \ldots, u_N)^t$ by a linear function. The Jacobian matrix J represents the variation of the function F with respect of $\mathbf{u}$. Each iteration of the Newton's method is given by

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{s}^k \tag{9}$$

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

$$J(\mathbf{u}^k)\mathbf{s}^k = -F(\mathbf{u}^k) \tag{10}$$

We may terminate the iteration when the relative non-linear residual $\|F(\mathbf{u}^k)\|/\|F(\mathbf{u}^0)\|$ 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 [8] suggested to stop the iteration if

$$\|F(\mathbf{u}^k)\| \leqslant \tau_r \|F(\mathbf{u}^0)\| + \tau_a \tag{11}$$

where the relative error tolerance τ_r and absolute error tolerance τ_a are input to the algorithm. When an iterative method is used to solve system (10), the resulting method is known as inexact Newton method [9]. 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 adaptively tolerances for the inner iterative method, or the forcing term, one suggested by Papadrakakis [10] and other suggested by Kelley [8]. The calculation of the linear system tolerances suggested by Papadrakakis [10] is given by

$$\eta_k = \min\left\{\eta_{\max}, \left(\frac{\|F(\mathbf{u}^k)\|}{\|F(\mathbf{u}^0)\|}\right)^v\right\} \tag{12}$$

where $\eta_{\max}$ and $0 < v < 1$ are known parameters. In this work we consider $\eta_{\max} = 0.9999$ and $v = 0.5$. On the other hand, Kelley [8] suggested that the convergence was superlinear using the values

$$\eta_k = \begin{cases} \eta_{\max}, & k = 0 \\ \min(\eta_{\max}, \eta_k^A), & k > 0 \text{ and } \gamma\eta_{k-1}^2 < 0.1 \\ \min(\eta_{\max}, \max(\eta_k^A, \gamma\eta_{k-1}^2)), & k > 0 \text{ and } \gamma\eta_{k-1}^2 > 0.1 \end{cases} \tag{13}$$

where the $\eta_{\max}$ is an upper limit on the sequence $\eta_k, k = 1, 2, \ldots$. The constant 0.1 is somewhat arbitrary and $\eta_k^A = \gamma(\|F(\mathbf{u}^k)\|^2/\|F(\mathbf{u}^{k-1})\|^2)$. The initial values suggested by Kelley [8] are $\gamma = 0.9$ and $\eta_{\max} = 0.9999$.

4. THE LEFT CONJUGATE DIRECTION ALGORITHM

The finite element discretization of the linear convection–diffusion equation and the finite difference discretization of the non-linear convection–diffusion equation described before yields a linear problem given in (4) and a non-linear problem given in (8). In any case we have to solve a system of linear equations of the form

$$Ax = b \tag{14}$$

where A is an $N \times N$ non-symmetric sparse matrix, x is the vector of nodal unknowns and b is a known vector.

The LCD method was recently introduced in Reference [2]. In this method vectors p_1, $p_2, \ldots, p_N \in \mathbb{R}^N$ are called left conjugate direction vectors of an $N \times N$ real non-singular matrix A if

$$\begin{aligned} p_i^{\mathrm{T}} A p_j &= 0 \quad \text{for } i<j \\ p_i^{\mathrm{T}} A p_j &\neq 0 \quad \text{for } i=j \end{aligned} \tag{15}$$

Suppose that the solution of system (14) is x^*, and $\{p_1, p_2, \ldots, p_N\}$ are left conjugate direction vectors of A. Then it follows that

$$x^* = x_0 + \sum_{i=1}^{N} \alpha_i p_i \tag{16}$$

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

$$r = r_0 - \sum_{i=1}^{N} \alpha_i A p_i \tag{17}$$

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

$$p_i^{\mathrm{T}} r = 0 \quad \forall i = 1, \ldots, N \tag{18}$$

From (18) we obtain

$$\alpha_i = \frac{p_i^{\mathrm{T}} r_{i-1}}{p_i^{\mathrm{T}} A p_i} \tag{19}$$

We also can write

$$r_i = b - Ax_i = r_{i-1} - \alpha_i A p_i \tag{20}$$

$$x_i = x_0 + \sum_{k=1}^{i} \alpha_k p_k = x_{i-1} + \alpha_i p_i \tag{21}$$

From (19)–(21) 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 direction 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}. For this, we need to know the first vector p_1 such that $p_1^{\mathrm{T}} A p_1 \neq 0$. Yuan *et al.* [2] described the complete left conjugate direction method as follows.

Algorithm 4.1

```
1. Input $x$, $A$, $p_1$ such that $p_1^{\mathrm{T}} A p_1 \neq 0$ and $b$;
2. $r = b - Ax$;
3. For $k = 1, \ldots, N$ do
    3.1. $q_k = A^{\mathrm{T}} p_k$,
         $\alpha_k = p_k^{\mathrm{T}} r / q_k^{\mathrm{T}} p_k$,
         $x = x + \alpha_k p_k$,
         $r = r - \alpha_k A p_k$;
    3.2. $p_{k+1} = r$,
         $\beta_i = -q_i^{\mathrm{T}} p_{k+1}/q_i^{\mathrm{T}} p_i$,
         $p_{k+1} = p_{k+1} + \beta_i p_i$ 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. Dai and Yuan [6] proposed new ideas for the LCD methods where one matrix–vector product is needed. The new algorithm can be written as follows.

Algorithm 4.2

```
1. Input $x$, $A$, $p_1$ such that $p_1^{\mathrm{T}} A p_1 \neq 0$ and $b$;
2. $r = b - Ax$;
3. $q_1 = A p_1$;
4. For $k = 1, \ldots, N$ do
    3.1. $\alpha_k = p_k^{\mathrm{T}} r / p_k^{\mathrm{T}} q_k$,
         $x = x + \alpha_k p_k$,
         $r = r - \alpha_k q_k$;
    3.2. $p_{k+1} = r$,
         $q_{k+1} = A p_{k+1}$,
         For $i = 1, \ldots, k$ do
             $\beta_i = -p_i^{\mathrm{T}} q_{k+1} / p_i^{\mathrm{T}} q_i$,
             $p_{k+1} = p_{k+1} + \beta_i p_i$,
             $q_{k+1} = q_{k+1} + \beta_i q_i$.
```

Catabriga *et al.* [3] introduced an algorithm similar to Algorithm 4.1, but with restart as in the GMRES algorithm implemented by Shakib *et al.* [11]. In this paper we consider the same restart scheme applied to Algorithm 4.2. The new algorithm is given below.

Algorithm 4.3 (LCD(k))

```
1. Given $x$, $A$, $b$, $l_{\max}$, $k_{\max}$ and $\eta$
2. $r = b - Ax$
3. $\varepsilon = \eta \|r\|$
4. Choose $p_1$ such that $p_1^{\mathrm{T}} A p_1 \neq 0$
5. For $l = 1, \ldots, l_{\max}$ do
    5.1. $q_1 = A p_1$
    5.2. For $k = 1, \ldots, k_{\max}$ do
        5.2.1. $\alpha_k = p_k^{\mathrm{T}} r / p_k^{\mathrm{T}} q_k$
               $x = x + \alpha_k p_k$
               $r = r - \alpha_k q_k$
        5.2.2. if $\|r\| < \varepsilon$ then exit loop $k$ and $l$, $x$ is the solution.
        5.2.3. $p_{k+1} = r$
               $q_{k+1} = A p_{k+1}$
               For $i = 1, \ldots, k$ do
                   $\beta_i = -p_i^{\mathrm{T}} q_{k+1} / p_i^{\mathrm{T}} q_i$
                   $p_{k+1} = p_{k+1} + \beta_i p_i$
                   $q_{k+1} = q_{k+1} + \beta_i q_i$
    5.3. Choose the new $p_1$ such that $p_1^{\mathrm{T}} A p_1 \neq 0$,
```

where $l_{\max}$ is the maximum number of iterations, $k_{\max}$ is the number of left conjugate directions considered for each restart. Note that the number of left conjugate direction vectors was considered equal to the number of restarts. In principle both can be different. We need to store $2k_{\max}$ N-dimensional vectors $\{p_1,\ldots,p_{k_{\max}}\}$ and $\{q_1,\ldots,q_{k_{\max}}\}$. For each iteration l we need only one matrix–vector product as in the GMRES algorithm. To start LCD($k_{\max}$), we have to choose p_1 and the subsequent p_1 for each $k_{\max}$ iteration. Catabriga *et al.* [3] reported numerical experiments about this choice. The best results were $p_1 = r$ for $l = 1$ and $p_1 = p_{k_{\max}+1}$ for $l = 2, 3, \ldots,$ and in this work we adopt this choice.

5. NUMERICAL RESULTS

In this section, we evaluate the LCD algorithm implemented by Yuan *et al.* [2] and Dai and Yuan [6]. All results were implemented using restarts unless stated otherwise. The LCD algorithm given by Yuan *et al.* [2] is denoted by LCD$_A$ and the algorithm in Reference [6] 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

$$\begin{aligned} u &= 0.0 \quad \text{along } y = 0.0 \\ u &= 0.0 \quad \text{along } x = 0.0 \text{ and } 0.0 < y < 0.25 \\ u &= 1.0 \quad \text{along } x = 0.0 \text{ and } 0.25 < y < 1.0 \end{aligned} \tag{22}$$

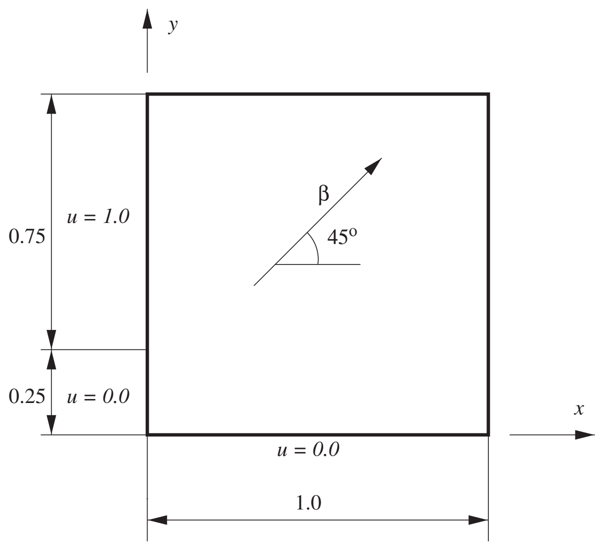


Figure 1. Problem set up—pure convection of a scalar on a square domain.

The diffusivity is $\kappa_x = \kappa_y = 1 \times 10^{-7}$, the flow direction is 45° from the x-axis, $\|\beta\| = 1$ and the stabilization parameter is computed as in Reference [7]. 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. For these meshes we studied the influence of restart, considering several test cases, from no restart until a restart every 40 iterations. We noted that the solutions computed by $\mathrm{LCD_A}$, $\mathrm{LCD_B}$ and GMRES are virtually identical for any numbers of left conjugate direction vectors.

Table I 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 the number of the unknowns. We can observe in Table I that the $\mathrm{LCD_A}$ and $\mathrm{LCD_B}$ methods converge with less iterations than GMRES in all cases, except in the case without restart. The $\mathrm{LCD_B}$ algorithm is faster than the $\mathrm{LCD_A}$ in most cases. Only when we consider 10 or less vectors to restart the $\mathrm{LCD_B}$ algorithm is faster than the GMRES algorithm. Figure 2 compares the relative residual evolution for $\mathrm{LCD(5)_A}$, $\mathrm{LCD(5)_B}$ and GMRES(5) for the four meshes defined before. Although the relative residual in $\mathrm{LCD(5)_A}$ and $\mathrm{LCD(5)_B}$ decrease more slowly than in GMRES(5) in the beginning of the process, the total number of $\mathrm{LCD(5)_A}$ and $\mathrm{LCD(5)_B}$ iterations are almost the same and it is smaller than the number of GMRES(5) iterations in all cases. Results for the other number of restart vectors are similar.

5.2. Non-linear convection–diffusion problem

We consider Equation (5) 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 $\phi = 20$, $\mathbf{u}^0 = 0$, $\tau_a = 10^{-9}$ and $\tau_r = 10^{-12}$. In this problem, we use the global Gauss–Seidel preconditioner for all cases and three types of forcing term (fixed, Papadrakakis and Kelley). We observed that $\mathrm{LCD_A}$, $\mathrm{LCD_B}$ and GMRES solutions are virtually identical for any numbers of left conjugate gradient vectors. For the following discussions, we consider the domain discretized on 512×512 cells.

Table II shows the number of linear iterations (Niter) and CPU times for the $\mathrm{LCD_A}$, $\mathrm{LCD_B}$ 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, except for GMRES(5). Now the $\mathrm{LCD_A}$ algorithm is faster than $\mathrm{LCD_B}$ for almost all number of restart vectors tested. This occurs because the calculation of two matrix–vector products per iteration is not the dominant cost in these finite difference experiments. Moreover, LCD tends to be slower than GMRES as the number of basis vectors increase. The best results in terms of CPU time using 20 restart vectors was for GMRES method with all types of forcing terms. When 40 restart vectors are considered, the best results are for GMRES, with the exception of the fixed tolerance case, where $\mathrm{LCD_A}$ is better than GMRES.

Figures 3(a)–(c) show plots of the non-linear residual norm and Figures 4(a)–(c) show the number of linear iterations in each non-linear iteration obtained using $\mathrm{LCD(10)_A}$, $\mathrm{LCD(10)_B}$ and GMRES(10). We can observe that the fixed forcing term needs less non-linear iterations for convergence than the other two criteria, but it needs more linear iterations for each non-linear iteration. Table II shows that, for all numbers of restart vectors tested, larger

Table I. Computational costs—pure convection of a scalar on a square domain.

1 Vector							
Mesh		GMRES(1)		LCD(1)$_A$		LCD(1)$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	714	7	654	7	654	7
128 × 128	32 768	1123	55	1042	53	1041	54
256 × 256	131 072	1918	385	1784	364	1784	371

5 Vectors							
Mesh		GMRES(5)		LCD(5)$_A$		LCD(5)$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	471	3	328	3	328	2
128 × 128	32 768	888	31	618	33	620	25
256 × 256	131 072	1661	238	1163	244	1182	186
512 × 512	524 288	3104	1726	2384	2086	2378	1533

10 Vectors							
Mesh		GMRES(10)		LCD(10)$_A$		LCD(10)$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	399	2	356	4	356	2
128 × 128	32 768	751	28	608	35	611	28
256 × 256	131 072	1479	220	1091	258	1105	204

20 Vectors							
Mesh		GMRES(20)		LCD(20)$_A$		LCD(20)$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	448	3	401	5	401	3
128 × 128	32 768	756	34	655	44	655	39
256 × 256	131 072	1383	255	1123	307	1150	285

40 Vectors							
Mesh		GMRES(40)		LCD(40)$_A$		LCD(40)$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	595	5	478	7	478	6
128 × 128	32 768	942	60	829	71	829	75
256 × 256	131 072	1552	405	1265	439	1269	467

Without restart							
Mesh		GMRES		LCD$_A$		LCD$_B$	
Cells	*N*eq	*N*iter	Time (s)	*N*iter	Time (s)	*N*iter	Time (s)
64 × 64	8192	261	6	262	11	262	13
128 × 128	32 768	533	320	534	307	538	481

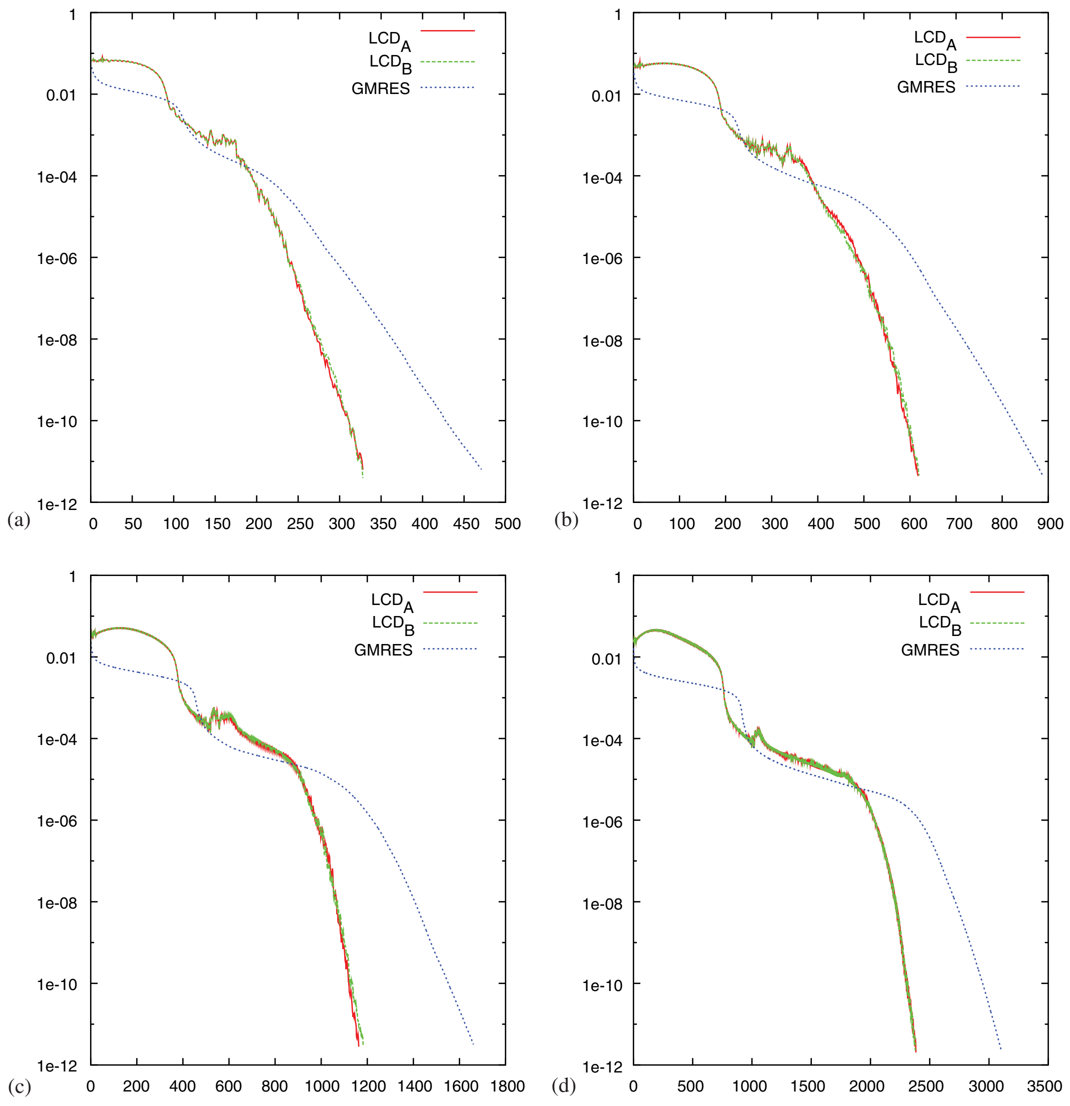


Figure 2. Relative residual evolution for LCD(5)$_A$, LCD(5)$_B$ and GMRES(5)—pure convection of a scalar on a square domain: (a) mesh 64×64; (b) mesh 128×128; (c) mesh 256×256; and (d) mesh 512×512.

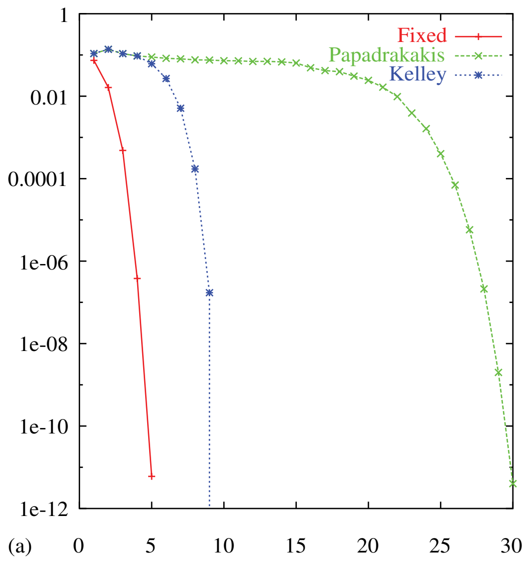
numbers of linear iterations are observed when the forcing term is fixed. Further, we can see in Figure 4 that when the Papadrakakis criterion is selected, less linear iterations are needed, which implies in less processing time. This may be due to the adaptative choice of tolerance in the inexact Newton method.

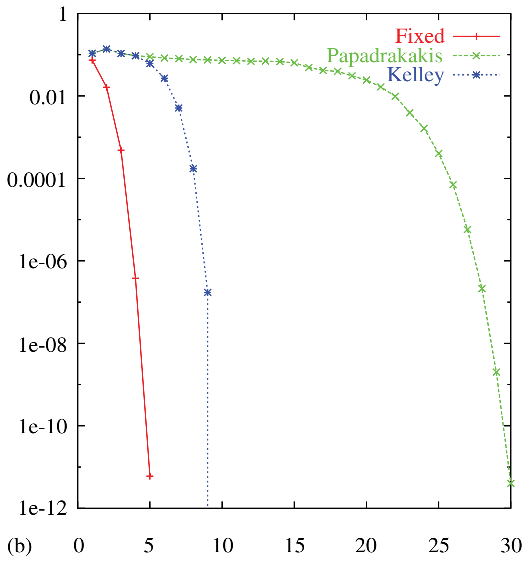
Table II. Computational costs—mesh with 512×512 cells—non-linear convection–diffusion problem.

5 Vectors						
	GMRES(5)		$\text{LCD}(5)_A$		$\text{LCD}(5)_B$	
η_k	*N*iter	Time (min)	*N*iter	Time (min)	*N*iter	Time (min)
Fixed (10^{-5})	35 834	103.18	8835	38.73	8833	38.36
Papadrakakis	7063	21.41	4214	18.35	4214	18.55
Kelley	6214	18.63	12 614	54.23	12 614	54.73
10 Vectors						
	GMRES(10)		$\text{LCD}(10)_A$		$\text{LCD}(10)_B$	
η_k	*N*iter	Time (min)	*N*iter	Time (min)	*N*iter	Time (min)
Fixed (10^{-5})	17 903	66.00	4907	25.36	4907	28.75
Papadrakakis	3144	11.41	1754	9.08	1754	10.53
Kelley	3444	12.76	3243	16.83	3243	17.93
20 Vectors						
	GMRES(20)		$\text{LCD}(20)_A$		$\text{LCD}(20)_B$	
η_k	*N*iter	Time (min)	*N*iter	Time (min)	*N*iter	Time (min)
Fixed (10^{-5})	9098	47.15	12 809	87.71	12 809	116.33
Papadrakakis	1721	9.10	1414	9.93	1414	12.11
Kelley	2450	13.10	2132	14.68	2132	19.36
40 Vectors						
	GMRES(40)		$\text{LCD}(40)_A$		$\text{LCD}(40)_B$	
η_k	*N*iter	Time (min)	*N*iter	Time (min)	*N*iter	Time (min)
Fixed (10^{-5})	4495	39.45	2962	29.96	2962	43.20
Papadrakakis	1319	10.45	1481	14.03	1481	22.06
Kelley	1650	18.83	1678	22.10	1678	25.70

6. CONCLUSION

In this work, we compared the performance of LCD and GMRES algorithms in the finite element and finite difference solutions for linear and non-linear convection–diffusion problems. The non-linear 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).





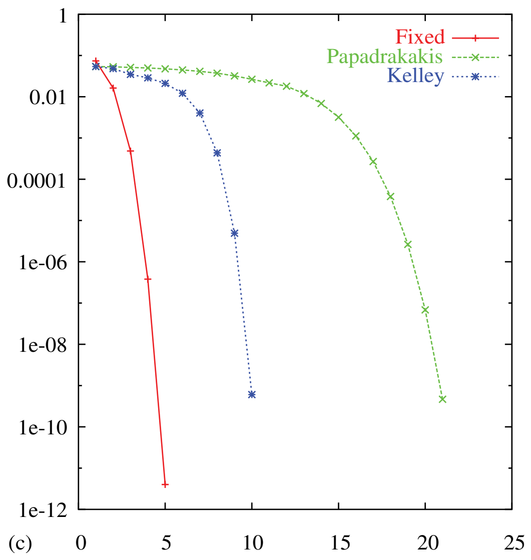


Figure 3. Convergence histories using different choices of the forcing terms (Non-linear iterations $\times$ $\|F(u^k)\|$)—mesh with 512$\times$512 cells—non-linear convection–diffusion problem: (a) LCD(10)$_A$ solution; (b) LCD(10)$_B$ solution; and (c) GMRES (10) solution.

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. In general, GMRES is faster

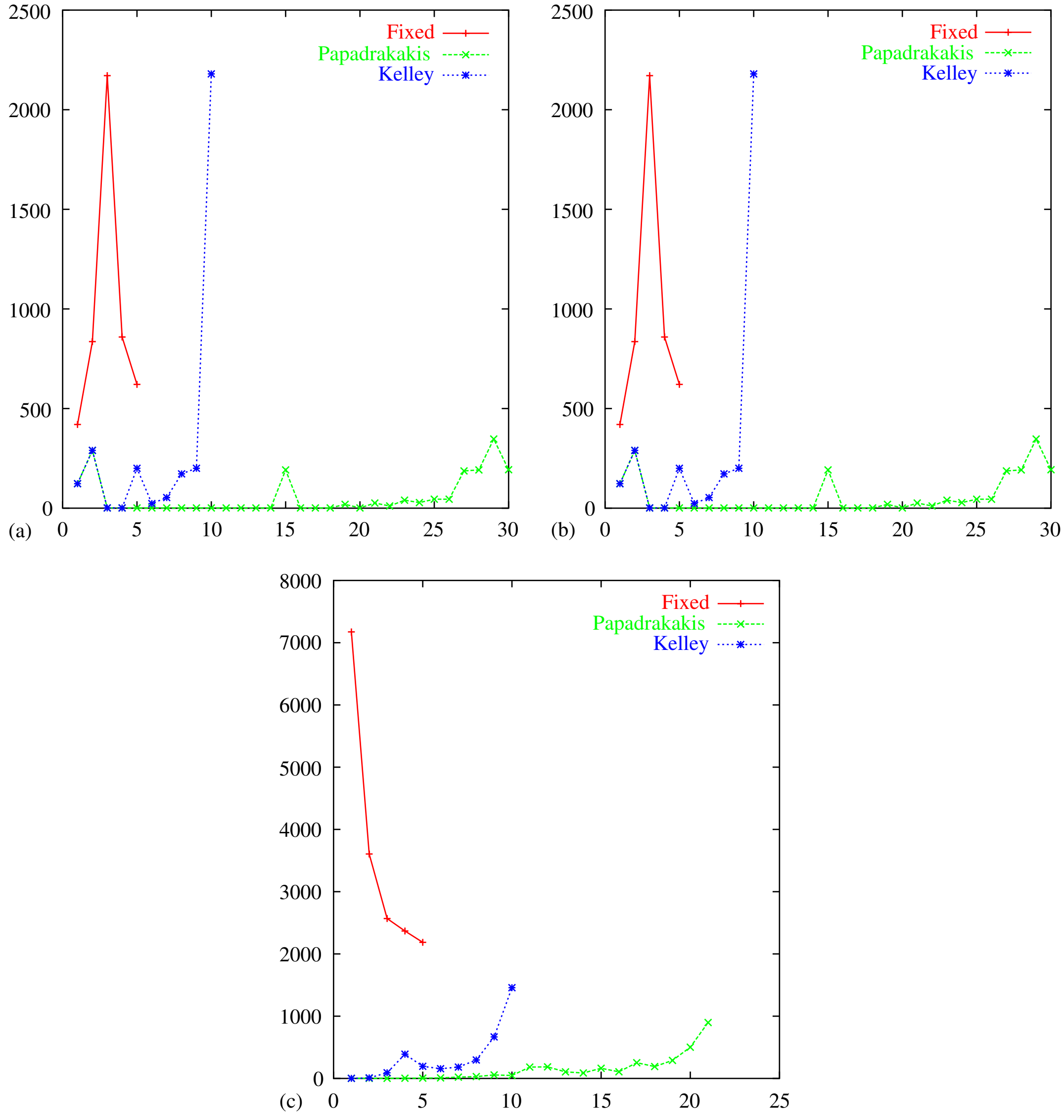


Figure 4. Number of linear iteration behaviour using different choices of the forcing terms—mesh with 512×512 cells—non-linear convection–diffusion problem: (a) LCD(10)$_A$ solution; (b) LCD(10)$_B$ solution; and (c) GMRES (10) solution.

than LCD when the number of restart vectors are increased. On the criteria to choose the forcing term for the inexact Newton method, we conclude that Papadrakakis criterion was the best option in all of our experiments.

The present numerical experiments are not exhaustive and certainly more tests are required. The apparent superiority of LCD_A in the finite difference test case should be further explored, and these will addressed in a forthcoming paper.

ACKNOWLEDGEMENTS

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

REFERENCES

1. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing: Boston, MA, 1996.
2. Yuan JY, Golub GH, Plemmons RJ, Cecílio WAG. Semiconjugate direction methods for real positive definite systems. *BIT Numerical Mathematics* 2004; **44**(1):189–207.
3. Catabriga L, Coutinho ALGA, Franca LP. Evaluating the LCD algorithm for solving linear systems of equations arising from implicit SUPG formulation of compressible flows. *International Journal for Numerical Methods in Engineering* 2004; **60**:1513–1534.
4. Catabriga L, Coutinho ALGA. Implicit SUPG solution of Euler equations using edge-based data structures. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**:3477–3490.
5. Valentim EC, Pessoa LM, Melotti BZ, Valli AMP, Catabriga L. Comparison between GMRES and LCD methods in the implementation of an inexact Newton method (in Portuguese). *10th Brazilian Congress of Thermal Engineering and Sciences CD-ROM*, Rio de Janeiro, Brazil, 2004; 1–12.
6. Dai Y, Yuan JY. Study on semi-conjugate direction methods for non-symmetric systems. *International Journal for Numerical Methods in Engineering* 2004; **60**:1383–1399.
7. Brooks AN, Hughes TJR. 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* 1982; **32**:199–259.
8. Kelley CT. *Iterative Methods for Linear and Non-linear Equations*. SIAM: Philadelphia, PA, 1995.
9. Pernice M, Walker HF. NITSOL: a Newton iterative solver for non-linear system. *SIAM Journal on Scientific Computing* 1998; **19**:302–318.
10. Papadrakakis M. *Solving Large-Scale Problems in Mechanics: The Development and Application of Computational Solution Procedures*. Wiley: London, 1993.
11. Shakib F, Hughes TJR, Johan Z. A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 1989; **65**:415–456.