CGMN Revisited: Robust and Efficient Solution of Stiff Linear Systems Derived from Elliptic Partial Differential Equations

DAN GORDON
University of Haifa
and
RACHEL GORDON
The Technion–Israel Institute of Technology

Given a linear system $Ax = b$, one can construct a related "normal equations" system $AA^Ty = b$, $x = A^Ty$. Björck and Elfving have shown that the SSOR algorithm, applied to the normal equations, can be accelerated by the conjugate gradient algorithm (CG). The resulting algorithm, called CGMN, is error-reducing and it always converges (in theory), even when the equation system is inconsistent and/or nonsquare. SSOR on the normal equations is equivalent to the Kaczmarz algorithm (KACZ), with a fixed relaxation parameter, run in a double (forward and backward) sweep on the original equations. CGMN was tested on nine well-known large and sparse linear systems obtained by central-difference discretization of elliptic convection-diffusion partial differential equations (PDEs). Eight of the PDEs were strongly convection-dominated, and these are known to produce very stiff systems with large off-diagonal elements. CGMN was compared with some of the foremost state-of-the art Krylov subspace methods: restarted GMRES, Bi-CGSTAB and CGS. These methods were tested both with and without various preconditioners. CGMN converged in all the cases, while none of the above algorithm/preconditioner combinations achieved this level of robustness. Furthermore, on varying grid sizes, there was only a gradual increase in the number of iterations as the grid was refined. On the eight convection-dominated cases, the initial convergence rate of CGMN was better than all the other combinations of algorithms and preconditioners, and the residual decreased monotonically. The CGNR algorithm was also tested, and it was as robust as CGMN, but slower.

Categories and Subject Descriptors: 15A06 [**Linear and multilinear algebra**]: Linear equations; 65F10 [**Numerical linear algebra**]: Iterative methods for linear systems; 65F50 [**Numerical linear algebra**]: Sparse matrices; 65N12 [**Partial differential equations**]: Stability and convergence of numerical methods.

General Terms: Algorithms, Performance, Reliability, Theory.
Additional Key Words and Phrases: CGMN, CGNR, conjugate-gradient, convection-dominated, elliptic equations, Kaczmarz, linear systems, normal equations, partial differential equations, row projections, SOR, SSOR, sparse linear systems, stiff equations.

1. INTRODUCTION

Large sparse linear systems are obtained when solving various problems in scientific, engineering and biomedical applications. Furthermore, many solution methods for non-linear problems, optimization problems and eigenvalue problems, require repetitive solutions of

Author's address: D. Gordon, Dept. of Computer Science, University of Haifa, Haifa 31905, Israel. Email: `gordon@cs.haifa.ac.il`
R. Gordon, Dept. of Aerospace Engineering, The Technion–Israel Inst. of Technology, Haifa 32000, Israel. Email: `rgordon@tx.technion.ac.il`
Note: This preprint contains the figures in color and minor corrections.

linear systems as an intermediate step. Direct solvers are usually preferred because of their robustness, but they cannot handle the huge matrices that often occur in practice. Iterative solvers are therefore the only viable method in many cases, but they often fail on huge and ill-conditioned systems. The approach taken in many cases is to combine one of the many available solvers with a suitable preconditioner which is sometimes tailor-fit to the problem at hand. However, such a solver/preconditioner combination may be hard to find, so there is always a need for robust general-purpose methods that are applicable to a wide range of problems.

This paper concentrates on large sparse linear systems arising from the discretization of elliptic convection-diffusion partial differential equations (PDEs), using the central-difference scheme (CDS). In this class of problems, some of the most challenging cases occur when the convection term is large, leading to large off-diagonal elements in the associated system matrix. Such systems arise frequently in computational fluid dynamics (CFD) applications when the Péclet number is high—see [Ferziger and Perić 2002], and [Gresho and Sani 1998]. Our main finding is that the CGMN algorithm of [Björck and Elfving 1979] is a very robust and efficient solution method for a wide range of such linear systems.

Krylov subspace methods are widely considered to be the leading techniques for solving sparse linear systems. Among the better-known of these methods are the conjugate gradient (CG) [Hestenes and Stiefel 1952], CGNR and CGNE [Saad 2003, §8.3.1], Bi-CG [Lanczos 1952; Fletcher 1976], Bi-CGSTAB (stabilized Bi-CG) [van der Vorst 1992], CGS (CG-Squared) [Sonneveld 1989], and GMRES [Saad and Schultz 1986]. All these methods can be used with and without preconditioners, but the robustness problem is still a hindrance in many applications.

Given a system of linear equations

$$Ax = b, \tag{1}$$

one can construct two related "normal equations" systems

$$A^TAx = A^Tb, \tag{2}$$

$$AA^Ty = b, \;\; x = A^Ty. \tag{3}$$

An important property of the system matrices of (2) and (3) is that their diagonal elements are relatively large, even when A has large large off-diagonal elements. Hence, solution methods based on the normal equations tend to be very robust. Of course, there may be a price to pay because their condition number is the square of the condition number of A, but CG-acceleration can be used to offset this problem, as will be shown.

In a seminal paper, [Björck and Elfving 1979] present several CG-type acceleration techniques for projection algorithms. One of their algorithms, called CGMN, is a CG acceleration of the SSOR algorithm [Saad 2003, §4.1] applied to the normal equations system (3). It is well known that SSOR applied to (3) is equivalent to the Kaczmarz algorithm (KACZ), with a fixed relaxation parameter, run in a double (forward and backward) sweep on the original equations. In [Björck and Elfving 1979], CGMN is also referred to as an SSOR preconditioning of CG. CGMN converges even when the system (1) is inconsistent and/or nonsquare. This is due to the fact that the normal equations systems are square and KACZ converges in a cyclic manner even on inconsistent systems—see [Tanabe 1971].

Our interest in CGMN grew out of our work on accelerating our block-parallel CARP algorithm [Gordon and Gordon 2005] using the same principles as in CGMN. CARP is equivalent to KACZ in some superspace, but taken with cyclic relaxation parameters. A generalization of CGMN, called CGMNC, was developed in [Gordon and Gordon 2008] to enable the use of cyclic relaxation parameters. CGMNC in the superspace is mathematically equivalent to a CG-acceleration of CARP, called CARP-CG. This algorithm is as robust as CARP, but converges significantly faster. When applied to linear systems derived from strongly convection-dominated elliptic PDEs, CARP-CG is very competitive with several prominent Krylov subspace algorithms combined with various preconditioners.

These positive results led to an examination of the original CGMN on a single processor, and the results are presented in this paper. CG-acceleration of block projection schemes was studied quite extensively in the context of parallelism; see [Arioli et al. 1992; Arioli et al. 1995; Bramley et al. 1990; Bramley and Sameh 1991; 1992; Kamath and Sameh 1989]. These techniques are based on applying the CG-acceleration methods of [Björck and Elfving 1979] to the block-projection methods of [Elfving 1980]. However, CGMN itself has received little attention as a sequential solution method for linear systems derived from PDEs. In [Kamath and Weeratunga 1990; 1992], CG-accelerated block-SSOR is compared with CGMN (which is referred to as a CG-acceleration of "symmetric Kaczmarz"), using only the unity relaxation parameter. They conclude that on a single processor, CGMN is preferable, whereas the CG-accelerated block-SSOR is preferable in a parallel processing environment.

In this study, CGMN is compared with restarted GMRES, CGS, and Bi-CGSTAB, which were all tested with and without a number of standard preconditioners. It was also compared with CGNR. Note that CGNR can also be regarded as a CG-acceleration of the Cimmino algorithm [Cimmino 1938]. The algorithms were tested on nine well-known large and sparse linear systems obtained by CDS discretization of elliptic convection-diffusion PDEs. Eight of the PDEs were strongly convection-dominated, and these are known to produce very stiff systems with large off-diagonal elements. CGMN converged in all the cases, whereas most of the others failed on the stiff problems. Furthermore, the convergence of its residual on the eight stiff problems was monotonic, whereas CGS and Bi-CGSTAB were oscillatory to some extent. CGMN's robustness was also apparent on various grid sizes, with the number of iterations increasing only gradually with grid refinement. CGNR was as robust as CGMN, but it was slower.

In addition to its robustness, our runtime results with CGMN showed that it was particularly efficient on the eight problems derived from the PDEs with large convection terms. On the one test case with a small convection term, the convergence rate of CGMN was mediocre, while most of the other methods performed quite well on this case. From these results we can conclude that CGMN is particularly advantageous on systems with large off-diagonal elements.

Note that in this study we are primarily interested in the numerical solution of the algebraic equation system derived from the PDEs by central-difference discretization (on a uniform mesh). In some cases, the algebraic equations cannot provide accurate solutions to the PDEs. For example, if the mesh Péclet number is large, then the (exact) solution of the derived linear system is sometimes oscillatory and it may diverge widely from the analytic solution of the PDE at some grid points—see [Ferziger and Perić 2002, Ch. 3,4], [Gresho and Sani 1998], and also [Saad 2003, §2.2.4]. There are several approaches to this problem; e.g., the use of upwind schemes, hybrid methods combining upwind schemes with CDS, variable meshes, higher-order schemes, and more. Upwind schemes are less accurate since they are only first-order and introduce artificial diffusion—see [Ferziger and Perić 2002]. Even so, it is interesting to compare the solutions obtained by CGMN with the analytic solution of the PDEs. Our results in this regard are mixed: in four cases we obtained excellent results, in four cases the results are reasonable for some applications (such as CFD), and in one case, they are poor.

The rest of this paper is organized as follows. Section 2 presents some essential background and the CGMNC algorithm. Sections 3 and 4 describe the setup of the numerical experiments and the results, and Section 5 concludes with a summary and some future research directions.

2. MATHEMATICAL BACKGROUND

This section presents the background leading to the CGMN algorithm. Actually, we will present CGMNC, which is CGMN extended to allow for cyclic relaxation parameters. Even though our experiments deal only with CGMN, cyclic relaxation parameters are potentially useful for certain problems. For example, in solving the Navier-Stokes equations for boundary-layer problems in CFD, different regions may have widely-varying equation coefficients, so it may be advantageous to use different relaxation parameters in the different regions.

Throughout the rest of the paper, we assume that all vectors are column vectors, and we use the following notation: $\langle p, q\rangle$ denotes the dot product of two vectors p and q, which is also p^Tq. By $\|\bullet\|$ we denote the L_2-norm of a vector. If A is an $m\times n$ matrix, we denote by $a_{i\bullet}$ the ith row-vector of A; i.e., $a_{i\bullet} = (a_{i1},\ldots,a_{in})^T$. Unless noted otherwise, we shall assume that the matrix A of equation system (1) is of dimension $m\times n$ (m rows and n columns). Throughout the rest of the paper we assume that every column of A is nonzero. Such a column of zeros, if it existed, can be removed as a preliminary step since it corresponds in effect to coefficients of a "fictitious" variable, i.e., one whose value can be arbitrary. Also, we can assume that every equation contains at least one nonzero coefficient.

One of the earliest algorithms, denoted as KACZ, is due to [Kaczmarz 1937]. KACZ starts from an arbitrary point $x^0 \in \mathbb{R}^n$ as the initial iterate, and in each step, the current iterate is projected orthogonally towards a hyperplane defined by one of the equations. The hyperplanes are chosen in cyclic order. Each projection may involve a relaxation parameter λ which determines the extent of the projection towards the hyperplane: the projection is either in front of the hyperplane, exactly on the hyperplane, or beyond the hyperplane, according to whether $\lambda < 1$, $\lambda = 1$, or $\lambda > 1$, respectively. We shall henceforth assume that every relaxation parameter λ satisfies the inequalities $0 \le \lambda \le 2$.

The term "cyclic relaxation parameters" means that for every $1 \le i \le m$, there is a relaxation parameter λ_i which is always used with the projection towards the ith hyperplane. In our application of KACZ, the projections are always performed with an entire sweep (or pass) of all the equations, and as far as we are aware, this is the standard practice in all applications of KACZ. Our formulation of KACZ incorporates this practice in the algorithm. KACZ with cyclic relaxation parameters $\lambda_1,\ldots,\lambda_m$, is the following:

ALGORITHM 1. **(KACZ):**

```
set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
for $k = 0,1,2,\ldots$ until convergence do
```

```
        set $y^0 = x^k$.
        for $i = 1,2,\dots,m$ do
            $y^i = y^{i-1} + \lambda_i \frac{b_i - \langle a_{i\bullet}, y^{i-1}\rangle}{\|a_{i\bullet}\|^2} a_{i\bullet}$    (4)
        enddo
        set $x^{k+1} = y^m$.
    enddo
```

KACZ has been studied very extensively, both theoretically and experimentally. In the context of image reconstruction from projections in computerized tomography, it is also known as ART (algebraic reconstruction technique)—see [Herman 1980]. Its convergence with relaxation parameters, for consistent systems, was shown in [Herman et al. 1978] and [Trummer 1981]. In [Tanabe 1971], it is shown that when the system is inconsistent, KACZ with a fixed relaxation parameter converges *cyclically*; this means that for each hyperplane, the sequence of projections onto that hyperplane converges to a limit. This result was extended in [Eggermont et al. 1981] to the case of cyclic relaxation parameters, and it holds true even when the system (1) is nonsquare. This particular result is essential to the extension of CGMN to CGMNC. See also [Elfving 2004] and the references therein.

The division by $\|a_{i\bullet}\|^2$ in KACZ means that KACZ is a *geometric* algorithm in the following sense: the sequence of iterates produced by KACZ depends only on the hyperplanes defined by the equations, and not on any particular algebraic representation. This is due to the fact that if we divide the ith equation by some constant $c_i \neq 0$, then the sequence of iterates produced by KACZ is unchanged, i.e., KACZ does not depend on the scaling of the equations. See [Gordon and Mansour 2007] for an application of this property. We shall henceforth assume that the equations are scaled by normalizing them, i.e., for $1 \le i \le m$, the ith equation has been divided by $\|a_{i\bullet}\|$. In practice, this scaling makes KACZ more efficient. Equation (4) in Algorithm 1 can now be replaced by

$$y^i = y^{i-1} + \lambda_i \left(b_i - \langle a_{i\bullet}, y^{i-1}\rangle\right) a_{i\bullet} \tag{5}$$

Note that in addition to dividing the ith equation by $\|a_{i\bullet}\|$, we can also multiply it by $\sqrt{\lambda_i}$ and thus save some additional time during the iterations. This option was not implemented.

We introduce three operators describing the internal Kaczmarz sweep: KSWP (the internal loop of KACZ), BKSWP is similar to KSWP, but with the equations traversed backwards, and DKSWP is KSWP followed by BKSWP.

Definition 2.1. Let A and b be as in (1), but after all the equations have been normalized. Let $\Lambda = (\lambda_1, \dots, \lambda_m)$ be a vector of relaxation parameters, and $x \in \mathbb{R}^n$. Then KSWP(A, b, x, Λ) is defined as y^m, where $y^0, \dots, y^m$ are obtained by executing the following code segment:

```
begin
    set $y^0 = x$.
    for $i = 1,2,\dots,m$ do
        $y^i = y^{i-1} + \lambda_i \left(b_i - \langle a_{i\bullet}, y^{i-1}\rangle\right) a_{i\bullet}$
    enddo
end
```

BKSWP(A,b,x,Λ) is defined similarly, except that the equations are traversed in the reverse order. Finally, DKSWP(A,b,x,Λ) = BKSWP$(A,b,$KSWP$(A,b,x,\Lambda),\Lambda)$. If all the relaxation parameters are fixed, i.e., $\lambda_i = \lambda$ for $1 \le i \le m$, then we will just write λ instead of Λ as the fourth parameter of KSWP, BKSWP and DKSWP.

One of the most basic iterative techniques is SOR (successive overrelaxation); see [Saad 2003, §4.1]. One can apply SOR, with a relaxation parameter λ, to the normal equations system (3); this algorithm is known as SOR-NE. However, it is well known that SOR-NE is equivalent to KACZ with the same fixed relaxation parameter λ. We extend SOR on the normal equations to allow for a vector of cyclic relaxation parameters Λ, and call the result SORC-NE. The result, which is identical to KACZ, is the following:

ALGORITHM 2. **(SORC-NE):**

```
set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
for $k = 0,1,2,\ldots$ until convergence do
    $x^{k+1} = \text{KSWP}(A,b,x^k,\Lambda)$
enddo
```

The *symmetric* SOR (SSOR) is similar to SOR, but with the forward sweep followed by a backward sweep. We also extend SSOR to allow a vector Λ of cyclic relaxation parameters, and call the resulting algorithm, applied to the normal equations (3), the SSORC-NE algorithm:

ALGORITHM 3. **(SSORC-NE):**

```
set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
for $k = 0,1,2,\ldots$ until convergence do
    $x^{k+1} = \text{DKSWP}(A,b,x^k,\Lambda)$
enddo
```

Note that a double KACZ sweep with cyclic relaxation parameters is in fact a regular KACZ sweep, also with cyclic relaxation parameters, applied to the system obtained from (1) by duplicating the original equations in reverse order. The vector of relaxation parameters is also doubled by repeating the first m values in reverse order. Hence, by the above-mentioned result of [Eggermont et al. 1981], SSORC-NE converges even when (1) is inconsistent and/or nonsquare.

The conjugate gradient algorithm (CG), which can be applied when the system matrix A of (1) is square, is the following:

ALGORITHM 4. **CG:**

```
set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
set $p^0 = r^0 = b - Ax^0$.
for $k = 0,1,2,\ldots$ until convergence do
```

```
        $q^k = Ap^k$
        $\alpha_k = \|r^k\|^2/\langle p^k, q^k\rangle$
        $x^{k+1} = x^k + \alpha_k p^k$
        $r^{k+1} = r^k - \alpha_k q^k$
        $\beta_k = \|r^{k+1}\|^2/\|r^k\|^2$
        $p^{k+1} = r^{k+1} + \beta_k p^k$
    enddo
```

It is well known that if A is symmetric and positive definite, then CG converges, in theory, to a solution of (1). In [Björck and Elfving 1979, Lemma 5.1], it is shown that the same holds true if A is only positive semidefinite, provided the system (1) is consistent. In fact, the above lemma provides more detailed information about the convergence properties of such a case, based on the (unique) representation of x^0 as $x^0 = x' + x''$, where $x' \in \text{Range}(A)$ and $x'' \in \text{Null}(A)$ (recall that Range(A) and Null(A) are orthogonal complements and that $\text{Range}(A) + \text{Null}(A) = \mathbb{R}^n$).

CG can also be applied to the normal equations systems (2) and (3), and the resulting algorithms are known respectively as CGNR and CGNE [Saad 2003, §8.3]. Note that the matrices AA^T and A^TA are always symmetric and positive semidefinite, so by the above-mentioned lemma, both CGNR and CGNE converge, in theory, when the system (1) is consistent (even if it is nonsquare).

The following description extends the one given in [Björck and Elfving 1979] for the CGMN algorithm by allowing different relaxation parameters for different equations. This is essential to the parallel extension of CGMN [Gordon and Gordon 2008], but it could also be useful in cases where different equations have very different characteristics due to varying materials or external governing forces. Assume as before that the system (1) is normalized (each equation is divided by the L_2-norm of its coefficients). Denote by A_i the symmetric $n \times n$ matrix obtained from $a_{i\bullet}$ as follows:

$$A_i = a_{i\bullet}a_{i\bullet}^T = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{in} \end{pmatrix} (a_{i1}, \ldots, a_{in}).$$

Let Λ be a vector of relaxation parameters. Denoting $Q_i = I - \lambda_i A_i$, we obtain the following representation for the KACZ inner iteration of the normalized system (Equation (5)):

$$y^i = Q_i y^{i-1} + \lambda_i b_i a_{i\bullet}. \quad (6)$$

Hence, in a complete double sweep, we obtain the following representation for the SSORC-NE (Algorithm 3) iteration:

$$x^{k+1} = Qx^k + Rb, \quad (7)$$

where $Q = Q_1 \cdots Q_m Q_m \cdots Q_1$ and R is the resulting matrix multiplying b. Note that the matrices Q_i are symmetric, so Q is also symmetric. However, in SSORC-NE, the inner loop is also obtained by the operator DKSWP, so for any vector $x \in \mathbb{R}^n$, we have the identity

$$Qx + Rb = \text{DKSWP}(A, b, x, \Lambda). \quad (8)$$

As noted earlier, SSORC-NE converges, and let x^* be a convergence point. Then x^* is a fixed point of the iteration (7), i.e., $x^* = Qx^* + Rb \;\Rightarrow\; (I-Q)x^* = Rb$. Therefore, x^* is a solution of the system

$$(I-Q)x = Rb\,. \tag{9}$$

It follows that the system (9) is consistent. Also, since Q is symmetric, so is $(I-Q)$, which is the system matrix of (9). Furthermore, in [Gordon and Gordon 2008, Theorem 1] it is shown that $(I-Q)$ is positive semidefinite. Hence, by [Björck and Elfving 1979, Lemma 5.1], CG can be applied to (3), and its convergence (in theory) is guaranteed. The CGMNC algorithm is the application of CG to the system (9). Note that neither of the two matrices in (9) have to be computed explicitly: using the identity of Equation (8), matrix-vector products are computed by suitable double sweeps of KACZ, as explained below.

ALGORITHM 5. **CGMNC [Gordon and Gordon 2008]:**

```
set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
set $p^0 = r^0 = Rb - (I-Q)x^0 = Qx^0 + Rb - x^0 = \mathrm{DKSWP}(A,b,x^0,\Lambda) - x^0$.
note: the rightmost equality follows from eq. (8).
for $k = 0,1,2,\ldots$ until convergence do
    $q^k = (I-Q)p^k = p^k - \mathrm{DKSWP}(A,0,p^k,\Lambda)$
    note: the above equality follows from eq. (8) with $b = 0$.
    $\alpha_k = \|r^k\|^2 / \langle p^k, q^k \rangle$
    $x^{k+1} = x^k + \alpha_k p^k$
    $r^{k+1} = r^k - \alpha_k q^k$
    $\beta_k = \|r^{k+1}\|^2 / \|r^k\|^2$
    $p^{k+1} = r^{k+1} + \beta_k p^k$
enddo
```

The required number of operations of Algorithm 5, for each iteration, can be evaluated as follows:

—From Definition 2.1, the execution of KSWP requires m scalar products of the type $\langle a_{i\bullet}, y^{i-1}\rangle$, where $a_{i\bullet}$ is a sparse vector and y^{i-1} is dense. This is equivalent to one matrix-vector product of type Ax, where A is sparse and x is dense. Also required are m operations of the type $\alpha x + y$, where α is a constant and x and y are dense vectors. However, these m operations are clearly equivalent to multiplying a very sparse matrix A by a dense vector x, where A has 1's on the main diagonal, some number on the diagonal above it, and zero elsewhere. Hence we consider it to be at most one-half of an Ax operation.

—Looking at the main loop of Algorithm 5, we see that each iteration requires one operation of the type $\mathrm{DKSWP}(A,0,p^k,\Lambda)$, which is a double KSWP operation with $b = 0$, so it requires 3 Ax operations.

—Counting the other operations in Algorithm 5, we get the following total number of operations of each type:

—3 matrix-vector operations Ax, where A is sparse and x is dense,
—4 operations of the type $\alpha x + y$,
—2 scalar products $\langle x, y \rangle$, where both x and y are dense.

For comparison, note that (the un-preconditioned) GMRES requires one Ax operation, and CGS and Bi-CGSTAB require two.

As to the memory requirements of Algorithm 5, we can easily see that a careful coding of the main loop requires memory for 4 (dense) vectors, in addition to A and b. The only significant difference in respect to memory requirements between CGMN and the other methods is that restarted GMRES requires km vectors, where k is the dimension of the Krylov subspace used.

3. SETUP OF THE NUMERICAL EXPERIMENTS

Tests were run on a Pentium IV 2.8GHz processor with 3GB memory. The code was compiled with the GNU compilers. This section describes the test problems, the nonsymmetric solvers and preconditioners which were used, implementation details, and the stopping tests.

3.1 The test problems

We examined nine well known problems derived from convection-diffusion elliptic PDEs. Problems 1–7 were collected from a variety of sources [Kincaid and Young 1981; Kuck et al. 1986; Kamath and Sameh 1989; Bramley et al. 1990; Elman and Golub 1990], and were also used as test problems in several other works [Bramley and Sameh 1991; 1992; Arioli et al. 1992; Arioli et al. 1995; Gordon and Gordon 2005]. Problems 3 and 7 also contain a reaction term. Problem 8 is from [Saad 2003, §3.7, Problem F3D] (also available from the SPARSKIT library [Saad 1990]). The problems are typical of many fields such as CFD, heat transfer and structural mechanics, and are commonly used in a wide variety of scientific, engineering and industrial applications. Problem 9 is derived from problem 8 by increasing the convection coefficient; this was done in order to examine the effect of such a change on the performance of the different solution methods. In the following description of the test problems, we use the standard notation $\Delta u = u_{xx} + u_{yy} + u_{zz}$.

(1) $\Delta u \; + \; 1000 u_x \; = \; F.$
(2) $\Delta u \; + \; 1000 \exp(xyz)(u_x + u_y - u_z) \; = \; F.$
(3) $\Delta u \; + \; 100 x u_x - y u_y + z u_z + 100(x+y+z)u/xyz \; = \; F.$
(4) $\Delta u \; - \; 10^5 x^2 (u_x + u_y + u_z) \; = \; F.$
(5) $\Delta u \; - \; 1000(1+x^2)u_x + 100(u_y + u_z) \; = \; F.$
(6) $\Delta u \; - \; 1000\,[(1-2x)u_x + (1-2y)u_y + (1-2z)u_z] \; = \; F.$
(7) $\Delta u \; - \; 1000 x^2 u_x + 1000 u \; = \; F.$
(8) $\Delta u \; - \; \partial(10 e^{xy} u)/\partial x - \partial(10 e^{-xy} u)/\partial y \; = \; F.$
(9) $\Delta u \; - \; \partial(1000 e^{xy} u)/\partial x - \partial(1000 e^{-xy} u)/\partial y \; = \; F.$

Problems 1–7 have the following analytical (preassigned) solutions:

Problem 1: $u(x,y,z) \; = \; xyz(1-x)(1-y)(1-z).$
Problem 2: $u(x,y,z) \; = \; x + y + z.$
Problems 3-7: $u(x,y,z) \; = \; \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z).$

The expression for the right-hand side function F in problems 1–7 is computed analytically, using the preassigned solution. For problems 8 and 9, the right-hand-side function F is irrelevant, because the equation system was set up by first computing the system matrix A, and then computing $b = Av$, where v was chosen as $v = (1,1,\ldots,1)^T$ (similarly to the approach taken in [Saad 2003, §3.7, Problem F3D]). All the test problems were solved on the unit cube domain $[0,1]\times[0,1]\times[0,1]$. In problems 1–7, Dirichlet boundary conditions were used, as determined by the preassigned solutions. For problems 8 and 9, the boundary conditions were taken as $u = 0$. The problems were discretized using a uniform grid with the same number of grid points in each direction, and the equations were obtained by using a seven-point centered difference scheme. Test runs were made for problems of size $80\times80\times80 = 512,000$ equations. Additional tests were also made on smaller grid sizes in order to study how the various algorithms perform as the grid is steadily refined.

3.2 Algorithms and preconditioners

CGMN is compared with several of the leading state-of-the-art methods, used with and without preconditioners. We used four Krylov-based iterative nonsymmetric solvers: CGNR [Saad 2003, §8.3], restarted GMRES [Saad and Schultz 1986], CGS [Sonneveld 1989] and Bi-CGSTAB [van der Vorst 1992]. As mentioned, all the equations were normalized before applying the various solvers, by dividing each equation by the L_2-norm of its coefficients. The following preconditioners were used with GMRES, CGS and Bi-CGSTAB: ILUT, Neumann and Least-Squares. We also tried the Jacobi and the symmetric Gauss-Seidel preconditioners but they failed in all the cases. For details of these preconditioners, see for example [Saad 2003, Ch. 10 & 12]. In all, CGMN was compared to 13 different combinations of algorithms and preconditioners (not counting the failed preconditioners). The comparisons used the AZTEC software library [Tuminaro et al. 1999], which is designed for the solution of large sparse systems of linear equations.

In all the tested algorithms, the initial estimate was taken as $x^0 = 0$. Restarted GMRES (denoted as RGMRES) was run with Krylov subspace size $k = 10$; larger values of k did not improve the robustness of RGMRES on our test problems, until k became a significant fraction of the problem size (with a large increase of the required storage). The Neumann and Least-Squares polynomial preconditioners were run with AZTEC's default polynomial order parameter of 3.

The ILUT preconditioner that was implemented is the one supplied by AZTEC; it does not use pivoting. This preconditioner [Saad 2003] depends on two parameters: the *drop tolerance* (the value below which elements are taken as zero), and the *fill-in*, which controls the maximum number of nonzeros allowed in each column/row of the incomplete LU factors. We used the default AZTEC values for ILUT's parameters: drop tolerance = 0 and fill-in = 1.0 (i.e., no additional elements). Note that experimenting with ILUT's parameters requires a search in a two-dimensional parameter space in order to find the optimal values for each particular problem and for each preconditioned algorithm. Since the default values produced good results when ILUT worked, we did not experiment with them.

3.3 Implementation details

Numerical PDE approximations on 3-dimensional grids (structured and unstructured) exhibit spatial locality, since each equation centered about a grid point involves only its neighboring grid points. For the structured 3-dimensional grids that we used, the resulting coefficient matrix is a 7-point stencil matrix. Our implementation of the data structures used by CGMN is similar to that of AZTEC. The system matrix was stored in the sparse matrix format called DMSR (distributed modified sparse row) [Tuminaro et al. 1999], which is a generalization of the MSR format [Saad 2003, §3.4].

3.4 Stopping tests

There are several stopping criteria which one may apply to iterative systems. Our stopping criterion was to use the relative residual: $\|b - Ax\|/\|b - Ax^0\| < 10^{-7}$. In some of the cases, this was not attainable. Since this stopping criterion depends on the scaling of the equations, we first normalized the equations (for all the tested methods) by dividing each equation by the L_2-norm of its coefficients. In order to limit the time taken by the methods implemented in AZTEC, the maximum number of iterations was set to 5000. The AZTEC library has several other built-in stopping criteria: numerical breakdown, numerical loss of precision and numerical ill-conditioning.

4. RESULTS AND DISCUSSION

4.1 Convergence results

In this subsection, we present the runtime results for problems 1–9 with 512,000 equations. For each problem, the plot for CGMN was obtained with the optimal λ. In §4.2 the dependence of the runtimes on λ will be studied. Note that we did not attempt to improve on the AZTEC code, nor did we optimize all the parameters of all the preconditioner coefficients. Most likely, such experimentation would have improved the runtime performance of CGMN's competitors to a certain extent. Hence, the figures should be viewed as an approximation to the relative performance of CGMN and the other solution methods.

Figures 1–9 show which algorithms converged on test problems 1–9. Only the plots of the converged algorithms are shown, with two exceptions: Bi-CGSTAB (without a preconditioner) on problem 1 and CGS+ILUT on problem 5; their convergence was either too slow or too oscillatory to be relevant. The plots show the relative residual versus the execution time for the nine test problems. Since $x^0 = 0$, the relative residual is $\|b-Ax\|/\|b\|$, and it is denoted by res/res$^{(0)}$. The plots for the preconditioned algorithms include the setup time for the preconditioner; this time interval was added to the time of the first iteration.

CGMN and CGNR are the only algorithms that converged on problem 3, so the convergence plot of KACZ is also shown for comparison. KACZ also converged, but much more slowly. Problems 3 and 7 are hard for solvers such as GMRES because they are indefinite, with eigenvalues surrounding the origin. It is well known that the eigenvalue distribution and the conditioning of the matrix of eigenvectors is more important for the convergence of GMRES than the conditioning of the system matrix. See also [Arioli et al. 1992; Bramley and Sameh 1991; 1992].

The following conclusions can be drawn from the convergence data:

(1) Robustness: CGMN and CGNR converged on all the problems—a property not shared by any other algorithm/preconditioner combination. Also, on some of the problems, most of the other methods failed.

(2) Efficiency: On problems 1–6 and 9, the convergence rate of CGMN shows up as better than that of the other methods. This difference is especially pronounced on problems 2,3,4,6 and 9.

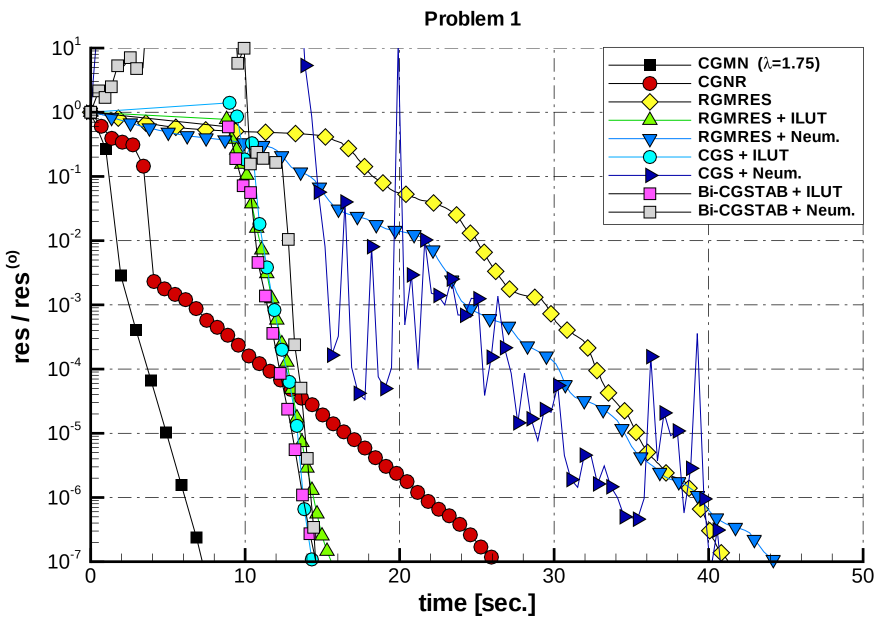

Fig. 1. Convergence results for problem 1.

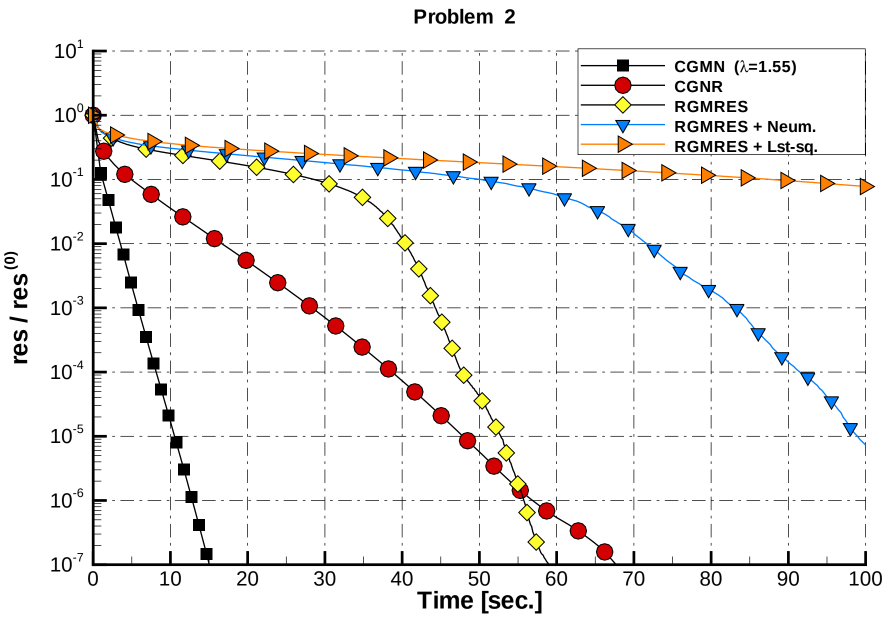

Fig. 2. Convergence results for problem 2.

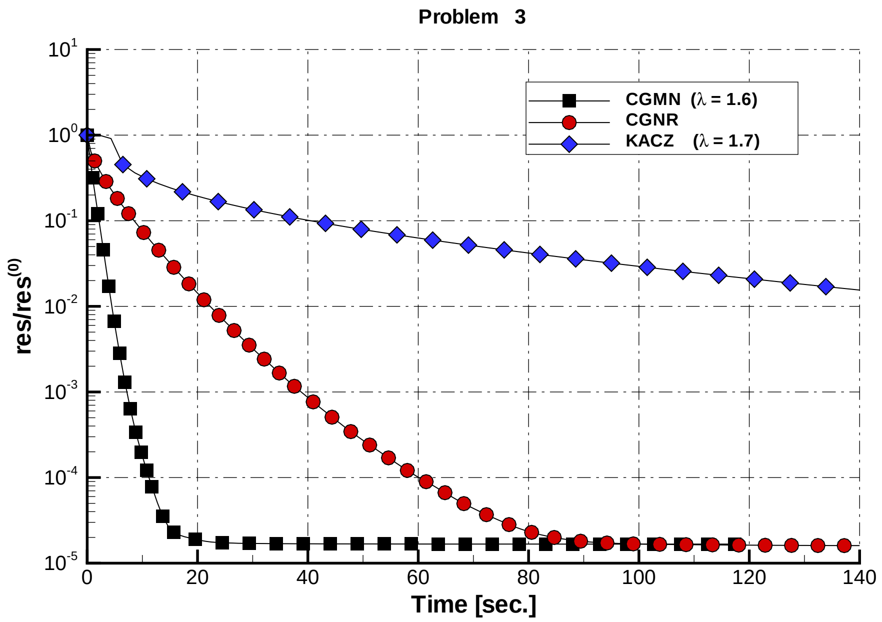

Fig. 3. Convergence results for problem 3.

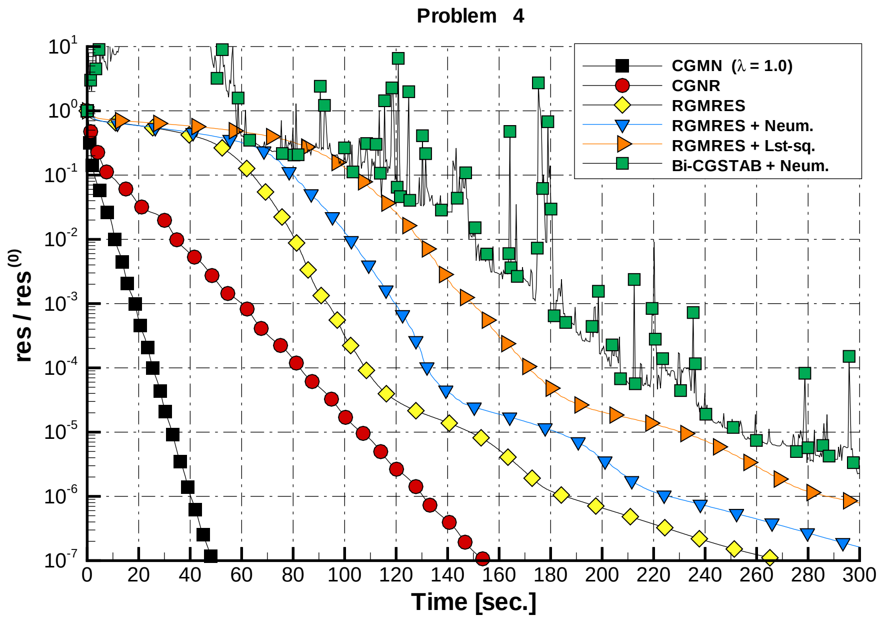

Fig. 4. Convergence results for problem 4.

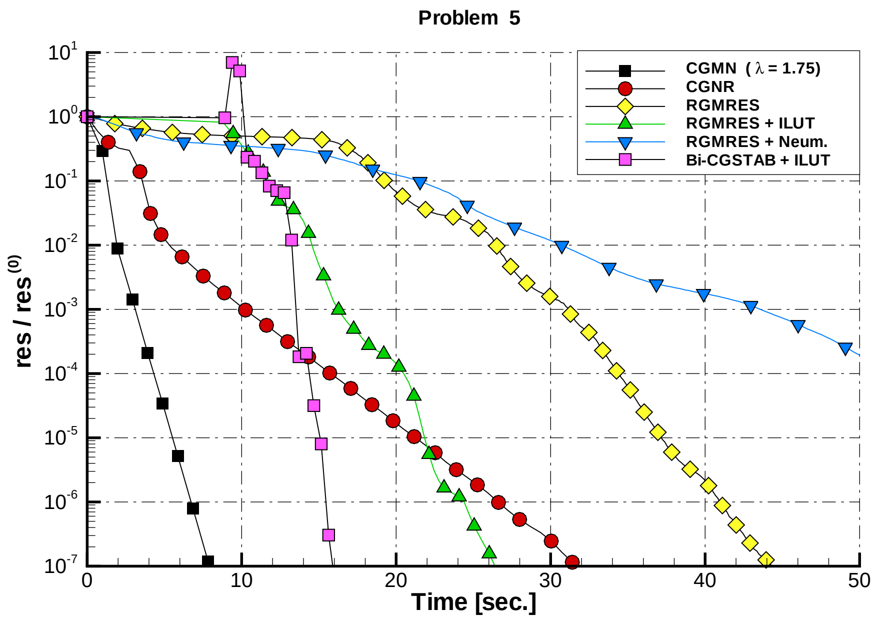

Fig. 5. Convergence results for problem 5.

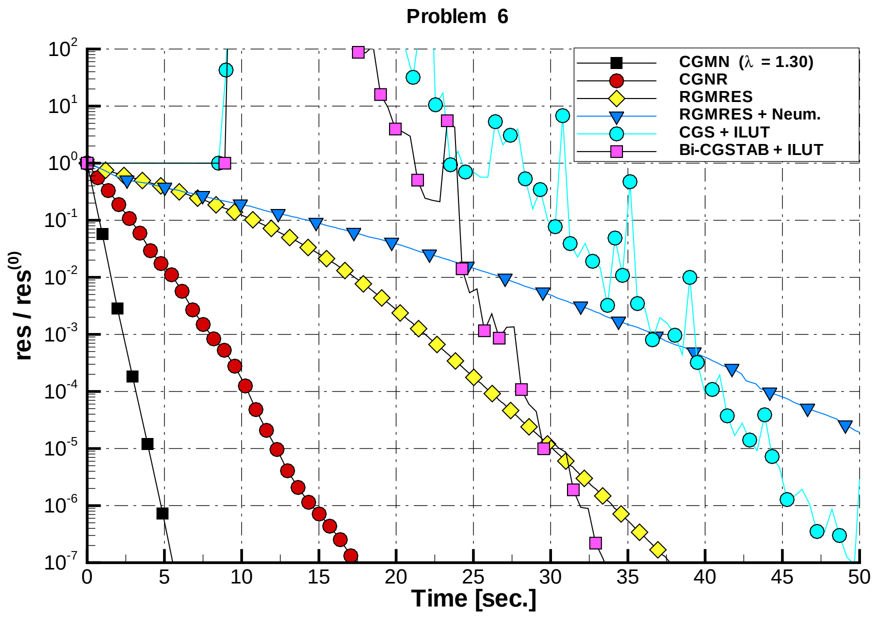

Fig. 6. Convergence results for problem 6.

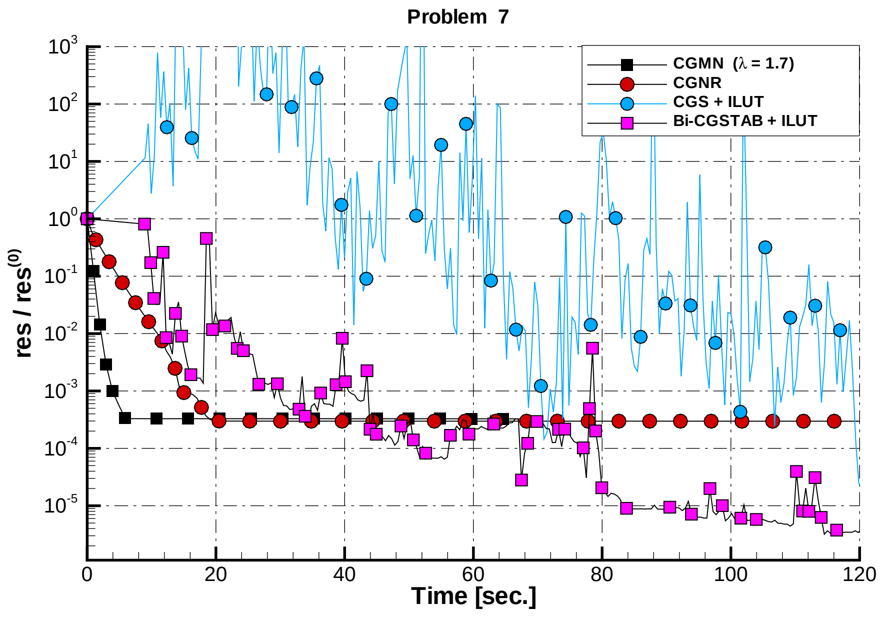

Fig. 7. Convergence results for problem 7.

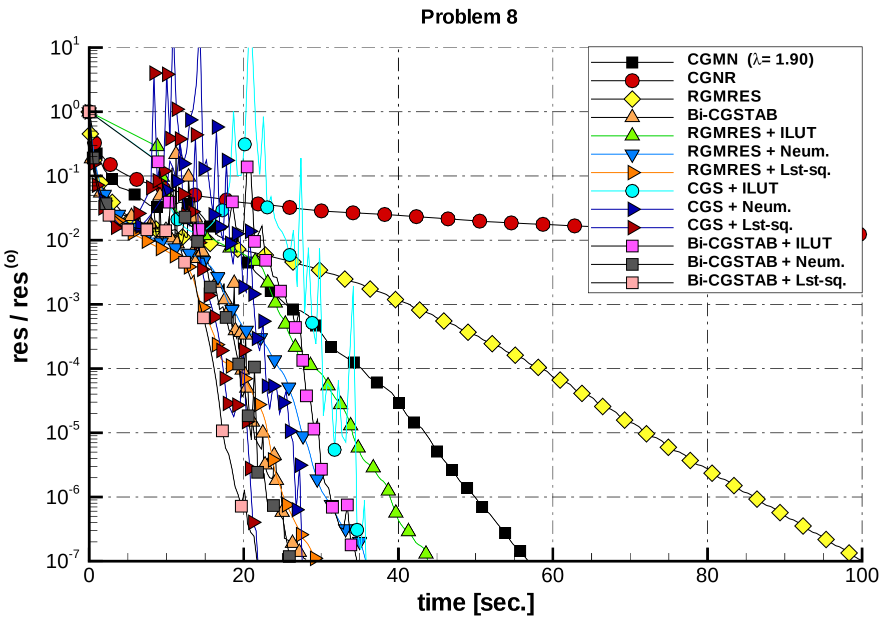

Fig. 8. Convergence results for problem 8.

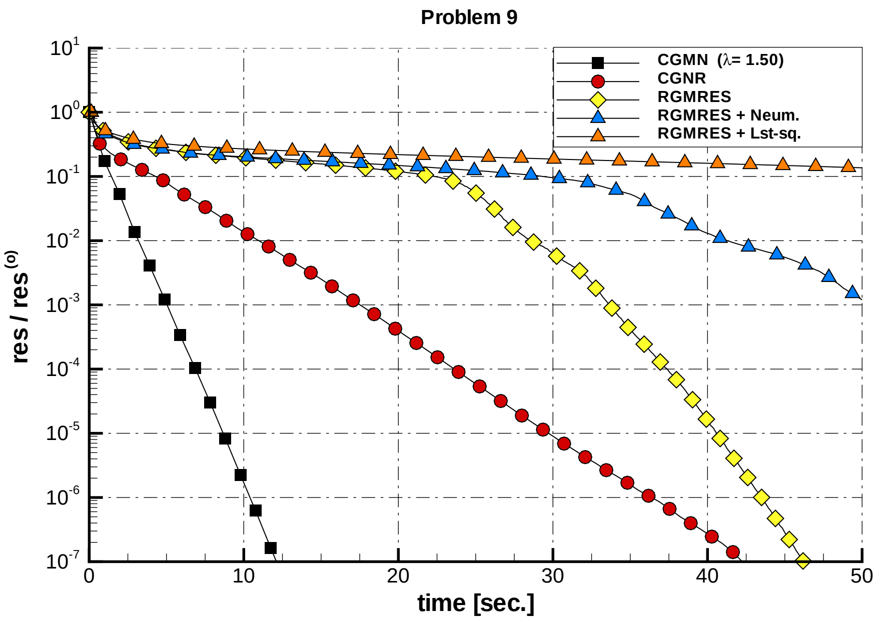

Fig. 9. Convergence results for problem 9.

(3) Initial rate of convergence: On problems 1–7 and 9, CGMN's initial rate of convergence was very good as compared to the other methods.

(4) Monotonicity: On problems 1–7 and 9, CGMN's convergence was monotonic, whereas CGS and Bi-CGSTAB were oscillatory to some extent. Some very slight deviations from monotonicity appeared in CGMN's behavior on problem 8.

(5) Problem 7: CGMN and CGNR seem to stagnate, but they actually continued to improve at a very gradual rate. Also, the comparison between CGMN and Bi-CGSTAB with ILUT depends on the prescribed relative residual: for 3.5×10^{-4}, CGMN is the fastest method; otherwise, it is Bi-CGSTAB with ILUT.

(6) Problem 8: CGMN performed worse than most of the other algorithm/preconditioner combinations, whereas CGNR performed very poorly. Hence, CGMN should not be considered as a first choice for problems with a small convection term. This fact suggests that it may be worthwhile to search for preconditioners for CGMN.

(7) Problem 9: As mentioned, this problem is derived from problem 8 by increasing the convection term from 10 to 1000. There are two very significant differences between the plots of problems 8 and 9: most of the successful methods for problem 8 failed on problem 9, and the covergence rate of CGMN (and CGNR) show a great improvement relatively to the other methods. It is also interesting to note that CGMN, CGNR and restarted GMRES actually performed better on problem 9 than on problem 8 (in absolute terms).

4.2 Comparison of the numerical solution with the analytic solution

In this section we present a comparison between the numerical solution of the linear system, as obtained with CGMN, and the analytic solution of the associated PDE. Denote the analytic solution by u, and by x^k the kth iterate. The relative error between the numerical and the analytic solution is defined as err-an $= \|u - x^k\|/\|u\|$. The maximal error, denoted by err-max, is simply the maximal difference between u and x^k at any grid point. Table I presents the relative residual (rel-res) obtained by CGMN, err-an, err-max, and the number of iterations used by CGMN, for the nine test problems.

Problem	res/res$^{(0)}$	err-an	err-max	iter
1	1.40E-14	7.40E-16	3.52E-17	180
2	7.14E-15	2.57E-15	6.15E-14	330
3	1.70E-05	2.13E-04	1.73E-03	300
4	3.50E-14	3.99E-04	2.05E-03	1500
5	1.34E-14	2.97E-04	3.33E-04	180
6	7.26E-15	2.40E-04	3.06E-04	120
7	8.10E-05	6.84E-02	1.69E-01	1635
8	3.65E-14	2.45E-15	1.44E-14	1050
9	6.75E-15	1.22E-15	6.66E-15	270

Table I. Difference between the numerical solution and the analytic solution of the PDEs.

We can see that on problems 1,2,8 and 9 CGMN solves the original PDE with excellent precision, on problems 3–6 the solution is reasonable for many applications, such as CFD, and on problem 7, the result is poor. As noted in the introduction, the problem is inherent in the CDS discretization at the given mesh size. Note in particular the large discrepancy between the relative residual and the relative (analytic) error in problems 4, 5 and 6.

4.3 Error and residual results of the algebraic systems

In order to evaluate the purely algebraic behavior of CGMN, problems 1–9 were recalculated as follows: The right-hand-side b was determined as $b = Ax^*$, with x^* being the analytic solution at the grid points for problems 1–7 and $(1,\ldots,1)^T$ for problems 8 and 9. Note that this way we have the precise algebraic solution of the linear systems, and the error results are not dependent on how well the algebraic systems model the PDEs. The relative error at the kth iteration is defined as rel-err $= \|x^k - x^*\|/\|x^*\|$. The relative residual is calculated as before, and it is denoted in this subsection by "rel-res". Figures 10 and 11, which show the convergence plots of rel-err and rel-res for problems 1–9, lead to the following conclusions:

(1) The convergence of the error is monotonic. This is in agreement with [Björck and Elfving 1979, Theorem 5.1].

(2) The convergence of rel-res is almost monotonic, with some very slight deviations in problem 8.

(3) With the exception of problems 3 and 7, the plots of rel-err and rel-res are very similar.

(4) In problems 3 and 7, rel-res decreases much more than rel-err.

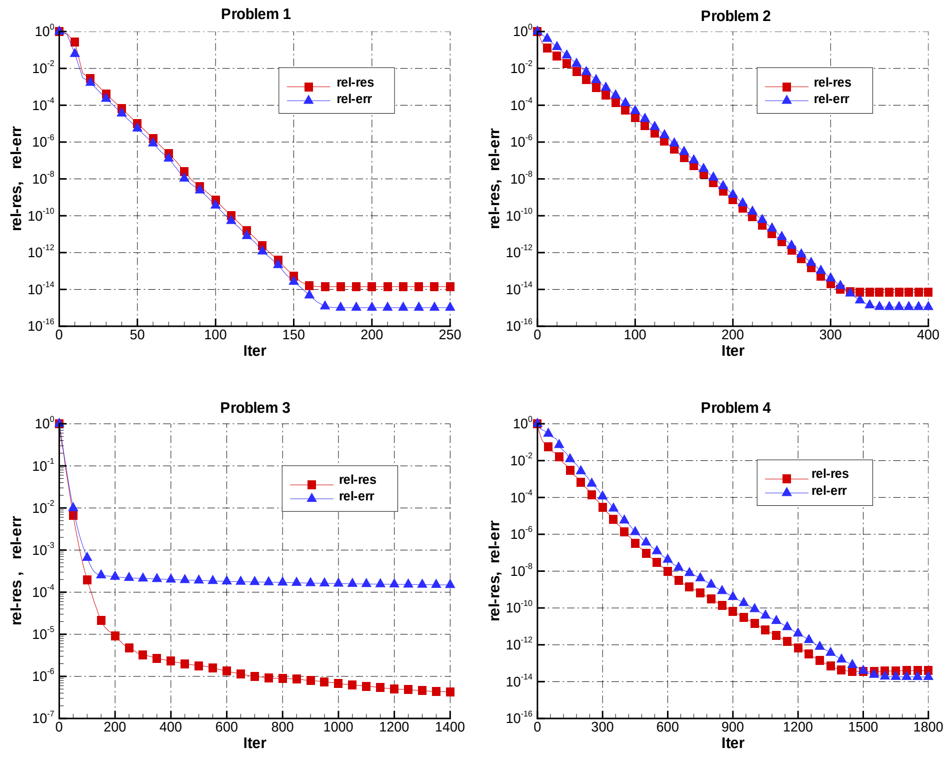

Fig. 10. Relative residual and error of CGMN for problems 1–4.

4.4 Convergence behavior as a function of grid size

In this subsection we study the behavior of CGMN when the grid size is varied. We compare CGMN with CGNR and with restarted GMRES and Bi-CGSTAB; the latter two were tested by themselves and with the ILUT preconditioner. The choice of GMRES, Bi-CGSTAB and ILUT was based on the fact that these combinations can be considered as being generally the most prominent in the convergence plots shown in Figures 1–9 (after CGMN). The algorithms were tested until the following relative residuals of were achieved: 2×10^{-4} for problem 3, 5×10^{-4} for problem 7, and 10^{-4} for the other problems. Figures 12 and 13 show bar plots based on the runtimes of these algorithms on problems 1–9, for grid sizes of $10\times10\times10$, $20\times20\times20$, $40\times40\times40$ and $80\times80\times80$. For each grid size, the runtimes were normalized with respect to CGMN. No bar plots are shown when the algorithm did not converge.

Figures 12 and 13 show that CGMN is consistently robust for varying grid sizes. This fact suggests that CGMN could be a good candidate for multilevel applications; this is a topic for further research. On problems 1–7 and 9, the performance of CGMN on the coarse grids was relatively better than the other methods (when they converged); this fact is significant since convection-dominated PDEs are known to be problematic on coarse grids; see [Ferziger and Perić 2002]. Note that relatively to CGMN, CGNR performs quite well on the $10\times10\times10$ grid, but its performance deteriorates as the grid is refined.

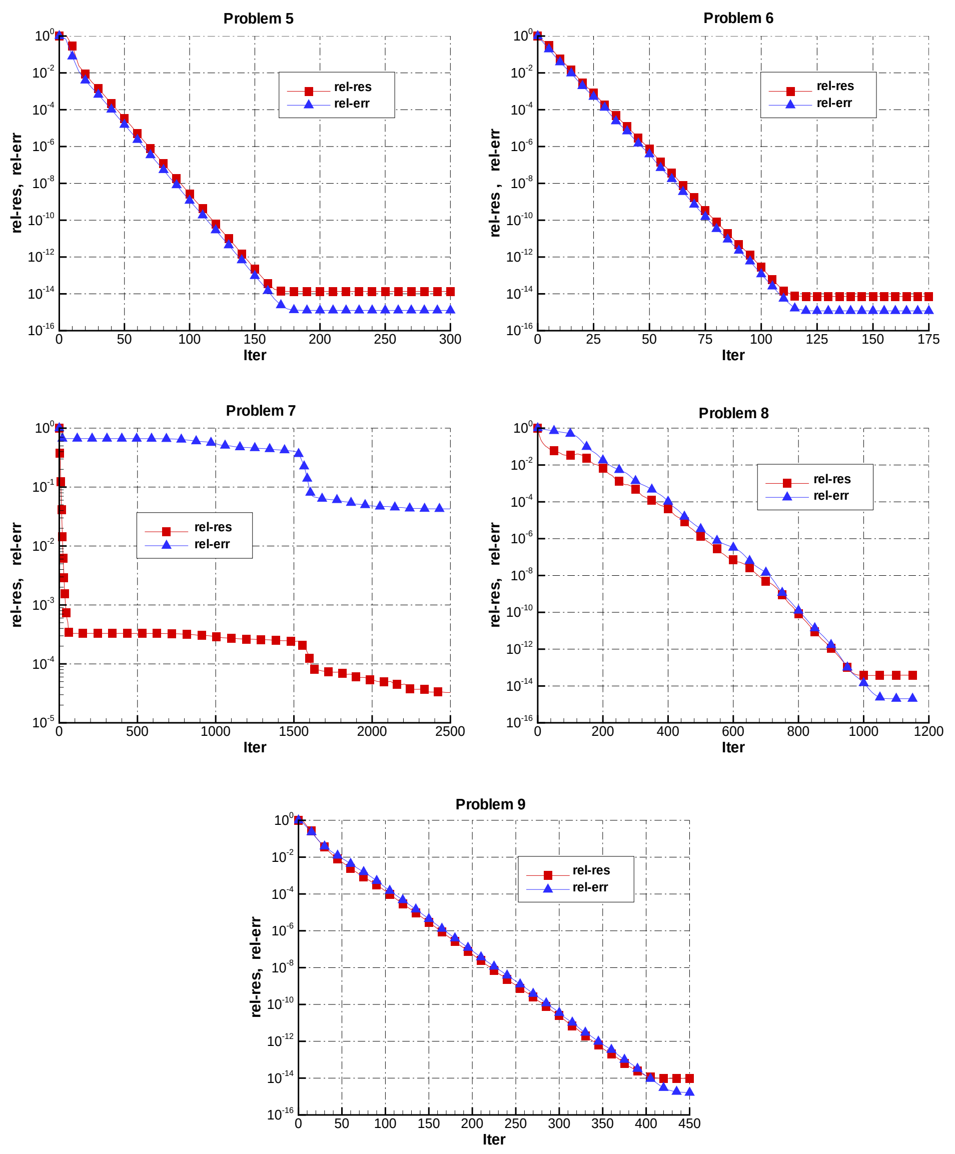

Fig. 11. Relative residual and error of CGMN for problems 5–9.

Tables II and III provide additional information. Shown in the tables are both the runtimes and the number of iterations for each of the nine cases, on all the tested grid sizes. Also shown are the optimal values of λ for CGMN for each of the grid sizes. We can see that in all cases, the optimal λ shows a gradual increase as the grid is refined.

Some comments on how the number of iterations changes as the grid is refined. Denote the grid sizes by $\ell \times \ell \times \ell$. We can see that except for case 8, the number of iterations required by CGMN increases approximately linearly or even sublinearly with ℓ. However,

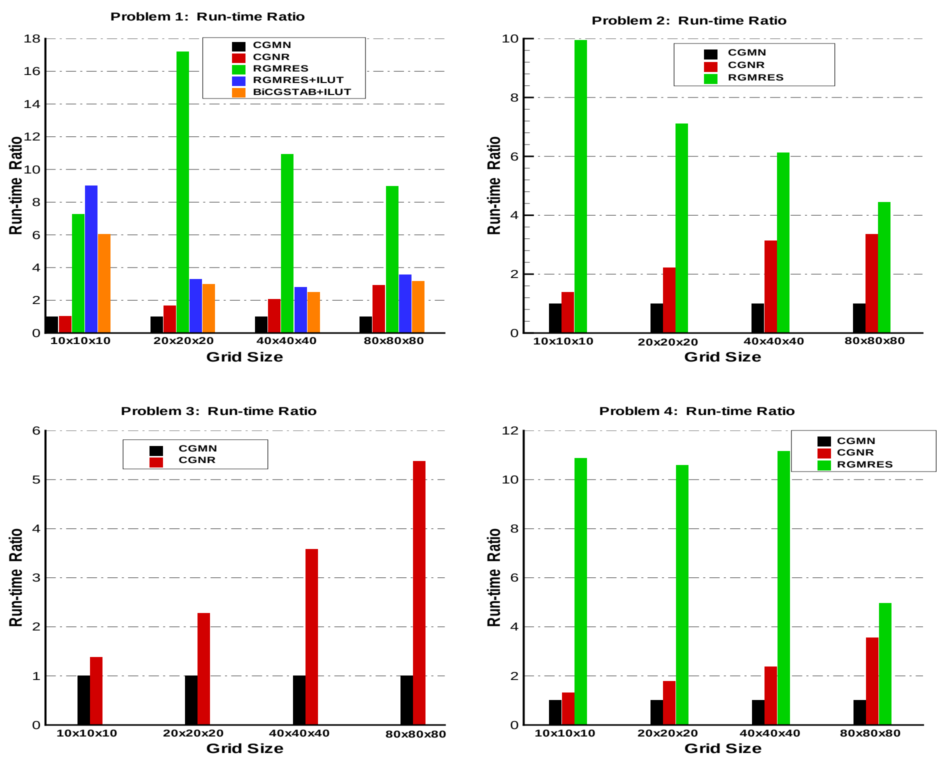

Fig. 12. Runtime ratios relative to CGMN, for varying grid sizes.

on problems 3 and 7, there is a nonlinear jump when going from 40×40×40 to 80×80×80, as the number of iterations approximately triples.

4.5 Convergence of CGMN as a function of λ

Table IV shows the values of the optimal λ for problems 1–9. Figure 14 show how the number of iterations of CGMN depends on the choice of λ. The number of iterations shown in these figures was the number required to achieve a relative residual of 10^{-7}, except for problem 3 with 10^{-4} and problem 7 with 5×10^{-4}, on the uniform $80\times80\times80$ grid.

From these figures, we can see that the dependence is always a concave function, so finding the optimal λ is simple, and it can be easily automated. One should also note that in all cases, except for case 8, the variation in the number of iterations shows only very moderate changes in the neighborhood of the optimal λ. This means that it is not necessary to find the optimal λ with high precision. In case 8, the slopes are relatively sharper. The optimal values of λ for cases 4 and 8 are in a certain sense "outliers". For the other problems, $\lambda = 1.6$ would be a good initial estimate for the current grid size. Again, the thorny case for CGMN is problem 8.

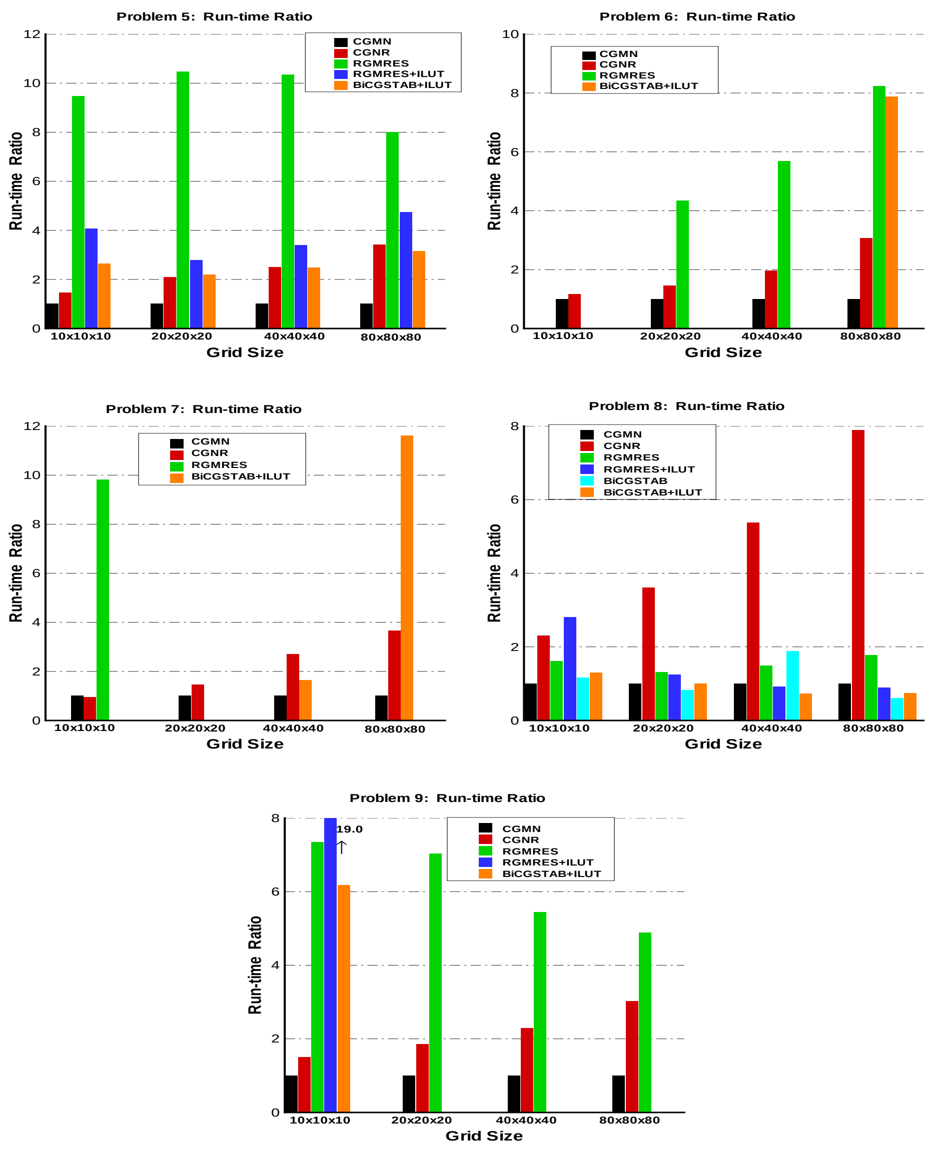

Fig. 13. Runtime ratios relative to CGMN, for varying grid sizes.

5. CONCLUSIONS AND FURTHER RESEARCH

This paper reexamined the CGMN algorithm, which was introduced in [Björck and Elfving 1979] as a CG-acceleration of the SSOR algorithm applied to the normal equations system. CGMN is also equivalent to a CG-acceleration of a double (back and forth) Kaczmarz sweep of the original equations. The mathematical results guarantee that it always converges (at least in theory), even if the equation system is inconsistent and/or nonsquare. Furthermore, the error reduction is monotonic according to [Björck and Elfving 1979,

Problem 1	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	6	6.73E-04	12	1.36E-02	22	2.48E-01	38	3.65
	$\lambda = 1.30$	---	$\lambda = 1.50$	---	$\lambda = 1.50$	---	$\lambda = 1.70$	---
CGNR	12	6.80E-04	36	2.27E-02	76	5.09E-01	168	10.71
RGMRES	32	4.88E-03	180	2.34E-01	162	2.71	220	32.77
RGMRES+ILUT	7	6.06E-03	7	4.48E-02	8	6.90E-01	14	13.02
BiCGSTAB+ILUT	4	4.06E-03	4	4.06E-02	4	0.62	8	11.58

Problem 2	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	42	4.71E-03	42	4.77E-02	58	6.54E-01	112	10.77
	$\lambda = 0.90$	---	$\lambda = 1.10$	---	$\lambda = 1.40$	---	$\lambda = 1.60$	---
CGNR	115	6.52E-03	167	1.06E-01	306	2.05	568	36.21
RGMRES	354	4.68E-02	261	3.39E-01	242	4.01	321	47.82

Problem 3	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	6	6.73E-04	12	1.36E-02	31	3.49E-01	96	9.23
	$\lambda = 1.00$	---	$\lambda = 1.20$	---	$\lambda = 1.50$	---	$\lambda = 1.70$	---
CGNR	16	9.07E-04	49	3.10E-02	187	1.25	778	49.6

Problem 4	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	92	1.03E-02	106	1.19E-01	136	1.53	226	21.73
	$\lambda = 0.90$	---	$\lambda = 0.90$	---	$\lambda = 1.00$	---	$\lambda = 1.30$	---
CGNR	239	1.35E-02	337	2.13E-01	541	3.62	1211	77.2
RGMRES	858	1.12E-01	965	1.26	1029	17.06	725	108

Problem 5	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	23	2.58E-03	27	3.06E-02	29	3.27E-01	45	4.32
	$\lambda = 1.20$	---	$\lambda = 1.40$	---	$\lambda = 1.50$	---	$\lambda = 1.70$	---
CGNR	66	3.74E-03	101	6.38E-02	122	8.16E-01	231	14.73
RGMRES	185	2.44E-02	247	3.20E-01	206	3.38	232	34.56
RGMRES+ILUT	32	1.05E-02	27	8.49E-02	26	1.11	37	20.5
BiCGSTAB+ILUT	17	6.80E-03	14	6.68E-02	11	0.81	13	13.56

Table II. Runtimes and number of iterations for problems 1–5.

Thm. 5.1].

CGMN was compared with CGNR, restarted GMRES, CGS and Bi-CGSTAB on nine test cases of linear systems derived from elliptic convection-diffusion PDEs by central-difference discretization on a uniform grid. Eight of the test cases consisted of stiff linear systems derived from PDEs with a very large convection term (i.e., high Péclet number). In all, CGMN was compared with 13 different combinations of algorithms and preconditioners. CGMN and CGNR were the only algorithm which converged on all the test cases,

Problem 6	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	31	3.48E-03	18	2.04E-02	22	2.48E-01	33	3.17
	$\lambda = 0.90$	---	$\lambda = 0.90$	---	$\lambda = 1.00$	---	$\lambda = 1.20$	---
CGNR	72	4.08E-03	47	2.97E-02	73	4.88E-01	153	9.75
RGMRES	---	---	68	8.87E-02	86	1.41	175	26.07
BiCGSTAB+ILUT	---	---	---	---	---	---	42	24.96

Problem 7	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	8	8.98E-04	8	9.08E-03	14	1.58E-01	39	3.75
	$\lambda = 1.00$	---	$\lambda = 1.10$	---	$\lambda = 1.40$	---	$\lambda = 1.80$	----
CGNR	15	8.50E-04	21	1.33E-02	64	4.28E-01	215	13.71
RGMRES	67	8.81E-03	---	---	---	---	---	---
BiCGSTAB+ILUT	---	---	---	---	77	2.58	89	43.5

Problem 8	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	21	2.36E-03	52	5.90E-02	132	1.49	344	33.08
	$\lambda = 1.70$	---	$\lambda = 1.80$	---	$\lambda = 1.90$	---	$\lambda = 1.93$	---
CGNR	96	5.44E-03	337	2.13E-01	1196	8.00	4093	260.93
RGMRES	29	3.80E-03	59	7.70E-02	135	2.21	394	58.69
RGMRES+ILUT	9	6.62E-03	19	7.34E-02	36	1.37	64	29.27
BiCGSTAB	20	2.74E-03	35	4.86E-02	69	0.93	136	19.86
BiCGSTAB+ILUT	6	4.23E-03	11	5.93E-02	21	1.08	41	24.61

Problem 9	10*10*10=1,000		20*20*20=8,000		40*40*40=64,000		80*80*80=512,000	
	no. it.	time	no. it.	time	no. it.	time	no. it.	time
CGMN	33	3.80E-03	34	3.81E-02	49	5.46E-01	71	7.32
	$\lambda = 1.10$	----	$\lambda = 1.10$	---	$\lambda = 1.30$	----	$\lambda = 1.50$	---
CGNR	101	5.72E-03	111	7.05E-02	187	1.25	347	22.13
RGMRES	222	2.79E-02	204	2.68E-01	179	2.97	252	35.75
RGMRES+ILUT	386	7.23E-02	---	---	---	----	---	---
BiCGSTAB+ILUT	117	2.35E-02	---	---	---	---	---	---

Table III. Runtimes and number of iterations for problems 6–9.

Problem	1	2	3	4	5	6	7	8	9
λ	1.75	1.55	1.60	1.00	1.75	1.30	1.70	1.90	1.50

Table IV. Optimal values of λ for problems 1–9.

demonstrating that both methods are extremely robust. In terms of runtime performance, CGMN was much better than CGNR. On seven of the test cases, CGMN appeared to perform better than any other combinations of algorithms and preconditioners. In one test case, the relative performance of CGMN and Bi-CGSTAB (with ILUT) depended on the desired convergence goal. CGMN performed worse than most methods on the one test

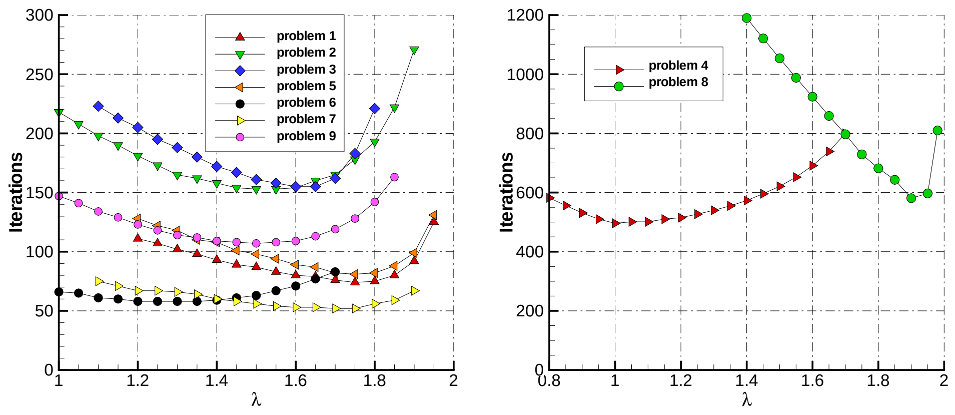

Fig. 14. No. of iterations to achieve the prescribed convergence on problems 1–9.

case which was *not* strongly convection-dominated.

In the eight convection-dominated test cases, CGMN'S initial rate of convergence was very good, and the relative residual decreased monotonically. This property indicates that it is potentially useful for applications in which solving a linear system is a frequent intermediate step, such as quasi-linearization methods for non-linear systems.

It should be noted that CGMN requires a relaxation parameter λ to achieve optimal results. However, the dependence of the convergence rate on the choice of λ is a concave function, so the optimal λ can be determined quite efficiently and even automatically. On the eight convection-dominated problems, the rate of convergence is not very sensitive to small deviations from the optimal λ, so there is no need to find the optimal λ with high precision.

In comparing CGMN to the other algorithms, the following differences can be noted: GMRES requires more memory due to the need to store the Krylov subspace basis, and CGS and Bi-CGSTAB are somewhat oscillatory. With regard to CGNR, note that although it is not generally considered useful (see [Saad 2003, §8.3.1]), it performed quite well on the eight convection-dominated problems. It is therefore a reasonable second choice, particularly on coarse grids or when the required convergence goal is not too small. One advantage of CGNR is that it is parameter-free.

To summarize, we can conclude from our experiments that CGMN is especially useful on stiff linear systems derived from elliptic PDEs which are strongly convection-dominated. The reason for the robustness of CGMN and CGNR is apparently due to the fact that both are based on the normal equations. A convection-dominated PDE, when discretized with central difference schemes, gives rise to large off-diagonal elements, whereas in the normal equations systems, the diagonal elements are usually the largest.

Several research directions and further applications are suggested by this work:

—Finding preconditioners for CGMN which will improve its performance, especially in cases such as problem 8.

—Application of CGMN to other problems requiring the solution of large sparse linear systems, such as computerized tomography and electron tomography—see [Fernández et al. 2008].

—Application of CGMN as a solver of intermediate linear systems obtained in several methods for solving nonlinear systems. For example, eigenvalue problems and CFD.

—Application of CGMN to other types of stiff linear systems with large off-diagonal elements.

ACKNOWLEDGMENTS

The authors are indebted to the anonymous reviewers whose many detailed and instructive comments were a great help in improving the paper.

REFERENCES

ARIOLI, M., DUFF, I. S., NOAILLES, J., AND RUIZ, D. 1992. A block projection method for sparse matrices. *SIAM J. on Scientific & Statistical Computing 13*, 47–70.

ARIOLI, M., DUFF, I. S., RUIZ, D., AND SADKANE, M. 1995. Block lanczos techniques for accelerating the block cimmino method. *SIAM J. on Scientific & Statistical Computing 16*, 1478–1511.

BJÖRCK, Å. AND ELFVING, T. 1979. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT 19*, 145–163.

BRAMLEY, R., CHEN, H.-C., MEIER, U., AND SAMEH, A. 1990. On some parallel preconditioned CG schemes. In *Proc. International Conf. on Preconditioned Conjugate Gradient Methods, Nijmegen, The Netherlands, June 1989*, O. Axelsson and L. Yu. Kolotilina, Eds. Lecture Notes in Mathematics, vol. 1457. Springer-Verlag, Berlin, 17–27.

BRAMLEY, R. AND SAMEH, A. 1991. Domain decomposition for parallel row projection algorithms. *Applied Numerical Mathematics 8*, 303–315.

BRAMLEY, R. AND SAMEH, A. 1992. Row projection methods for large nonsymmetric linear systems. *SIAM J. on Scientific & Statistical Computing 13*, 168–193.

CIMMINO, G. 1938. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. *La Ricerca Scientifica XVI, Series II, Anno IX 1*, 326–333.

EGGERMONT, P. P. B., HERMAN, G. T., AND LENT, A. 1981. Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. *Linear Algebra & Its Applications 40*, 37–67.

ELFVING, T. 1980. Block-iterative methods for consistent and inconsistent linear equations. *Numerische Mathematik 35*, 1–12.

ELFVING, T. 2004. A projection method for semidefinite linear systems and its applications. *Linear Algebra & its Applications 391*, 57–73.

ELMAN, H. AND GOLUB, G. 1990. Iterative methods for cyclically reduced non-self-adjoint linear systems. *Mathematics of Computation 54,* 190 (Apr.), 671–700.

FERNÁNDEZ, J.-J., GORDON, D., AND GORDON, R. 2008. Efficient parallel implementation of iterative reconstruction algorithms for electron tomography. *J. of Parallel & Distributed Computing 68,* 5 (May), 626–640.

FERZIGER, J. H. AND PERIĆ, M. 2002. *Computational Methods for Fluid Dynamics*, 3rd. ed. Springer-Verlag, Berlin.

FLETCHER, R. 1976. Conjugate gradient methods for indefinite systems. In *Proc. Dundee Biennial Conf. on Numerical Analysis, 1975*, G. A. Watson, Ed. Lecture Notes in Mathematics, vol. 506. Springer-Verlag, Berlin, 73–89.

GORDON, D. AND GORDON, R. 2005. Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems. *SIAM J. on Scientific Computing 27*, 1092–1117.

GORDON, D. AND GORDON, R. 2008. CARP-CG: a robust and efficient parallel solver for linear systems, applied to strongly convection-dominated elliptic partial differential equations. Tech. rep., Dept. of Computer Science, Univ. of Haifa, Israel. Dec. Submitted for publication. http://cs.haifa.ac.il/~gordon/carp-cg.pdf.

GORDON, D. AND MANSOUR, R. 2007. A geometric approach to quadratic optimization: an improved method for solving strongly underdetermined systems in CT. *Inverse problems in Science & Engineering 15,* 8 (Dec.), 811–826.

Gresho, P. and Sani, R. 1998. *Incompressible Flow and the Finite Element Method: Advection-Diffusion and Isothermal Laminar Flow*. John Wiley & Sons, Ltd., Chichester, England.

Herman, G. T. 1980. *Image Reconstruction From Projections: The Fundamentals of Computerized Tomography*. Academic Press, New York.

Herman, G. T., Lent, A., and Lutz, P. H. 1978. Relaxation methods for image reconstruction. *Communications of the ACM 21*, 152–158.

Hestenes, M. R. and Stiefel, E. 1952. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards 49*, 409–436.

Kaczmarz, S. 1937. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin de l'Académie Polonaise des Sciences et Lettres A35*, 355–357.

Kamath, C. and Sameh, A. 1989. A projection method for solving non-symmetric linear systems on multi processors. *Parallel Computing 9,* 3, 291–312.

Kamath, C. and Weeratunga, S. 1990. Implementation of two projection methods on a shared memory multiprocessor: DEC VAX 6240. *Parallel Computing 16*, 375–382.

Kamath, C. and Weeratunga, S. 1992. Projection methods for the numerical solution of non-self-adjoint elliptic partial differential equations. *Numerical Methods for Partial Differential Equations 8*, 59–76.

Kincaid, D. and Young, D. 1981. Adapting iterative algorithms developed for symmetric systems to non-symmetric systems. In *Elliptic Problem Solvers*, M. Schultz, Ed. Academic Press, New York, 353–359.

Kuck, D., Davidson, E., Lawrie, D., and Sameh, A. 1986. Parallel super-computing today and the cedar approach. *Science 231*, 967–974.

Lanczos, C. 1952. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards 49*, 33–53.

Saad, Y. 1990. SPARSKIT: a basic tool kit for sparse matrix computations. Tech. Rep. RIACS-90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA.

Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems*, 2nd. ed. SIAM, Philadelphia, PA.

Saad, Y. and Schultz, M. H. 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. on Scientific & Statistical Computing 7*, 856–869.

Sonneveld, P. 1989. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. on Scientific & Statistical Computing 10*, 36–52.

Tanabe, K. 1971. Projection method for solving a singular system of linear equations and its applications. *Numerische Mathematik 17*, 203–214.

Trummer, M. R. 1981. Reconstructing pictures from projections: On the convergence of the ART algorithm with relaxation. *Computing 26*, 189–195.

Tuminaro, R. S., Heroux, M. A., Hutchinson, S. A., and Shadid, J. N. 1999. AZTEC user's guide. Tech. Rep. SAND99-8801J, Sandia National Laboratories, Albuquerque, New Mexico.

van der Vorst, H. A. 1992. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. on Scientific & Statistical Computing 13*, 631–644.