An assessment of the LCD method for solving linear systems arising in SDFEM discretizations of convection–diffusion equations

Leopoldo P. Franca [1]

Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364, U.S.A.

Christopher Harder [2]

Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364, U.S.A.

Volker John [3]

FR 6.1 – Mathematik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, Germany

Abstract

A numerical assessment of the recently proposed left conjugate direction method (LCD) for solving non–symmetric systems of linear equations is presented. A restarted version of LCD is compared to GMRES with restart for systems which arise in the streamline–diffusion finite element discretization (SDFEM) of convection–dominated scalar convection–diffusion equations. We consider equations whose solution posses boundary or interior layers in two or three space dimensions. The equations are discretized with finite elements up to fourth order. The numerical studies show that GMRES with restart works in general more efficiently than LCD with restart.

Key words: convection–diffusion equations, iterative solvers for non–symmetric linear systems, left conjugate direction method, GMRES
1991 MSC: 65N22, 65N30

[1] E-mail: lfranca@math.cudenver.edu, The research of this author was partially funded by the NSF grant 0339107 and grant 0325314
[2] E-mail: charder@math.cudenver.edu, The research of this author was partially funded by the NSF grant 0339107
[3] corresponding author, E-mail: john@math.uni-sb.de, The research of this author

1 Introduction

The numerical solution of convection–dominated convection–diffusion equations is not only of interest by itself. It is also part of many numerical simulations of complex problems, like disperse systems, air and water pollution simulations, etc.

A standard discretization scheme for such equations are finite element methods. One crucial issue in the discretization is the stabilization of the dominating convective term. Popular approaches are the Streamline–Diffusion Finite Element Method (SDFEM), [3], or the use of residual free bubble (RFB) functions, [1,2]. For an overview on possible approaches, consult [11]. The discrete systems are in general large, sparse and non–symmetric. An iterative scheme becomes necessary for their efficient solution.

There are a number of established iterative schemes for non–symmetric linear systems of equations, like GMRES [13], BicGStab [14], QMR and their variants, see [12] for an overview. Recently, the left conjugate direction method (LCD) has been introduced in [15]. A natural question is how LCD behaves in comparison to established schemes for certain classes of non–symmetric linear systems of equations. This paper studies this question for systems which arise from the SDFEM discretization of scalar convection–diffusion equations in two and three space dimensions. The numerical studies compare a restarted version of LCD with a restarted GMRES method.

A comparison of LCD(restart) and GMRES(restart) has been started in [4]. In this paper, two–dimensional convection–diffusion equations have been considered which were discretized with first order finite elements on triangular grids. LCD(restart) was compared with GMRES(restart) without preconditioner and with diagonal preconditioner. The conclusion of the numerical studies in [4] show on the one hand that LCD(restart) required in general less iterations than GMRES(restart) but often a larger computing time. On the other hand, it is stated that "much more experiments are needed before reaching a final conclusion on its effectiveness". The present paper provides these numerical experiments. In comparison to [4], the study of LCD was extended into three directions:

- the consideration of three–dimensional convection–diffusion equations,
- the consideration of higher order finite element discretizations,
- the consideration of more preconditioners (SOR, ILU).

That in applications the solution of three–dimensional problems is of importance does not need any comment. The use of higher order discretizations has

was partially funded by the Deutsche Akademische Austauschdienst (D.A.A.D)

recently become quite popular and it is now a current topic of research. An interesting approach in this respect for convection–diffusion equations is the use of high order finite elements in subregions where the solution is smooth and of fine meshes in the other subregions, the so–called h–p adaptive method. It is well known that the algebraic properties of linear systems of equations are different for low and high order discretizations. Systems arising from high order discretizations are in general much harder to solve and the construction of efficient solvers for such systems is a current field of research. Last, the diagonal preconditioner is the simplest one and the use of other preconditioners will in general enhance the efficiency of the iterative solver. This will be demonstrated in the numerical studies in this paper.

The paper is organized as follows. In Section 2, the scalar convection–diffusion equations and their SDFEM discretization are introduced. The LCD method is presented and commented in Section 3. The numerical studies can be found in Section 4. Section 5 contains the final conclusions of these studies.

2 The governing equation and its finite element discretization

We consider scalar reaction–convection–diffusion equations of the form

$$\begin{aligned} -\nabla \cdot \varepsilon \nabla u + \mathbf{b} \cdot \nabla u + cu &= f \quad \text{in } \Omega \\ u &= g_D \quad \text{on } \partial\Omega_D \\ \varepsilon \frac{\partial u}{\partial \mathbf{n}} &= g_N \quad \text{on } \partial\Omega_N, \end{aligned} \tag{1}$$

where $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, is a bounded domain with boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. The diffusion parameter $\varepsilon > 0$, the convection field $\mathbf{b}$, the reaction c, the right hand side f, the Dirichlet boundary condition g_D and the Neumann boundary condition g_N are given functions, $\mathbf{n}$ denotes the outward normal on $\partial\Omega$.

For simplicity of presentation, let us now consider the case that $\partial\Omega = \partial\Omega_D$ and $g_D = 0$ on $\partial\Omega$. Defining $V = H_0^1(\Omega)$, the variational formulation of (1) reads as follows: Find $u \in V$ such that for all $v \in V$

$$(\varepsilon \nabla u, \nabla v) + (\mathbf{b} \cdot \nabla u + c, v) = (f, v), \tag{2}$$

where $(\cdot, \cdot)$ denotes the inner product in $L^2(\Omega)$.

We are interested in the convection–dominated case $\varepsilon << \|\mathbf{b}\|_{L^\infty}$. It is well known that in this case the standard Galerkin finite element solution of (2)

possesses spurious oscillations when the mesh cell Peclét number

$$Pe_K := \frac{\|\mathbf{b}\|_{L^\infty(K)} h_K}{2\varepsilon},$$

is larger than one, where h_K is the size of the mesh cell K. It becomes necessary to introduce some stabilization. There are different proposals in the literature, such as the Streamline–Diffusion Finite Element Method (SDFEM), (also known as Streamline–Upwind Petrov–Galerkin method (SUPG)), the Galerkin/least–squares method, RFB methods or multiscale methods. See [7] for a recent overview. The SDFEM, proposed in [3], might be the most popular of these. We will use it in the numerical studies.

Let $\mathcal{T}^h$ be a triangulation of Ω and let $V^h \subset V$ be a conforming finite element space. The SDFEM has the form: Find $u^h \in V^h$ such that for all $v^h \in V^h$

$$\begin{aligned}
&(\varepsilon \nabla u^h, \nabla v^h) + (\mathbf{b} \cdot \nabla u^h + c u^h, v^h) \\
&\quad + \sum_{K \in \mathcal{T}^h} \delta_K (-\varepsilon \Delta u^h + \mathbf{b} \cdot \nabla u^h + c u^h, \mathbf{b} \cdot \nabla v^h)_K \\
&= (f, v^h) + \sum_{K \in \mathcal{T}^h} \delta_K (f, \mathbf{b} \cdot \nabla v^h)_K. \qquad (3)
\end{aligned}$$

Different proposals for choosing the parameters δ_K can be found in the literature. We have used in our numerical studies, see [6],

$$\delta_K = \begin{cases} \dfrac{h_K}{2\|\mathbf{b}\|_{L^\infty(K)}} \left(\coth(Pe_K) - \dfrac{1}{Pe_K} \right) & \text{if } \|\mathbf{b}\|_{L^\infty(K)} \neq 0 \\ 0 & \text{if } \|\mathbf{b}\|_{L^\infty(K)} = 0. \end{cases}$$

In the numerical studies, finite element spaces $V^h = P_k$, $k \in \{1, 2, 3, 4\}$, on triangular grids, $k \in \{1, 2, 3\}$ on tetrahedral grids, and $V^h = Q_k$, $k \in \{1, 2, 3, 4\}$, in quadrilateral and hexahedral grids are considered.

3 The left preconditioned LCD method with restart

The application of the SDFEM leads to an algebraic system of equations of the form

$$Ax = b. \qquad (4)$$

The matrix A is sparse and in general non–symmetric. Let M be a non–singular matrix which will be applied as a left preconditioner to (4)

$$M^{-1}Ax = M^{-1}b. \qquad (5)$$

Also, the system matrix $M^{-1}A$ is in general non–symmetric.

The usual way of solving (5) consists of applying an iterative method, like GMRES [13] or BiCGStab [14], which can treat systems with a non–symmetric matrix. See [12] for an overview. In particular GMRES has become quite popular since the norm of the residual cannot increase during the iteration. This property results from the construction of the method where the j–th iterate is chosen in the Krylov subspace $x^{(0)} + \text{span}\{r^{(0)}, Ar^{(0)}, \ldots, A^{j-1}r^{(0)}\}$ such that the Euclidean norm of the residual becomes minimal. Here, $x^{(0)}$ and $r^{(0)}$ are the initial iterate and the initial residual, respectively. Recently, an alternative method has been proposed, the left conjugate direction method (LCD), [15]. Unlike GMRES, LCD does not rely upon the minimization of the residual in some Krylov subspace. Instead, a basis of the Krylov subspace is constructed where the solution of (4) can be determined by solving a triangular system. Both methods share the property that they require the storage of additional vectors in each iteration step. To keep the additional memory requirements reasonable, restarted versions have been proposed for both.

As the left preconditioned GMRES method with restart can be found elsewhere in the literature, e.g. in [12], we shall not present it here. However, we present the left preconditioned LCD method with restart in detail:

Algorithm 3.1 left preconditioned LCD method with restart for the solution of (5). Given A, M, b and x.

```
 1. z := M^{-1}(b - Ax)
 2. res := ||z||_2
 3. if (res ≤ resmin)
 4.    maxit := minit
 5. choose v^0
 6. j := 1
 7. while (j ≤ maxit)
 8.    for (i := 0; i < restart AND j ≤ maxit; i++, j++)
 9.       q^i := A^T M^{-T} v^i
10.       α := v^i · z
11.       α_1 := v^i · q^i
12.       α := α/α_1
13.       x := x + α v^i
14.       d := M^{-1} A v^i
15.       z := z - α d
16.       res := ||z||_2
17.       if (res ≤ resmin)
18.          return
19.       v^{i+1} := z
```

```
20.       for (k := 0; k <= i; k++)
21.         β^k := q^k · v^{i+1}
22.         if (k == i)
23.           β := q^i · v^i
24.         β := −β/β^k
25.         v^{i+1} := v^{i+1} + β v^k
26.       endfor
27.     endfor
28.     choose v^0
29. endwhile
30. return
```

From now on, we will speak of "GMRES(k)" instead of "left preconditioned GMRES method with restart" and of "LCD(k)" instead of "left preconditioned LCD method with restart", where k is the restart parameter.

Let $A = L + D + U$, where L is the strict lower triangle, D is the diagonal and U the strict upper triangle. In the numerical studies, we have used the following preconditioners:

1. no preconditioner: $M = I$ (identity),
2. diagonal or Jacobi preconditioner: $M = D$,
3. SOR preconditioner: $M = D + \omega L$, where $\omega = 1.5$ was used in all computations,
4. ILU preconditioner: $M = \tilde{L}\tilde{U}$, where $\tilde{L}\tilde{U}$ is an incomplete factorization of A.

We tested also the ILU$_\beta$ preconditioner with various values of $\beta > 0$, where all entries of the LU–factorization of A which do not fit into the sparsity pattern of A are multiplied with β and the modulo of this product is added to the diagonal of $\tilde{U}$. The results were in most cases similar to the ILU preconditioner.

Remark 3.2 1. GMRES(k) requires in each iteration one matrix–vector product and one application of the preconditioner. LCD(k) needs two of both operations, lines 9 and 14. The solution is, however, directly available in LCD(k), line 13, whereas it has to be computed in a post–processing step in GMRES(k). But altogether, one iteration of LCD(k) is more expensive than one GMRES iteration.

2. The LCD(k) method requires the multiplication with A^T and the application of the transposed preconditioner, line 9. The multiplication with A^T can be easily performed if the matrix is stored in the standard CSR sparse matrix structure. However, the application of the transposed preconditioner may cause difficulties, namely if there is not a simple representation of M. An example is

$$M = \mathrm{SSOR}(A)$$
$$= (D+\omega U)^{-1}\left((1-\omega)D-\omega L\right)(D+\omega L)^{-1}\left((1-\omega)D-\omega U\right), \quad (6)$$

with $\omega \in (0,2)$. The implementation of this preconditioner requires only a few lines of coding where its matrix representation is not used. However, to apply the transposed preconditioner, we do not see any other possibility than to use the transpose of (6). This will be quite inefficient and therefore we did not pursue the use of $M = \mathrm{SSOR}(A)$ in LCD(k). Another example of this kind would be that M is given by a multigrid method. In summary, there are preconditioners whose application in LCD(k) is rather difficult.

3. The required application of the transposed matrix is also problematic in Jacobian–free or matrix–free methods, e.g., see [10]. In these methods, the entries of A are not stored explicitly and a multiplication by A^T cannot be performed that easy way as described in the Remark 3.2.2.
4. GMRES(k) needs one vector with length proportional to the number of unknowns per iteration (the next basis vector of the Krylov subspace). In comparison, LCD(k) needs two vectors of this length per iteration ($\mathtt{q}^i$ and $\mathtt{v}^{i+1}$). Thus, also from the point of additional memory requirements, one iteration of LCD(k) is roughly twice as expensive as one iteration of GMRES(k).

   We performed our computations with the flexible finite element code *MooNMD*, [9]. With respect to the total memory requirements, one has to store also the (unstructured) grid information, the matrix, the right hand side and the solution vector. We found out that for small restart parameters, as they are used in the numerical studies presented in Section 4, the total memory requirements of LCD(k) are only insignificantly larger than that of GMRES(k) for the same restart parameter k.
5. The use of the SOR and the ILU preconditioner in LCD(k) requires the solution of triangular systems with the matrices $(D+\omega L)^T$, $\tilde{L}^T$ and $\tilde{U}^T$, respectively. The matrices $(D+\omega L)$, $\tilde{L}$ and $\tilde{U}$ are stored using the CSR sparse matrix structure. For the efficient solution of the triangular systems with the transposed matrices, it is necessary to have a CSR sparse matrix structure for the transposed matrices. This structure has to be computed in a pre–processing step. Its storage requires some additional memory.
6. Crucial for the efficiency of LCD(k) are the choices of v^0 in lines 5 and 28. Let x be the solution of (5) and x^0 be the initial iterate. It is shown in [15] that the optimal choice of v^0 is a vector which is parallel to $\bar{x} = x - x^0$. Then LCD converges in one iteration. However, the same holds true for GMRES if the initial iterate is parallel to the solution.

   One obtains from (5)

$$\bar{x} = A^{-1}(b - Ax^0).$$

In the numerical tests presented in this paper, we used the approximation

of $\bar{x}$

$$5.\ \mathtt{v}^0 := \mathtt{D}^{-1}(\mathtt{b} - \mathtt{A}\mathtt{x}^0).$$

In addition to this choice, we also tested also $v^0 := b$, as proposed in [4], and $v^0 := M^{-1}b$. The final number of iterations was in all tests similar for all three choices of the initial iterate.

Concerning the choice of v^0 after the restart, line 28, we used also

$$28.\ \mathtt{v}^0 := \mathtt{D}^{-1}(\mathtt{b} - \mathtt{A}\mathtt{x}),$$

where x is the current iterate. The best option found in [4] was $v^0 := v^{\text{restart}}$ (last vector v computed before the restart). We also tested this choice and we found that the number of iterations changed in general only slightly compared to the vector which we used.

Altogether, there were no substantial differences in the numerical results with all the different choices of v^0 in lines 5 and 28 which we tested.

4 The Numerical Studies

The computational studies have been performed with the code *MooNMD*, [8,9]. The GMRES(k) and LCD(k) methods have been assessed on two examples in two dimensions and on two three–dimensional examples. All examples were defined in $\Omega = (0,1)^d$, $d \in \{2,3\}$. In each dimension, we considered one example with interior layers (Examples 4.1 and 4.3) and one example with layers at the outflow boundary (Examples 4.2 and 4.4). Such layers are typical for solutions of convection–diffusion equations.

The coarsest quadrilateral grid (level 0) in two dimensions and the coarsest hexahedral grid in three dimensions consist just of one mesh cell. The coarsest triangular grid is obtained by dividing the unit square from $(0,0)$ to $(1,1)$. The coarsest tetrahedral grid is obtained by decomposing the unit cube into six tetrahedra. The numbers of degrees of freedom (d.o.f.), including Dirichlet nodes, are given for different finite element spaces on different levels in Tables 1 and 2. The number of non–zero matrix entries in the stiffness matrices on the finest levels from Tables 1 and 2 are given in Table 3.

The initial iterate on each level was set to be zero for all interior nodes and to the value of the boundary condition for nodes on Dirichlet boundaries. This is a natural situation in one–level methods for solving a linear system of equations. If a hierarchy of levels is available, a multigrid method should be used as solver or as preconditioner. We also performed studies with initial iterates which were prolongations from the next coarser grid. In these studies, we found the same conclusions concerning the efficiency of GMRES(k) and LCD(k) as will be reported below for the zero initial iterate.

Table 1
Degrees of freedoms in the two–dimensional examples.

level	P_1, Q_1	P_2, Q_2	P_3, Q_3	P_4, Q_4
3			625	1089
4		1089	2401	4225
5	1089	4225	9409	16641
6	4225	16641	37249	66049
7	16641	66049	148225	263169
8	66049	263169		
9	263169			

Table 2
Degrees of freedoms in the three–dimensional examples.

level	P_1, Q_1	P_2, Q_2	P_3, Q_3	Q_4
2			2197	4913
3		4913	15625	35937
4	4913	35937	117649	274625
5	35937	274625		
6	274625			

Table 3
Number of non–zero matrix entries on the finest levels given in Tables 1 and 2.

	P_1	Q_1	P_2	Q_2	P_3	Q_3	P_4	Q_4
2d	1831939	2354692	3006472	4181515	2495632	3668374	6137371	9398821
3d	3895491	6980352	7408330	16399545	5214680	13319110		54884481

We compare GMRES(k) and LCD(k) with the same restart parameter k. Numerical tests were performed for $k \in \{5, 10, 20\}$ which are commonly used values. Since the qualitative results are the same for all these parameters, we present only results for $k = 20$ in detail. Comments to the other restart parameters will be given below. The iterations were stopped if the Euclidean norm of the residual was less than 10^{-10}. We found out in the course of our numerical studies that changing the stopping criterion, e.g. to the Euclidean norm of the residual less than 10^{-6} or 10^{-8}, does not change the conclusions in comparing GMRES(k) and LCD(k). All computations were performed on a PC with a 3 GHz Intel Pentium 4 processor. The computing times are given in seconds.

We did not observe substantial differences in the behavior of the solvers between the linear systems of equations coming from finite elements on triangular and on quadrilateral grids. Therefore, we will present only a subset of the numerical results covering each polynomial degree once.

Example 4.1 2D solution with interior layers. The coefficients in (1) are given by $\varepsilon = 1e-8$, $\mathbf{b} = (-y, x)^T$, $c = 0$ and $f = 0$. Homogeneous Neumann boundary conditions $g_N = 0$ are prescribed on $\partial\Omega_N = \{(x, y) \in \partial\Omega \ : \ x = 0, y \in (0, 1)\}$. The Dirichlet boundary conditions on the rest of the boundary are as follows:

$$g_D = \begin{cases} 1 \text{ if } x \in (1/3, 2/3) \text{ and } y = 0, \\ 0 \text{ else.} \end{cases}$$

An illustration of the solution is presented in Figure 1. The number of iterations and the computing times for GMRES(20) and LCD(20) for computing the solution of this example are presented in Tables 4 – 7.

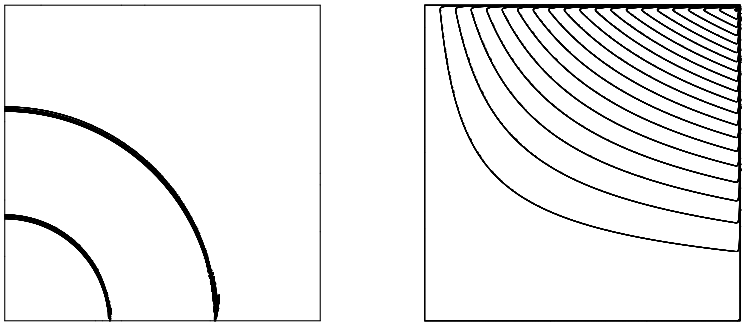

Fig. 1. Contour lines of the solutions of Example 4.1 (left) and Example 4.2 (right).

Example 4.2 2D solution with regular boundary layers. We consider (1) with $\varepsilon = 1e-8$, $\mathbf{b} = (2, 3)^T$, $c = 1$ and $\partial\Omega_D = \partial\Omega$. The right hand side and the boundary conditions are chosen such that

$$\begin{aligned} u(x, y) = xy^2 - y^2 \exp\left(\frac{2(x-1)}{\varepsilon}\right) - x \exp\left(\frac{3(y-1)}{\varepsilon}\right) \\ + \exp\left(\frac{2(x-1)+3(y-1)}{\varepsilon}\right) \end{aligned}$$

is the solution, see Fig. 1. The solution possesses typical regular boundary layers at $x = 1$ and $y = 1$. The behavior of GMRES(20) and LCD(20) in this example is presented in Tables 8 – 11.

Evaluation of the computational results. With respect to the computing time, GMRES(20) was in most cases better than LCD(20). Only for first order dis-

Table 4
Example 4.1, Q_1 finite element discretization on a quadrilateral mesh

	level	5		6		7		8		9	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	278	0.02	350	0.20	599	1.46	970	17.12	1698	126.59
	LCD(20)	181	0.04	221	0.19	329	1.47	534	13.16	958	97.12
diag.	GMRES(20)	203	0.03	284	0.24	430	1.50	701	15.28	1191	108.42
	LCD(20)	100	0.03	164	0.20	241	1.50	409	13.20	719	99.10
SOR	GMRES(20)	76	0.01	166	0.16	234	0.91	412	9.60	746	73.18
	LCD(20)	68	0.02	107	0.18	194	1.45	316	11.96	572	90.92
ILU	GMRES(20)	30	0.01	56	0.06	179	0.85	240	6.59	459	51.02
	LCD(20)	31	0.01	47	0.08	92	0.89	178	8.24	255	49.50

Table 5
Example 4.1, P_2 finite element discretization on a triangular mesh

	level	4		5		6		7		8	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	362	0.07	515	0.33	850	2.27	1431	26.70	2648	202.85
	LCD(20)	353	0.07	424	0.45	656	3.24	1178	31.31	2092	226.32
diag.	GMRES(20)	224	0.03	362	0.31	540	2.02	909	19.98	1634	150.26
	LCD(20)	173	0.04	244	0.34	371	2.57	626	21.93	1117	161.07
SOR	GMRES(20)	118	0.02	252	0.25	352	1.49	537	13.43	860	90.97
	LCD(20)	90	0.04	128	0.26	193	1.66	322	13.64	541	95.84
ILU	GMRES(20)	72	0.01	196	0.25	296	1.67	456	13.47	694	84.08
	LCD(20)	74	0.04	162	0.41	193	2.23	260	14.15	440	99.18

cretizations in Example 4.1, Table 4, LCD(20) was sometimes a little faster than GMRES(20). For higher order discretizations, GMRES(20) was always superior. Considering the ratio of computing times of LCD(20) and GMRES(20), one finds that this ratio tends to grow with the order of the finite element. Consequently, for fourth order finite elements, GMRES(20) was clearly faster. The smaller sensitivity of GMRES(20) with respect to the polynomial degree of the finite element can be also observed at the number of iterations. Whereas LCD(20) needs often much less iterations for linear and bilinear finite elements than GMRES(20), the numbers of iterations became similar for both schemes for higher order discretizations. Since one iteration of

Table 6
Example 4.1, Q_3 finite element discretization on a quadrilateral mesh

	level	3		4		5		6		7	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	318	0.01	456	0.27	721	1.51	1128	12.89	2037	106.49
	LCD(20)	319	0.07	434	0.40	681	2.61	1008	19.21	1794	143.98
diag.	GMRES(20)	218	0.02	325	0.23	469	1.26	743	10.76	1330	83.03
	LCD(20)	195	0.03	279	0.34	369	1.94	586	14.31	973	101.05
SOR	GMRES(20)	109	0.02	213	0.18	287	1.10	458	8.30	719	56.89
	LCD(20)	110	0.04	150	0.28	214	1.75	296	10.97	466	72.99
ILU	GMRES(20)	diverges									
	LCD(20)	diverges									

Table 7
Example 4.1, P_4 finite element discretization on a triangular mesh

	level	3		4		5		6		7	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	834	0.17	1167	1.06	1867	7.04	3747	84.89	6100	548.88
	LCD(20)	937	0.29	1577	2.62	2217	15.28	3557	121.76	6439	903.14
diag.	GMRES(20)	411	0.11	592	0.64	837	4.11	1319	36.07	2577	279.88
	LCD(20)	399	0.18	612	1.32	1017	9.51	1421	62.98	2344	424.20
SOR	GMRES(20)	227	0.07	320	0.51	475	3.15	714	24.44	1203	158.79
	LCD(20)	212	0.12	286	1.02	388	5.57	573	37.67	901	239.39
ILU	GMRES(20)	diverges									
	LCD(20)	diverges									

GMRES(20) is considerably cheaper than one iteration of LCD(20), the corresponding computing times with GMRES(20) are much better for higher order discretizations. For smaller restart parameters $k \in \{5, 10\}$, we obtained qualitatively the same results as for $k = 20$. We could observe that the superiority of GMRES(k) in comparison to LCD(k) was even larger for $k \in \{5, 10\}$.

Concerning the preconditioners, one can observe that the diagonal preconditioner improved the performance of the solvers, measured in computing time, only in Example 4.1. In Example 4.2, no preconditioning was in general better than applying this preconditioner. For the systems coming from the first

Table 8
Example 4.2, P_1 finite element discretization on a triangular mesh

	level	5		6		7		8		9	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	100	0.01	168	0.08	310	0.70	426	7.39	736	52.88
	LCD(20)	85	0.02	179	0.15	237	0.94	377	8.82	644	62.67
diag.	GMRES(20)	111	0.02	198	0.15	340	1.07	477	9.99	801	68.44
	LCD(20)	97	0.03	184	0.22	260	1.43	404	13.24	680	88.38
SOR	GMRES(20)	48	0.01	107	0.08	174	0.61	276	5.98	470	43.78
	LCD(20)	53	0.01	98	0.15	167	1.11	236	8.33	352	51.85
ILU	GMRES(20)	22	0.01	39	0.03	84	0.36	200	5.05	326	34.46
	LCD(20)	28	0.01	49	0.10	100	0.83	168	6.99	233	40.61

Table 9
Example 4.2, Q_2 finite element discretization on a quadrilateral mesh

	level	4		5		6		7		8	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	107	0.02	244	0.17	376	1.13	536	10.36	923	75.80
	LCD(20)	115	0.04	232	0.32	320	1.73	501	14.77	831	100.77
diag.	GMRES(20)	110	0.03	283	0.29	380	1.46	595	14.85	1026	106.05
	LCD(20)	127	0.03	241	0.44	353	2.74	517	20.82	917	148.16
SOR	GMRES(20)	63	0.01	116	0.14	225	1.14	346	9.93	560	63.47
	LCD(20)	65	0.03	103	0.25	197	2.09	324	16.49	466	95.29
ILU	GMRES(20)	37	0.02	72	0.11	125	0.83	260	9.05	355	51.84
	LCD(20)	40	0.02	68	0.24	128	1.91	193	13.03	305	83.74

and second order discretizations, ILU was the most efficient preconditioner. However, ILU failed in general for higher order discretizations. Obviously, the higher the order of the discretization becomes the worse is the approximation of A^{-1} by $(\mathrm{ILU}(A))^{-1}$. Failures of ILU preconditioning in other situations, e.g., with matrices arising from linearized Navier–Stokes equations, are observed also in [5]. The best preconditioner for the solution of the systems from third and fourth order discretizations was in general SOR. It can be observed that both iterative schemes behaved similar with respect to the preconditioners studied, i.e., there was no case in which a certain preconditioner improved one scheme much more than the other one.

Table 10
Example 4.2, P_3 finite element discretization on a triangular mesh

	level	3		4		5		6		7	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	117	0.01	181	0.09	299	0.51	401	4.05	680	32.33
	LCD(20)	116	0.01	208	0.13	253	0.80	421	6.24	660	46.17
diag.	GMRES(20)	119	0.01	204	0.10	314	0.76	478	5.95	743	41.28
	LCD(20)	126	0.03	187	0.17	329	1.42	457	8.85	697	65.43
SOR	GMRES(20)	74	0.01	99	0.07	198	0.56	302	4.58	471	30.79
	LCD(20)	77	0.02	107	0.12	154	0.94	256	7.19	392	47.58
ILU	GMRES(20)	80	0.01	118	0.13	207	0.80	369	6.93	639	52.60
	LCD(20)	80	0.03	121	0.21	226	1.93	438	16.85	760	122.41

Table 11
Example 4.2, Q_4 finite element discretization on a quadrilateral mesh

	level	3		4		5		6		7	
prec.		ite	time	ite	time	ite	time	ite	time	ite	time
no	GMRES(20)	325	0.11	379	0.43	587	2.81	1016	25.98	1775	199.61
	LCD(20)	299	0.13	434	0.94	639	5.51	1079	46.15	1902	330.81
diag.	GMRES(20)	314	0.10	384	0.53	582	3.32	1071	32.51	1801	227.92
	LCD(20)	314	0.18	447	1.20	621	6.80	1039	55.18	1921	414.39
SOR	GMRES(20)	210	0.08	300	0.60	419	3.47	667	26.94	1074	184.73
	LCD(20)	215	0.18	302	1.43	422	8.15	690	59.82	1141	398.96
ILU	GMRES(20)	diverges									
	LCD(20)	diverges									

4.2 The three–dimensional examples

In the three–dimensional examples, we found only small differences for the behavior of the solvers on tetrahedral and hexahedral meshes. For this reason, we will present for each examples only one result for each polynomial degree of the finite element spaces.

Example 4.3 3D solution with interior layers. This example is given by the coefficients $\varepsilon = 1e-8$, $\mathbf{b} = (0.5, 0.75, -0.75)^T$, $c = 0$ and $f = 0$ in (1).

Homogeneous Neumann boundary conditions $g_N = 0$ are described on the outflow boundary $\partial\Omega_N = \{(x, y, z) \in \partial\Omega \ : \ x = 1 \text{ or } y = 1 \text{ or } z = 0\}$. The Dirichlet boundary condition on the rest of the boundary is given by

$$g_D = \begin{cases} 1 \text{ if } (x, y, z) \in \partial\Omega \in \text{ and } \sqrt{(x - 1/4)^2 + y^2 + (z - 3/4)^2} < 0.1, \\ 0 \text{ else.} \end{cases}$$

In this example, an inflow of the form of a circular disk is transported through the domain to the outflow boundary, see Figure 2. The behavior of GMRES(20) and LCD(20) is presented in Tables 12 – 15.

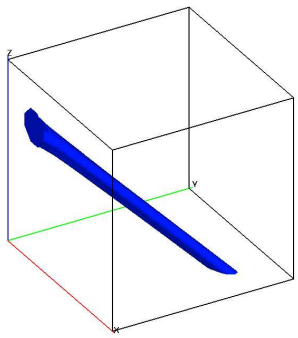

Fig. 2. Isosurface $u = 1$ of the solution of Example 4.3.

Table 12
Example 4.3, Q_1 finite element discretization on a hexahedral mesh

	level	4		5		6	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	86	0.08	163	1.80	290	30.11
	LCD(20)	67	0.11	90	1.70	152	24.52
diag.	GMRES(20)	107	0.16	195	2.71	349	43.88
	LCD(20)	81	0.21	115	2.75	189	38.53
SOR	GMRES(20)	45	0.09	59	1.00	164	25.16
	LCD(20)	46	0.19	60	2.29	102	32.02
ILU	GMRES(20)	24	0.08	30	0.90	59	13.25
	LCD(20)	25	0.17	31	1.83	57	26.61

Example 4.4 3D solution with regular boundary layers. In this example, we consider (1) with $\varepsilon = 1e - 8$, $\mathbf{b} = (2, 3, 4)^T$, $c = 1$ and $\partial\Omega_D = \partial\Omega$. The right hand side f and the Dirichlet boundary conditions have been chosen such that

Table 13
Example 4.3, P_2 finite element discretization on a tetrahedral mesh

	level	3		4		5	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	269	0.27	392	4.46	524	55.66
	LCD(20)	281	0.56	361	6.40	521	86.70
diag.	GMRES(20)	220	0.29	347	5.03	533	66.02
	LCD(20)	216	0.58	321	8.14	443	91.92
SOR	GMRES(20)	115	0.21	176	3.20	293	46.28
	LCD(20)	119	0.52	161	6.50	229	76.01
ILU	GMRES(20)	124	0.34	218	5.70	325	72.15
	LCD(20)	127	0.80	195	11.38	290	138.45

Table 14
Example 4.3, P_3 finite element discretization on a tetrahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	417	0.27	485	2.44	858	46.70
	LCD(20)	380	0.44	537	5.14	937	85.65
diag.	GMRES(20)	286	0.21	477	2.82	725	45.35
	LCD(20)	281	0.39	447	5.39	676	73.67
SOR	GMRES(20)	187	0.19	283	2.45	383	32.53
	LCD(20)	203	0.50	299	6.66	361	71.83
ILU	GMRES(20)	diverges					
	LCD(20)	diverges					

$$u(x,y,z) = \left(x - \exp\left(\frac{2(x-1)}{\varepsilon}\right)\right)\left(y^2 - \exp\left(\frac{3(y-1)}{\varepsilon}\right)\right)\left(z^3 - \exp\left(\frac{4(z-1)}{\varepsilon}\right)\right)$$

is the solution of (1). The solution possesses regular boundary layers at $x = 1$, $y = 1$ and $z = 1$. This is a three–dimensional version of Example 4.2. The computational results obtained with GMRES(20) and LCD(20) are presented in Tables 16 – 19.

Table 15
Example 4.3, Q_4 finite element discretization on a hexahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	714	3.14	874	33.6	1663	559.89
	LCD(20)	627	5.31	1375	104.73	1657	1070.47
diag.	GMRES(20)	732	3.43	1312	55.17	1117	400.08
	LCD(20)	654	6.13	839	69.17	1230	852.53
SOR	GMRES(20)	365	2.92	534	36.84	726	427.9
	LCD(20)	379	8.34	557	105.78	769	1217.19
ILU	GMRES(20)	diverges					
	LCD(20)	diverges					

Table 16
Example 4.4, P_1 finite element discretization on a tetrahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	59	0.04	82	0.71	147	12.77
	LCD(20)	62	0.07	87	1.19	132	16.73
diag.	GMRES(20)	71	0.08	112	1.33	171	18.02
	LCD(20)	73	0.13	113	2.07	173	29.09
SOR	GMRES(20)	39	0.05	50	0.64	68	8.02
	LCD(20)	39	0.10	50	1.20	69	14.83
ILU	GMRES(20)	22	0.03	30	0.51	40	6.26
	LCD(20)	22	0.08	30	1.04	41	12.68

Evaluation of the computational results. The evaluation of the three–dimensional computational results shows a picture similar to that of the two–dimensional results. With respect to the computing time, LCD(20) was occasionally competitive with GMRES(20) for the first order discretizations, Table 12. For higher order discretizations, GMRES(20) was clearly better. In addition, we could observe that the superiority of GMRES(k) in comparison to LCD(k) increased for $k \in \{5, 10\}$. The best preconditioner for the solution of the systems coming from the first order discretizations was ILU. This preconditioner failed often for higher order discretizations. SOR was the best preconditioner for higher order discretizations on tetrahedral grids. For higher order dis-

Table 17
Example 4.4, Q_2 finite element discretization on a hexahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	89	0.14	102	1.64	176	26.06
	LCD(20)	96	0.28	111	3.14	159	39.72
diag.	GMRES(20)	105	0.19	128	2.36	199	32.56
	LCD(20)	110	0.36	134	4.54	199	58.39
SOR	GMRES(20)	56	0.17	72	1.96	92	22.52
	LCD(20)	57	0.38	72	4.89	99	56.86
ILU	GMRES(20)	34	0.37	44	3.89	55	37.45
	LCD(20)	34	0.58	43	6.55	59	68.23

Table 18
Example 4.4, P_3 finite element discretization on a tetrahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	140	0.08	200	0.88	257	13.16
	LCD(20)	145	0.13	197	1.69	244	21.12
diag.	GMRES(20)	127	0.09	211	1.14	303	18.22
	LCD(20)	133	0.12	205	2.22	337	35.31
SOR	GMRES(20)	77	0.07	117	0.96	161	13.47
	LCD(20)	82	0.18	121	2.39	167	31.33
ILU	GMRES(20)	640	0.84	diverges			
	LCD(20)	diverges					

cretizations on hexahedral grids, often no preconditioning was the best choice among the approaches which were studied in this paper.

5 Concluding remarks

The paper presented an assessment of the LCD(restart) algorithm for solving non–symmetric linear systems of equations which arise in the SDFEM discretization of convection–dominated convection–diffusion equations in two

Table 19
Example 4.4, Q_4 finite element discretization on a hexahedral mesh

	level	2		3		4	
prec.		ite	time	ite	time	ite	time
no	GMRES(20)	500	1.81	520	18.59	718	235.51
	LCD(20)	506	3.70	538	37.94	657	410.33
diag.	GMRES(20)	348	1.40	507	19.69	789	276.17
	LCD(20)	372	2.95	581	44.36	699	469.43
SOR	GMRES(20)	234	1.59	345	22.54	483	280.00
	LCD(20)	267	4.73	362	62.05	520	790.32
ILU	GMRES(20)	diverges					
	LCD(20)	diverges					

and three space dimensions. LCD(restart) was compared to GMRES(restart). With respect to the computing times, GMRES(restart) was more efficient than LCD(restart) for all systems which came from second or higher order finite element spaces. LCD(restart) was occasionally competitive to GMRES(restart) for solving systems coming from first order discretizations. Among the studied preconditioners, ILU worked best for low order discretizations but it failed in general for higher order discretizations. The best preconditioner for systems generated from high order discretizations was in general SOR.

Since the preconditioned LCD method requires the application of the transposed preconditioner, the application of several preconditioners which is easily with GMRES, like SSOR or multigrid methods, becomes much harder with LCD. In particular, the application of a multigrid preconditioner enhances the performance of GMRES considerably. Together with the numerical results obtained in this paper, one must conclude that LCD, in its current form, does not present an efficient alternative to GMRES (or other iterative schemes which perform comparable to GMRES) for the solution of linear systems arising in SDFEM discretizations of convection–dominated convection diffusion equations. The development of a transposed–free version of LCD in the future might change this conclusion.

References

[1] C. Baiocchi, F. Brezzi, and L.P. Franca. Virtual bubbles and the galerkin/least squares method. *Comp. Meth. Appl. Mech. Engrg.*, 105:125 – 142, 1993.

[2] F. Brezzi, L.P. Franca, T.J.R. Hughes, and A. Russo. $b = \int g$. *Comp. Meth. Appl. Mech. Engrg.*, 145(329 - 339), 1997.

[3] A.N. Brooks and T.J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 32:199–259, 1982.

[4] L. Catabriga, A.L.G.A. Coutinho, and L.P. Franca. Evaluating the LCD algorithm for solving linear systems of equations arising from implicit supg formulation of compressible flows. *Int. J. Numer. Meth. Engrg.*, 60:1513 – 1534, 2004.

[5] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *J. Comp. Appl. Math.*, 86:387 – 414, 1997.

[6] I. Christie, D.F. Griffiths, A.R. Mitchell, and O.C. Zienkiewicz. Finite element methods for second order differential equations with significant first derivatives. *Int. J. Numer. Meth. Engrg.*, 10:1389 – 1396, 1976.

[7] T.J.R. Hughes, G. Scovazzi, and L.P. Franca. Mulitiscale and stabilized methods. In E. Stein, R. de Borst, and T.J.R. Hughes, editors, *Encyclopedia of Computational Mechanics*, volume 3, chapter 2. John Wiley & Sons, 2004.

[8] V. John. *Large Eddy Simulation of Turbulent Incompressible Flows. Analytical and Numerical Results for a Class of LES Models*, volume 34 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag Berlin, Heidelberg, New York, 2004.

[9] V. John and G. Matthies. MooNMD - a program package based on mapped finite element methods. *Comput. Visual. Sci.*, 6:163 – 170, 2004.

[10] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193:357 – 397, 2004.

[11] H.-G. Roos, M. Stynes, and L. Tobiska. *Numerical Methods for Singularly Perturbed Differential Equations*. Springer, 1996.

[12] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2nd edition, 2003.

[13] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856 – 869, 1986.

[14] H.A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of non-symmetric linear systems. *SIAM Sci. Stat. Comp.*, 12:631 – 644, 1992.

[15] J.Y. Yuan, G.H. Golub, R.J. Plemmons, and W.A.G. Cecilio. Semi-conjugate direction methods for real positive definite systems. *BIT Numerical Mathematics*, 44:189 – 207, 2004.