AN ENGINEERING APPROACH TO GENERALIZED CONJUGATE GRADIENT METHODS AND BEYOND

Willi Schönauer, Rüdiger Weiss
Rechenzentrum der Universität Karlsruhe
Postfach 6980
D-76128 Karlsruhe
Germany
e-mail: schonauer@rz.uni-karlsruhe.de
tel.: +49/721/608-3754

1. Introduction, Normalization, Smoothing.

The purpose of this paper is to give for non-experts a simple introduction to generalized CG (conjugate gradient) methods as a starting paper for the lecture series. Therefore emphasis is put rather on the understanding than on a rigorous mathematical treatment. We want to solve a linear system of n equations

$$(1) \qquad Ax = b \ , \ residual \ for \ x_k \ is \ r_k = Ax_k - b \ .$$

We use an iterative method, x_k is the k-th iterate of x. We assume n very large, e.g. 10^6.

Normalization: CG-type methods are residual-reduction methods. If we multiply one of the n equations e.g. by 10^6, the solution does not change, but the residual $r_{k,i}$ of that equation i has 10^6-fold weight and dominates the whole iteration process. Therefore it ist mandatory for residual reduction methods to normalize the equations that they have equal weight. We found that in most (not all) cases normalization to absolute row sum equal to 1 and non-negative main diagonal was the best method. If $A'x = b'$ denotes the un-normalized system, the normalized system is

$$(2) \qquad DA'x = Db' \rightarrow Ax = b \ , \ D = diag \frac{1}{\sum_{k=1}^{n} |a'_{i,k}|} sign \ a'_{i,i} \ .$$

Some people call this already a preconditioning. We assume (1) to be normalized.

For a symmetric matrix A' normalization must preserve symmetry. In this case we normalize in the following way:

$$(3) \qquad D_s A' D_s y = D_s b' \ , \ x = D_s y \ , \ D_s = diag(\frac{1}{\sum_{k=1}^{n} |a'_{i,k}|})^{1/2} \ .$$

Smoothing: The residual of original generalized CG-type methods that do not minimize the true residual oscillates heavily. Therefore we compute parallel to and independent of the r_k-sequence a smoothed sequence s_k. The new smoothed vector s_{k+1} is a weighted mean of the previous s_k and the proposition r_{k+1} of the CG-type method (residual of x_{k+1}):

$$(4) \qquad s_{k+1} = \alpha_k s_k + (1 - \alpha_k) r_{k+1} = r_{k+1} + \alpha_k (s_k - r_{k+1}) \ ,$$

with α_k from the condition $s_{k+1}^2 = min$:

$$(5) \qquad \alpha_k = - \frac{r_{k+1}^T (s_k - r_{k+1})}{(s_k - r_{k+1})^2} \ .$$

Because s_{k+1} is a residual, $s_{k+1} = Ax_{k+1,smoothed} - b$, we get from (4) for the iterates

$$(6) \qquad x_{k+1,smoothed} = \alpha_k x_{k,smoothed} + (1 - \alpha_k) x_{k+1} \ .$$

A direct consequence is that

$$\|s_{k+1}\| \le \|r_{k+1}\|, \tag{7}$$

$$\|s_{k+1}\| \le \|s_k\|. \tag{8}$$

Smoothing has been introduced in [24, 23]. It turns out that the smoothed sequences have new and remarkable properties [34, 38, 41]. We used smoothing for a reasonable control of the iteration process and as a sequence from which a restart could be effected.

2. The basic approach to generalized CG.

The error e_k for an iterate x_k is

$$e_k = x - x_k = -A^{-1} r_k \tag{9}$$

$(A(x-x_k) = Ax-b-(Ax_k-b) = 0-r_k)$. What one wants to know during an iteration process is the error e_k. Unfortunately, only the residual r_k is known. Equation (9) relates these quantities and we are tempted to assume that a reduction of r_k also means a reduction of e_k. If the matrix A is ill-conditioned, i.e. A^{-1} is nearly singular, r_k may decrease and e_k may increase, see [34].

The solution of (1) can be obtained by minimizing the functional

$$F(x_k) = e_k^T A e_k = r_k^T A^{-1} r_k$$

whose gradient is $\partial F/\partial x_k = -2A(x-x_k) = 2r_k$, here A must be SPD (symmetric positive definite). The gradient direction suggests an approach $x_{k+1} = \alpha_k r_k + x_k$ (steepest descent). Multiplication by A and subtracting b on both sides yields $r_{k+1} = \alpha_k A r_k + r_k$. The coefficient α_k is determined from $F(x_{k+1}) = min$, $\alpha_k = -r_k^2/r_k^T A r_k$.

Now we generalize this approach for arbitary A (non-SPD). We operate with true residuals r_{k+1} and with pseudo-residuals $\overline{r}_{k+1}$ that are proportional to the true residuals (same direction, different length):

$$r_{k+1} = \varphi_k \overline{r}_{k+1}\ . \tag{10}$$

The basic approach combines a new pseudo-residual from a **search direction** d_k and p preceding true residuals in the following way:

$$\overline{r}_{k+1} = Ad_k + \alpha_{1,k} r_k + \alpha_{2,k} r_{k-1} + \cdots + \alpha_{p,k} r_{k-p+1}\ . \tag{11}$$

Now we want the corresponding formula for x_{k+1}. Using (10) for r_{k+1} and (1) for all r_k's yields from (11)

$$\frac{1}{\varphi_k} A x_{k+1} - \frac{b}{\varphi_k} = A(d_k + \alpha_{1,k} x_k + \cdots + \alpha_{p,k} x_{k-p+1}) - b(\alpha_{1,k} + \cdots + \alpha_{p,k})\ . \tag{12}$$

If we want an x-formula the b-terms on the l.h.s. and r.h.s. must cancel from which we get

$$\varphi_k = \frac{1}{\alpha_{1,k} + \cdots + \alpha_{p,k}}\ . \tag{13}$$

The remaining equation (12) is formally multiplied by $\varphi_k A^{-1}$:

$$x_{k+1} = \varphi_k (d_k + \alpha_{1,k} x_k + \cdots + \alpha_{p,k} x_{k-p+1})\ . \tag{14}$$

This way from r_{k+1} to x_{k+1} shows why we need Ad_k in (11) and why we used pseudo-residuals $\bar{r}_{k+1}$. We need Ad_k (and not d_k) in the approach (11) that we can multiply (12) formally by A^{-1}, and we need pseudo-residuals, i.e. the coefficient φ_k that we can use this additional degree of freedom to cancel the b-terms in (12).

The choice of the coefficients $\alpha_{i,k}$ is made that we get a "better" approximation. We get explicit formulas if we choose as target

$$\bar{r}_{k+1}^2 = min\ , \ \rightarrow \frac{\partial}{\partial \alpha_{i,k}} \bar{r}_{k+1}^2 = 2\bar{r}_{k+1}^T \frac{\partial \bar{r}_{k+1}}{\partial \alpha_{i,k}} = 0\ . \tag{15}$$

From (11) we get $\partial \bar{r}_{k+1}/\partial \alpha_{i,k} = r_{k-i+1}$, thus (15) induces

$$\bar{r}_{k+1}^T r_{k-i+1} = 0\ , \ \rightarrow r_{k+1}^T\ r_{k-i+1} = 0 \tag{16}$$

because of (10). The new r_{k+1} is orthogonal to all preceding vectors that appear in the approach (11). This is an essential property of CG-type methods to build up an orthogonal basis. Some authors use this property to define methods. From (15) we get

$$\alpha_{i,k} = -r_{k-i+1}^T A d_k / r_{k-i+1}^2\ . \tag{17}$$

If we use $r_{k+1}^2 = min$ (true residual) instead of (15) we get (by a sophisticated trick) a linear system of p equations for the $\alpha_{i,k}$. We use the approach (10) to (14). But now we determine the coefficients $\alpha_{i,k}$ from the condition

$$r_{k+1}^2 = r_{k+1}^T\ r_{k+1} = min\ , \ \rightarrow r_{k+1}^T \frac{\partial r_{k+1}}{\partial \alpha_{i,k}} = 0\ . \tag{18}$$

Using (10) in the derivative yields

$$r_{k+1}^T \frac{\partial(\varphi_k \bar{r}_{k+1})}{\partial \alpha_{i,k}} = r_{k+1}^T \left(\frac{\partial \varphi_k}{\partial \alpha_{i,k}} \bar{r}_{k+1} + \varphi k \frac{\partial \bar{r}_{k+1}}{\partial \alpha_{i,k}} \right) = 0\ . \tag{19}$$

With (13) we get $\partial \varphi_k / \partial \alpha_{i,k} = -\varphi_k^2$, thus (19) yields

$$r_{k+1}^T(-\varphi_k^2 \bar{r}_{k+1} + \varphi_k r_{k-i+1}) = 0\ .$$

By use of (10) for the φ_k^2-term and multiplication by $1/\varphi_k$ we get

$$r_{k+1}^T(-r_{k+1} + r_{k-i+1}) = 0\ . \tag{20}$$

The equations (20) for $i = 1$ to $i = p$ are again orthogonality relations. They induce that the correction vectors $r_{k+1} - r_{k-i+1}$ between the new residual and the old residuals r_{k-i+1} are orthogonal to the new residual r_{k+1}. Equations (20) are p equations for the p coefficients $\alpha_{i,k}$. Unfortunately, these equations are highly nonlinear because of the nonlinear term $r_{k+1}^T r_{k+1}$ (using (10) introduces $\bar{r}_{k+1}$, using (11) introduces the $\alpha_{i,k}$). We will now show how we can get linear equations for the $\alpha_{i,k}$.

We write down the first two and the last of the equations (20):

$$\begin{array}{lll} i = 1 & r_{k+1}^T(-r_{k+1} + r_k) & = 0\ , \\ i = 2 & r_{k+1}^T(-r_{k+1} + r_{k-1}) & = 0\ , \\ & \vdots & \\ i = p & r_{k+1}^T(-r_{k+1} + r_{k-p+1}) & = 0\ . \end{array}$$

Now we subtract from each equation the preceding equation, starting with i = 2:

$$
\begin{aligned}
r_{k+1}^T(-r_k + r_{k-1}) &= 0 \; , \\
r_{k+1}^T(-r_{k-1} + r_{k-2}) &= 0 \; , \\
&\;\vdots \\
r_{k+1}^T(-r_{k-p+2} + r_{k-p+1}) &= 0 \; .
\end{aligned}
\tag{21}
$$

If we multiply the equations by $1/\varphi_k$, use $r_{k+1}^T/\varphi_k = \overline{r}_{k+1}$ (10) and replace $\overline{r}_{k+1}$ by (11) we get $p-1$ linear equations for the p coefficients $\alpha_{i,k}$. The missing p-th equation is obtained by multiplication of the equations (20) by $\varphi_k \alpha_{i,k}$ for equation i and summing up:

$$
\begin{aligned}
r_{k+1}^T \varphi_k [\alpha_{1,k}(-r_{k+1} + r_k) + \alpha_{2,k}(-r_{k+1} + r_{k-1}) + \cdots & \\
+\alpha_{p,k}(-r_{k+1} + r_{k-p+1})] &= 0 \\
-\varphi_k(\alpha_{1,k} + \alpha_{2,k} + \cdots + \alpha_{p,k}) r_{k+1}^T r_{k+1} + & \\
+r_{k+1}^T \varphi_k(\alpha_{1,k} r_k + \alpha_{2,k} r_{k-1} + \cdots + \alpha_{p,k} r_{k-p+1}) &= 0 \; .
\end{aligned}
$$

Using (13) ($\varphi_k(\alpha_{i,k} + \cdots + \alpha_{p,k}) = 1$) and taking $\varphi_k r_{k+1}^T$ out of the parentheses yields

$$
\varphi_k r_{k+1}^T(-\frac{1}{\varphi_k} r_{k+1} + \alpha_{1,k} r_k + \cdots + \alpha_{p,k} r_{k-p+1}) = 0 \; .
$$

With (10) we have $r_{k+1}/\varphi_k = \overline{r}_{k+1}$ and using (11) for $\overline{r}_{k+1}$ yields

$$
\varphi_k r_{k+1}^T(-Ad_k) = 0 \; .
$$

Multiplication by $-1/\varphi_k^2$ and again using (10) yields finally

$$
\overline{r}_{k+1}^T A d_k = 0 \; . \tag{22}
$$

This is again an orthogonality relation. It means that $\overline{r}_{k+1}$ and thus r_{k+1} (because of (10)) is orthogonal to Ad_k, or in other words r_{k+1} is A-conjugate to the search direction d_k. If we take $d_k = r_k$, i.e. the search direction is the old residual, the new residual is orthogonal to Ar_k.

If we replace in (22) $\overline{r}_{k+1}$ by (11) we get the p-th linear equation for the $\alpha_{i,k}$. The first and last equation resulting from (21) and equation (22) yield the following linear system (if we transpose the equations for conveniency)

$$
\begin{aligned}
(-r_k^T + r_{k-1}^T)(Ad_k + \alpha_{1,k} r_k + \cdots + \alpha_{p,k} r_{k-p+1}) &= 0 \; , \\
&\;\vdots \\
(-r_{k-p+2}^T + r_{k-p+1}^T)(Ad_k + \alpha_{1,k} r_k + \cdots + \alpha_{p,k} r_{k-p+1}) &= 0 \; , \\
(Ad_k)^T(Ad_k + \alpha_{1,k} r_k + \cdots + \alpha_{p,k} r_{k-p+1}) &= 0 \; .
\end{aligned}
\tag{23}
$$

This is a full system of p linear equations whose p^2 coefficients and r.h.s. are scalar products, e.g. $(-r_k^T + r_{k-1}^T) r_k$ for the first coefficient and $(r_k^T - r_{k-1}^T) A d_k$ as the first r.h.s. If p, the number of terms with preceding residuals in (11) is large, the computation of the coefficients and the solution of the linear system is rather expensive. This is the price that must be paid if we want to minimize the true residual by the condition (18). If we minimize at the other hand the pseudo-residual by the condition (15), we get explicit formulas for the $\alpha_{i,k}$, but the true residual may oscillate heavily and we must combine this procedure with smoothing (4).

In both types of algorithms we have to compute Ad_k. This matrix-vector multiplication (MVM) determines essentially the computation of an iteration step. Remember that in technical applications the matrix A is extremely large and sparse.

In the following we go back to the discussion of pseudo-residual methods, i.e. we assume that the $\alpha_{i,k}$ are computed from the principle (15) and by the explicit formulas (17), i.e. we minimize the pseudo-residual.

Search direction $d_k = r_k$: This choice is stimulated by the method of steepest descent (p = 1, one term) and the "classical" CG, see below. We select a starting vector x_0, then $r_0 = Ax_0 - b$. The method can start with p = 1 for x_1, p = 2 for x_2 etc. **Exact sequence:** we take all preceding residuals, i.e. p = k + 1. Then we should get the exact solution x after n iterations because the orthogonality relations (16) hold, but only n linearly independent orthogonal vectors can be built up. In reality rounding errors destroy the orthogonalities. Note that we needed to store n vectors. This method is called ORTHORES (orthogonal residuals) [15], but it is impracticable. Nevertheless at least theoretically the solution for an arbitrary nonsingular linear system could be obtained after n iterations, i.e. by n MVMs. If we **truncate** the method so that we take only the last p residuals, the convergence deteriorates drastically. We found by "numerical engineering" experiments for different types of linear systems, that to take p = 5 terms and to make a **restart** after 20 iterations from the smoothed sequence yields an optimal method for matrices A with sufficiently clustered eigenvalues. We call this method **PRES20**. Thus PRES20 is a truncated ORTHORES (5) method that is restarted from the smoothed sequence (4), (6) [23, 24].

If A is SPD, from (17) follows for $d_k = r_k$ that the coefficients $\alpha_{i,k} = 0$ for i > 2 because of inherent orthogonalities so that we get an "exact" sequence although we have only two α-terms in (11). This is a variant of the "classical" CG. We want to show for a specific coefficient $\alpha_{i,k}$ the vanishing for i > 2 to illustrate the principle (the generalization to arbitrary i is possible in the same way but it is less transparent). From (11) we get for k = 2

$$\overline{r}_3^T = Ar_2 + \alpha_{1,2}r_2 + \alpha_{2,2}r_1 + \alpha_{3,2}r_0 \ ,$$

with $\alpha_{3,2} = -r_0^T A r_2 / r_0^2$ from (17) for i = 3.

We want to show $\alpha_{3,2} = 0$. The trick is to "eliminate" A in the expression $r_0^T A$. For k = 0 (11), (13) yield

$$\overline{r}_1 = Ar_0 + \alpha_{1,0}r_0 \quad , \quad \varphi_0 = 1/\alpha_{1,0} \ .$$

With (10) we get

$$r_1 = \varphi_0 \overline{r}_1 = \frac{1}{\alpha_{1,0}} Ar_0 + r_0 \ .$$

Resolving for Ar_0 and transposing yields

$$Ar_0 = \alpha_{1,0}(r_1 - r_0) \quad , \quad r_0^T A^T = \alpha_{1,0}(r_1^T - r_0^T) \ .$$

Now we need the symmetry of A: $A = A^T$, i.e. $r_0^T A^T = r_0^T A$. In $\alpha_{3,2}$ we have

$$r_0^T A r_2 = r_0^T A^T r_2 = \alpha_{1,0}(r_1^T - r_0^T) r_2 = \alpha_{1,0}(r_1^T r_2 - r_0^T r_2) = 0$$

because from (16) we have $r_1^T r_2 = 0$, $r_0^T r_2 = 0$. Thus follows $\alpha_{3,2} = 0$ from the inherent orthogonalities. This principle can be generalized for arbitrary i > 2. This means that we have for A SPD an "exact" sequence that orthogonalizes to all preceding residuals although we have only 2 terms in the approach (11). Unfortunately, rounding errors destroy these orthogonalities in the finite arithmetic of the computer. Nevertheless this is an obvious access to the classical CG.

Search direction $d_k = A^T r_k$: This is Craig's method [5]. With our smoother for the control (no feedback on the original sequence) we call this method ATPRES. It turns out to be the most robust, unfortunately a very slowly converging method of those methods we investigated. Weiss [36] has shown that ATPRES minimizes the true residual. Without smoothing Craig's method minimizes the error e_k (9). So this is the example of an error minimizing method, although we started from minimizing the pseudo-residual. The additional property of error minimization results from the search direction $d_k = A^T r_k$. Now the iteration matrix is AA^T because we have in (11) $Ad_k = AA^T r_k$ which needs two matrix-vector multiplications per iteration step that must be executed consecutively: $v = A^T r_k$, Av (it would be much more expensive to compute AA^T for sparse A). Because AA^T is SPD we get again an "exact" method with only two non-zero α-terms. Its behavior is quite similar to the least squares CG: application of classical CG to $A^T Ax = A^T b$ (multiplication of (1) by A^T).

Biconjugate Gradient method: We introduce the auxiliary system $A^T y = b$ with residual $r_k^* = A^T y_k - b$ for the iterate y_k and make the approach for the pseudo-residuals

$$\begin{aligned} \overline{r}_{k+1} &= Ar_k + \alpha_{1,k} r_k + \alpha_{2,k} r_{k-1} + \alpha_{3,k} r_{k-2} + \cdots \ , \\ \overline{r}^*_{k+1} &= A^T r^*_k + \alpha_{1,k} r^*_k + \alpha_{2,k} r^*_{k-1} + \alpha_{3,k} r^*_{k-2} + \cdots \end{aligned} \tag{24}$$

For r_{k+1} the relation (10) and similarly for $r^*_{k+1} = \varphi_k \overline{r}^*_{k+1}$ holds, with φ_k from (13). The corresponding solution vector x_{k+1} is given by (14) with $d_k = r_k$. The value of y_{k+1} is not needed. The coefficients $\alpha_{i,k}$ are determined from the condition

$$\overline{r}^T_{k+1}\overline{r}^*_{k+1} = min \ , \ \frac{\partial(\overline{r}^T_{k+1}\overline{r}^*_{k+1})}{\partial\alpha_{i,k}} = \frac{\partial\overline{r}^T_{k+1}}{\partial\alpha_{i,k}}\overline{r}^*_{k+1} + \overline{r}^T_{k+1}\frac{\partial\overline{r}^*_{k+1}}{\partial\alpha_{i,k}} = 0 \ . \tag{25}$$

Here we see that we have a "weak solution" of $\overline{r}_{k+1} = 0$ by the weight function $\overline{r}^*_{k+1}$. This is the only reason for the introduction of the second sequence $\overline{r}^*_{k+1}$. The differential terms in (25), using (24), yield the corresponding vectors r^T_{k-i+1} and r^*_{k-i+1} that are associated with $\alpha_{i,k}$. So we get from (25), transposing the second term ($scalar^T = scalar$)

$$r^T_{k-i+1}\overline{r}^*_{k+1} + r^{*T}_{k-i+1}\overline{r}_{k+1} = 0 \ . \tag{26}$$

This could be called a "generalized orthogonality" because only this combination of the two scalar products is zero. From (26) follows that for any pair of residuals holds

$$r^T_j\overline{r}^*_i + r^{*T}_j\overline{r}_i = 0 \ , \rightarrow r^T_j r^*_i + r^{*T}_j r_i = 0 \ , \ i \neq j \ , \tag{27}$$

because of (10) and the corresponding relation for the stared residuals.

The i relations (26) yield i equations for the i coefficients $\alpha_{i,k}$. The i-th equation is

$$\begin{aligned} & r^T_{k-i+1}(A^T r^*_k + \alpha_{1,k} r^*_k + \alpha_{2,k} r^*_{k-1} + \alpha_{3,k} r^*_{k-2} + \cdots) + \\ & + r^{*T}_{k-i+1}(Ar_k + \alpha_{1,k} r_k + \alpha_{2,k} r_{k-1} + \alpha_{3,k} r_{k-2} + \cdots) \quad = \quad 0 \ . \end{aligned} \tag{28}$$

We sort (28) for the $\alpha_{i,k}$:

$$\begin{aligned} & \alpha_{1,k}(r_{k-i+1}^T r_k^* + r_{k-i+1}^{*T} r_k) + \alpha_{2,k}(r_{k-i+1}^T r_{k-1}^* + r_{k-i+1}^{*T} r_{k-1}) + \\ & \qquad + \alpha_{3,k}(\cdots) + \cdots = -(r_{k-i+1}^T A^T r_k^* + r_{k-i+1}^{*T} A r_k) \ . \end{aligned} \tag{29}$$

This is the equation for general i. We write down explicitly the equations for i = 1,2:

$$i = 1 :$$

$$\begin{aligned} & \alpha_{1,k}(r_k^T r_k^* + r_k^{*T} r_k) + \alpha_{2,k}(r_k^T r_{k-1}^* + r_k^{*T} r_{k-1}) + \\ & + \alpha_{3,k}(\cdots) + \cdots = -(r_k^T A^T r_k^* + r_k^{*T} A r_k) \ , \end{aligned} \tag{30}$$

$$i = 2 :$$

$$\begin{aligned} & \alpha_{1,k}(r_{k-1}^T r_k^* + r_{k-1}^{*T} r_k) + \alpha_{2,k}(r_{k-1}^T r_{k-1}^* + r_{k-1}^{*T} r_{k-1}) + \\ & + \alpha_{3,k}(\cdots) + \cdots = -(r_{k-1}^T A^T r_k^* + r_{k-1}^{*T} A r_k) \ . \end{aligned} \tag{31}$$

For i = 1 only the coefficient of $\alpha_{1,k}$ and for i = 2 only the coefficient of $\alpha_{2,k}$ is non-zero, all other coefficients are zero because of the generalized orthogonality (27). Below we show that the $\alpha_{i,k} = 0$ for i > 2, i.e. we obtain an exact sequence that orthogonalizes in the sense of (27) to all preceding residuals although there are only two terms in the approach (24).

Now we want to show that $\alpha_{3,2} = 0$, this is the first $\alpha_{i,k}$ with i > 2 that appears in (24). The method can be extended to general i basically in the same way. The extension of (30), (31) to i = 3 for the special value k = 2 yields

$$\alpha_{3,2} 2 r_0^{*T} r_0 = -(r_0^T A^T r_2^* + r_0^{*T} A r_2) \ . \tag{32}$$

Similarly to the procedure for the "PRES-CG" we must try to eliminate A^T and A so that we can use the generalized orthogonality relations. For k = 0 we get from (24)

$$\overline{r}_1 = A r_0 + \alpha_{1,0} r_0 \ , \ \overline{r}_1^* = A^T r_0^* + \alpha_{1,0} r_0^* \ , \tag{33}$$

and from (13) we get $\varphi_0 = 1/\alpha_{1,0}$. With (10) and the corresponding stared relation we get

$$r_1 = \varphi_0 \overline{r}_1 = \frac{1}{\alpha_{1,0}} A r_0 + r_0 \ , \ r_1^* = \varphi_0 \overline{r}_1^* = \frac{1}{\alpha_{1,0}} A^T r_0^* + r_0^* \ .$$

We resolve for the A- and A^T- terms:

$$A r_0 = \alpha_{1,0}(r_1 - r_0) \ , \ A^T r_0^* = \alpha_{1,0}(r_1^* - r_0^*) \ .$$

If we transpose these equations we get the corresponding factors in the r.h.s. of (32), which now becomes

$$\begin{aligned} r_0^T A^T r_2^* + r_0^{*T} A r_2 &= \alpha_{1,0}(r_1^T r_2^* - r_0^T r_2) + \alpha_{1,0}(r_1^{*T} r_2 - r_0^{*T} r_2) \\ &= \alpha_{1,0}[(r_1^T r_2^* + r_1^{*T} r_2) - (r_0^T r_2^* + r_0^{*T} r_2)] = 0 \ . \end{aligned}$$

In the last parentheses we have again the generalized orthogonality relations (27). So from (32) follows $\alpha_{3,2} = 0$.

If one follows the derivation of this result one can see that this could not be obtained if we had used in the approach (24) general search directions d_k and d_k^*, i.e. terms $A d_k$ and $A^T d_k^*$, instead of the search directions r_k and r_k^*. This means that in this case we do not get an exact sequence that breaks up automatically after two terms. Then we had a situation like for PRES20 where we must truncate the sequences after p terms and eventually do a restart from the smoothed sequence. Because we do not minimize the true residual by the condition (25) the true residual may oscillate heavily and we use the smoothed sequence for control, but we have no feedback for the exact sequence.

For the exact sequence, i.e. the approach (24), we get the coefficients (using (30, 31))

$$\alpha_{i,k} \begin{cases} -\frac{r_{k-i+1}^T A^T r_k^* + r_{k-i+1}^{*T} A r_k}{2 r_{k-i+1}^T r_{k-i+1}^*} & , \text{i=1,2} \\ 0 & , \text{i} > 0 \end{cases} \quad . \tag{34}$$

The starting of the iteration process is by selecting x_0, then $r_0 = Ax_0 - b$, $r_0^* = r_0$.

From the above relations follows that

$$r_i^T r_j^* = r_j^{*T} r_i = 0 \; for \; i \neq j \; , \tag{35}$$

i.e. the r- and r^*-vectors are mutually orthogonal for unequal indices (biconjugate). This cannot be seen directly but needs some computation. We want to show the above relation for two examples. At first we want to show that $r_0^T r_1^* = 0$ and $r_0^{*T} r_1 = 0$.

For i = 1, j = 0 follows from (27)

$$r_0^T \overline{r}_1^* + r_0^{*T} \overline{r}_1 = 0 \; . \tag{36}$$

Replacing $\overline{r}_1$, $\overline{r}_1^*$ by (33) we get

$$\begin{gathered} r_0^T (A^T r_0^* + \alpha_{1,0} r_0^*) + r_0^{*T} (A r_0 + \alpha_{1,0} r_0) = 0 \; , \\ r_0^T A^T r_0^* + \alpha_{1,0} r_0^T r_0^* + \underbrace{r_0^{*T} A r_0}_{r_0^T A^T r_0^*} + \alpha_{1,0} \underbrace{r_0^{*T} r_0}_{r_0^T r_0^*} = 0 \; . \end{gathered}$$

Summing up, dividing by 2 and taking r_0^T out of the parentheses yields

$$r_0^T \underbrace{(A^T r_0^* + \alpha_{1,0} r_0^*)}_{\overline{r}_1^* \; from \; (33)} = 0 \; . \tag{37}$$

Thus we get with (10)

$$r_0^T \overline{r}_1^* = r_0^T \frac{1}{\varphi_0} r_1^* = 0 \; \Rightarrow \; r_0^T r_1^* = 0 \; . \tag{38}$$

Transposing (37) and taking r_0^* out of the parentheses yields similarly

$$r_0^{*T} \underbrace{(A r_0 + \alpha_{1,0} r_0)}_{\overline{r}_1} = 0 \; \Rightarrow \; r_0^{*T} r_1 = 0 \; . \tag{39}$$

In a similar way we now want to show that $r_0^{*T} r_2 = 0$ holds. For k = 1 we get from (24)

$$\overline{r}_2 = A r_1 + \alpha_{1,1} r_1 + \alpha_{2,1} r_0 \quad , \quad \overline{r}_2^* = A^T r_1^* + \alpha_{1,1} r_1^* + \alpha_{2,1} r_0^* \; . \tag{40}$$

From (27) we get for i = 2, j = 0

$$r_0^T \overline{r}_2^* + r_0^{*T} \overline{r}_2 = 0 \; . \tag{41}$$

We replace $\overline{r}_2^*$ by (40), but do not replace $\overline{r}_2$. (If we would replace $\overline{r}_2$ and leave $\overline{r}_2^*$ we would get the relation $r_0^T r_2^* = 0$ in a similar way.) We get

$$r_0^T(A^T r_1^* + \alpha_{1,1} r_1^* + \alpha_{2,1} r_0^*) + r_0^{*T}\overline{r}_2 = 0 \ ,$$
$$r_0^T A^T r_1^* + \underbrace{\alpha_{1,1} r_0^T r_1^*}_{0\ (38)} + \alpha_{2,1} r_0^T r_0^* + r_0^{*T}\overline{r}_2 = 0 \ .$$

We resolve the first equation of (33) for Ar_0 and transpose the result to replace $r_0^T A^T$ in the first term of the above equation:

$$\overline{r}_1^T r_1^* - \underbrace{\alpha_{1,0} r_0^T r_1^*}_{0\ (38)} + \alpha_{2,1} r_0^T r_0^* + r_0^{*T}\overline{r}_2 = 0 \ . \tag{42}$$

Now we want to process the $\alpha_{2,1}$-term. From (34) we get for i = 2, k = 1

$$\alpha_{2,1} r_0^T r_0^* = -\frac{r_0^T A^T r_1^* + r_0^{*T} A r_1}{2 r_0^T r_0^*} r_0^T r_0^* \ .$$

Replacing $r_0^T A^T$ like above and similarly $r_0^{*T} A$ by the second equation of (33) we get

$$\alpha_{2,1} r_0^T r_0^* = -\frac{1}{2}(\overline{r}_1^T r_1^* - \underbrace{\alpha_{1,0} r_0^T r_1^*}_{0\ (38)} + \overline{r}_1^{*T} r_1 - \underbrace{\alpha_{1,0} r_0^{*T} r_1}_{0\ (38)}) \ .$$

We have with (10)

$$\overline{r}_1^{*T} r_1 = \frac{1}{\varphi_0} r_1^{*T} r_1 = r_1^{*T}\left(\frac{1}{\varphi_0} r_1\right) = r_1^{*T}\overline{r}_1 = \overline{r}_1^T r_1^* \ .$$

Thus we get

$$\alpha_{2,1} r_0^T r_0^* = -\frac{1}{2} 2\overline{r}_1^T r_1^* = -\overline{r}_1^T r_1^*$$

and in (42) this term cancels with the first term and we get from (42), using (10)

$$r_0^{*T}\overline{r}_2 = 0 \ , \rightarrow r_0^{*T} r_2 = 0 \ . \tag{43}$$

If we had used instead of (41) the relation that follows from (27) for i = 2, j = 1

$$r_1^T \overline{r}_2^* + r_1^{*T}\overline{r}_2 = 0$$

we would get in a similar way $r_1^{*T} r_2 = 0$ and $r_1^T r_2^* = 0$.

This rather elementary and detailed exposition should demonstrate the tools that are needed to show the biorthogonality of the r_i and r_j^* vectors for $i \neq j$. This procedure can be generalized to an induction from k to k + 1, using (38), (39) as initial condition. This procedure also demonstrates that for this approach the biorthogonality relation (34) is a "secondary" property compared to the "primary" property of generalized orthogonality (27). Because the biorthogonality (35) results by a lengthy process in which the additive cancelling of some terms is needed, this property gets easily lost by rounding errors in the iteration process.

Fletcher [6] has reformulated BICO in four one-term sequences instead of the two two-term sequences (24). Usually this is what is called the biconjugate gradient method. Controlled by our smoother (4) we call this the BICO method. It turns out that this is a fast method if the complex eigenvalues of A are scattered in the positive real halfplane.

Up to now we have presented with all details some generalized CG methods that are all based on the fundamental approach (11). The reader should be able to follow the development of the corresponding set of formulas. However, there is a wide range of CG-type methods and methods that go beyond. In the following we give a verbal description of the most important of these methods. The theoretical investigation of these methods, i.e. of their convergence properties, is not easy and obvious. Weiß [36] has developed an unified theory that encompasses all these methods.

3. Different Implementations.

Iterative methods can be differently implemented. We consider the methods to be the same if they are mathematically equivalent in exact arithmetic. In finite precision different implementations can be more or less stable. There are many investigations on the stability of different implementations of distinct methods, see for example [1, 19, 20, 33]. We will show how to obtain different implementations roughly. From the equations (10) and (11) we get with $d_k = r_k$

$$r_{k+1} = \varphi_k \left[A r_k + \alpha_{1,k} r_k + \cdots + \alpha_{p,k} r_{k-p+1} \right]. \tag{44}$$

If we apply (44) for $r_k, \ldots, r_{k-p+1}$ and insert again into (44), then we get

$$\begin{aligned} r_{k+1} &= \varphi_k \left(A\varphi_{k-1} \left[A r_{k-1} + \alpha_{1,k-1} r_{k-1} + \cdots + \alpha_{p,k-1} r_{k-p} \right] \right. \\ &\quad + \alpha_{1,k}\varphi_{k-1} \left[A r_{k-1} + \alpha_{1,k-1} r_{k-1} + \cdots + \alpha_{p,k-1} r_{k-p} \right] \\ &\quad + \cdots \\ &\quad \left. + \, \alpha_{p,k}\varphi_{k-p} \left[A r_{k-p} + \alpha_{1,k-1} r_{k-p} + \cdots + \alpha_{p,k-p} r_{k-2p+1} \right] \right) \\ &= \varphi_k \left(\varphi_{k-1} \left[A^2 r_{k-1} + \alpha_{1,k-1} A r_{k-1} + \cdots + \alpha_{p,k-1} A r_{k-p} \right] \right. \\ &\quad + \alpha_{1,k}\varphi_{k-1} \left[A r_{k-1} + \alpha_{1,k-1} r_{k-1} + \cdots + \alpha_{p,k-1} r_{k-p} \right] \\ &\quad + \cdots \\ &\quad \left. + \, \alpha_{p,k}\varphi_{k-p} \left[A r_{k-p} + \alpha_{1,k-1} r_{k-p} + \cdots + \alpha_{p,k-p} r_{k-2p+1} \right] \right). \end{aligned}$$

Succesive application of (44) finally yields

$$r_{k+1} = \sum_{i=1}^{k+1} \beta_i A^i r_0 + r_0. \tag{45}$$

If we multiply (45) formally by A^{-1} we obtain

$$\begin{aligned} A^{-1} r_{k+1} = A^{-1}(A x_{k+1} - b) &= A^{-1} \left(\sum_{i=1}^{k+1} \beta_i A^i r_0 + r_0 \right) \\ &= \sum_{i=1}^{k+1} \beta_i A^{i-1} r_0 + A^{-1}(A x_0 - b), \end{aligned}$$

and we can calculate the approximations by

$$x_{k+1} = \sum_{i=1}^{k+1} \beta_i A^{i-1} r_0 + x_0. \tag{46}$$

All here discussed methods, except GMERR, construct the approximations so that (46) is valid, i. e. the approximations are in a shifted space that is spanned by the vectors $A^i r_0$. This space is called Krylov space. It can be generated by different bases and a distinct iterative method can be formulated so that different bases are used leading to different mathematically equivalent implementations. We will give in the sequel the two most important methods for the construction of a basis of the Krylov space.

Arnoldi process.

An orthonormal vector sequence v_{k+1} is constructed successsively by the Arnoldi process with

$$span(v_0, \ldots, v_k) = span(r_0, \ldots, A^k r_0) \tag{47}$$

and

$$v_i^T v_j \quad = \quad \delta_{i,j} \tag{48}$$

for $i, j = 0, \ldots, k$, where $\delta_{i,j}$ is Kronecker's delta. If we start with

$$v_0 = \frac{r_0}{||r_0||},$$

then the vectors can be calculated by the following recurrence for $k \geq 1$

$$\bar{v}_k \quad = \quad A v_{k-1} + \sum_{i=1}^{k} \alpha_{i,k} v_{k-i}, \tag{49}$$

$$v_k \quad = \quad \frac{1}{||\bar{v}_k||} \bar{v}_k. \tag{50}$$

Here the coefficients are determined from (48)

$$\alpha_{i,k} \quad = \quad -v_{k-i}^T A v_{k-1}. \tag{51}$$

A more stable implementation for the construction of orthonormal vectors is based on Householder transformations.

From (46) and (47) follows that the approximations of any iterative method based on a Krylov subspace for the solution of (1) can be calculated by

$$x_{k+1} = \sum_{i=1}^{k+1} \beta_i A^{i-1} r_0 + x_0 = \sum_{i=0}^{k} \gamma_i v_i + x_0. \tag{52}$$

We obtain from (52) for the residuals

$$r_{k+1} = \sum_{i=0}^{k} \gamma_i A v_i + r_0. \tag{53}$$

The coefficients γ_i are determined by inserting (53) into the special minimization or orthogonalization condition. If we choose for example the residual-minimizing condition (18) $r_{k+1}^2 = min$, then we obtain

$$\frac{\partial r_k^2}{\partial \gamma_j} = 0, \tag{54}$$

i. e.

$$\begin{aligned}\frac{\partial r_k^2}{\partial \gamma_j} &= r_{k+1}^T \frac{\partial r_k}{\partial \gamma_j} \\ &= r_{k+1}^T A v_j = 0.\end{aligned}$$

Thus by applying (53) the γ_i can be calculated by the solution of the linear system

$$\sum_{i=0}^{k} \gamma_i v_i^T A^T A v_j + r_0^T A v_j = 0. \tag{55}$$

Lanczos process.

In 1950, Lanczos introduced a method for the successive transformation of a given matrix to a tridiagonal matrix [17]. The purpose was to calculate the eigenvalues by a successive procedure. If we apply the nonsymmetric Lanczos process to the Krylov spaces generated by A applied to r_0 and A^T applied to r_0^*, then the algorithm constructs the two vector sequences v_k and w_k with

$$span(v_0, \ldots, v_k) = span(r_0, \ldots, A^k r_0), \tag{56}$$

$$span(w_0, \ldots, w_k) = span(r_0^*, \ldots, \left[A^T\right]^k r_0^*), \tag{57}$$

and

$$w_i^T v_j = \delta_{i,j} \tag{58}$$

for $i, j = 0, \ldots, k$, where $\delta_{i,j}$ is Kronecker's delta. If we start with

$$v_0 = \frac{r_0}{\eta_0}, \; w_0 = \frac{r_0^*}{\mu_0},$$

where η_0 and μ_0 are such that $\eta_0 \mu_0 = r_0^T r_0^*$, then the vectors can be calculated by a 3-term recurrence for $k \geq 2$

$$\bar{v}_k = A v_{k-1} - \alpha_k v_{k-1} - \beta_k v_{k-2}, \tag{59}$$

$$\bar{w}_k = A^T w_{k-1} - \alpha_k w_{k-1} - \gamma_k w_{k-2}, \tag{60}$$

$$v_k = \frac{1}{\eta_k} \bar{v}_k, \tag{61}$$

$$w_k = \frac{1}{\mu_k} \bar{w}_k. \tag{62}$$

Here the coefficients are determined from (58)

$$\alpha_k = w_{k-1}^T A v_{k-1}, \tag{63}$$

$$\beta_k = w_{k-2}^T A v_{k-1}, \tag{64}$$

$$\gamma_k = v_{k-2}^T A^T w_{k-1}, \tag{65}$$

and η_k and μ_k are such that (58) is fulfilled, i. e.

$$\eta_k \mu_k = \bar{w}_k^T \bar{v}_k. \tag{66}$$

For $k = 1$ use the same formulas but omit the terms with β_1 and γ_1.

Note that the algorithm breaks down if $\bar{w}_k^T \bar{v}_k = 0$ because then $\eta_k = 0$ or $\mu_k = 0$ by (66) and a division by zero occurs in (61) or (62). Remedies to avoid breakdowns were proposed by Parlett, Taylor and Liu [18]. They are based on block Lanczos algorithms called look-ahead strategies. These strategies were investigated by Gutknecht [14], Brezinski, Redivo-Zaglia and Sadok [1, 2, 3], by Freund, Gutknecht and Nachtigal [8] and by many others.

If A is symmetric, then the two sequences v_k and w_k coincide for $r_0^* = r_0$ resulting in the symmetric Lanczos process that requires the half of operations and storage. It is equivalent to the Arnoldi process.

From (46) and (56) follows that the approximations of any Krylov subspace-based iterative method for the solution of (1) can be calculated by

$$x_{k+1} = \sum_{i=1}^{k+1} \beta_i A^{i-1} r_0 + x_0 = \sum_{i=0}^{k} \mu_i v_i + x_0, \tag{67}$$

and the residuals by (67)

$$r_{k+1} = \sum_{i=0}^{k} \mu_i A v_i + r_0. \tag{68}$$

The coefficients μ_i are determined by inserting (68) into the special minimization or orthogonalization condition similarly as described for the Arnoldi process.

4. Overview of and relations between different CG-type methods.

In this section we give an overview of the most important CG-type methods and we discuss the relations between these methods. We do this as far as possible verbally, because it would be far beyond this paper to give all the formulas for the methods to be discussed. Nevertheless, some formulas must be presented in order to make clear some special points.

4.1. ORTHORES-based Methods.

In this section we describe direct generalizations of the classical CG method for nonsymmetric matrices.

Classical CG. Hestenes and Stiefel introduced classical CG [16] for the solution of systems with a symmetric, positive definite matrix. An easy-to-read investigation is given by Shewchuk [28]. The method is given by the equations (10) – (15) with $d_k = r_k$ and $p = 2$.

ORTHORES. The orthogonal residual method (ORTHORES) [40] is described by the equations (10) – (15) with $d_k = r_k$. For large non-symmetric matrices the exact ORTHORES method (with storing of all x_k and r_k) is not feasible because the storage requirements and the computational work increase with the iteration. Therefore truncated or restarted or truncated and restarted versions have to be applied. Truncated ORTHORES methods here are denoted by OR(p). We will investigate OR(5), a truncated version with p=5, i. e. 5 preceding residuals and approximations have to be stored for the calculation of the new residual and approximation. The Euclidean norm of the residuals of ORTHORES may oscillate heavily.

PRES20. The pseudo-residual method 20 (PRES20) [23], already described in section 1, results from applying the smoothing algorithm (4) – (6) to the truncated ORTHORES method OR(5). A restart is performed periodically from the smoothed sequence every 20th step. As a result of many tests with very large and sparse matrices the truncation parameter p=5 and the restart parameter 20 have been optimized resulting in a very efficient method with limited storage requirements.

4.2. Residual- and Error-Minimizing Methods.

We discuss methods that minimize the true residual or the true error in the Euclidean norm.

GMRES. If we minimize the true residual according to (18), then we have to determine the coefficients according to (23). A different, more stable implementation based on the Arnoldi process according to (55) was introduced by Saad and Schultz [22]. This method is called GMRES (Generalized Minimal RESidual method). As for ORTHORES truncated or restarted versions have to be used for realistic problems in order to limit the computational resources. We denote the restarted versions that use p preceding residuals and approximations for the calculation of the new residual and approximation by GMRES(p). Note that exact ORTHORES and exact GMRES can be computed in parallel without essential additional costs because the smoothing algorithm (4) – (6) transforms ORTHORES to a method mathematically equivalent to GMRES [34].

GMERR. The generalized minimal error method (GMERR) was introduced in [35]. The method is a generalization of a method proposed by Fridman [10] for symmetric matrices. The method is beyond the here described framework for generalized cg methods. It minimizes the error in the Euclidean norm, though the error never can be calculated. As for GMRES there exist exact, truncated and restarted versions. The latter two have to be applied for large non-symmetric matrices. We denote the restarted versions that use p preceding search directions for the calculation of the new approximation by GMERR(p).

4.3. Biconjugate Gradient-based methods.

The here discussed family of methods is based on the biconjugate gradient method (BCG). BCG itself and CGS produce residuals and errors that oscillate heavily. All other methods (BICO, QMR, TFQMR and BiCGSTAB) try to obtain a smoother convergence.

BCG. The biconjugate gradients (BCG) [17, 6] are based on the Lanczos algorithm, see the special section. BCG can be considered as method applied to a double system, see (24). The algorithm can be formulated as short recurrence. The residuals r_k in the step k can be represented with a matrix polynomial Π_k applied to the starting residual:

$$r_k = \Pi_k(A)r_0 = \left(\sum_{i=1}^{k} \beta_i A^i + I\right) r_0. \tag{69}$$

The algorithm may break down if a division by zero occurs by the calculation of the coefficients $\alpha_{i,k}$ in (25). In this case look-ahead strategies can avoid breakdowns, see the section on the Lanczos process.

BICO. It is known from practice that the convergence behavior of BCG in the Euclidean norm can be very erratic. In order to obtain a better behavior Schönauer [23] applied minimal residual smoothing (4) – (6) to BCG. This algorithm is called BICO. Let r_k^{BCG} be the residual resulting from BCG and r_k^{BICO} the residual resulting from BICO. A direct result of the residual-minimizing smoothing (4) –(6) is that the BICO residual is a function of the iteration index, with a non-increasing norm, i. e.

$$\|r_k^{BICO}\| \le \|r_{k-1}^{BICO}\|, \tag{70}$$

and

$$\|r_k^{BICO}\| \le \|r_k^{BCG}\|. \tag{71}$$

QMR. The quasi-minimal residual method (QMR) was developed by Freund and Nachtigal [9]. It is based on BCG. A look-ahead process for the underlying BCG is proposed in order to avoid breakdowns. Then a quasi-residual minimizing technique is applied resulting in far less oscillating residuals. Zhou and Walker [41] show that quasi-residual smoothing transforms BCG into QMR.

CGS. The conjugate gradient squared method (CGS) was introduced by Sonneveld [31]. The idea is to square the BCG matrix-polynomial Π_k, see equation (69), in order to get a faster convergence. It is obvious that the CGS residuals oscillate at least as heavily as the BCG residuals. We get

$$r_k = \Pi_k^2(A) r_0. \tag{72}$$

The calculation of the shadow sequence r_k^*, see (24), is not neccessary. Therefore, CGS has the advantage that no matrix-vector multiplication by A^T is needed. Thus it is suited for problems, where the matrix A and therefore A^T are not available.

TFQMR. The transpose-free quasi minimal residual method (TFQMR) [7], was introduced by Freund. The idea is to apply a quasi-residual minimizing technique as for QMR to a method that does not need a multiplication by the transposed matrix. Zhou and Walker [41] show that quasi-residual smoothing transforms an auxiliary sequence based on CGS into TFQMR.

BiCGSTAB. The biconjugate gradient stabilized method (BiCGSTAB) was introduced by van der Vorst [32]. The idea is to substitute the squared matrix polynomial Π_k^2 of CGS, see (72), by the product of matrix polynomials $\Psi_k \Pi_k$ in order to get a smoother convergence and to stabilize the iteration. Thus we get

$$r_k = \Psi_k(A)\Pi_k(A) r_0. \tag{73}$$

Ψ_k is recursively defined by

$$\Psi_k(A) = (\gamma_k A + I)\Psi_{k-1}(A), \tag{74}$$

where γ_k is determined from a 1-dimensional minimization

$$\|r_k\| = \min_{\gamma_k} \|(\gamma_k A + I)\Psi_{k-1}(A)\Pi_k(A) r_0\|. \tag{75}$$

The behavior of BiCGSTAB has been visualized very impressively by Fujino, Zhang and Mori [11]. Gutknecht [13], Sleijpen and Fokkema [29] and Sleijpen, van der Vorst and Fokkema [30] generalized BiCGSTAB so that Ψ_k is determined by an ℓ-dimensional minimization. These methods are called BiCGSTAB(ℓ).

4.4. Methods based on the normal equations.

Instead of solving the linear system (1) the two equivalent systems

$$A^T A x = A^T b, \tag{76}$$

$$A A^T y = b \tag{77}$$

can be considered. These systems are called normal equations of the first kind (76) and of the second kind (77). The solution of (1) is obtained from (77) by $x = A^T y$. As for both systems the system matrix is symmetric classical CG can be applied. The resulting methods can be reformulated in terms of the original system, i. e. with the residuals and approximations of (1).

The speed of convergence may be very slow if the eigenvalues of the iteration matrix $A^T A$, AA^T, respectively, are more scattered than those of the original system matrix A. For symmetric matrices the condition number is squared, i. e. the convergence is slow, but in the general case this is not true. Theoretically the methods converge always. Thus they are very robust but not very efficient.

CGNE. The classical CG method applied to the normal equations of the second kind (77) can be reformulated so that only terms of the original system are used. The method was introduced by Craig [5] and can be implemented by the equations (10) – (15) with $d_k = A^T r_k$. The method minimizes the errors of the original system (1) in the Euclidean norm, though the errors never never can be calculated. Therefore, it is called CGNE (CG Normal equations Error-minimizing).

CGNR-ATPRES. If we apply the classical CG method to the normal equations of the first kind (76), then we obtain CGNR. The method minimizes the residuals of the original system (1) in the Euclidean norm. Therefore, it is called CGNR (CG Normal equations Residual-minimizing). The method can be realized by applying the smoothing algorithm, equations (4) – (6), to CGNE. This implementation is called ATPRES [25].

4.5. Others.

R3-CKS. The rank-3 update conjugate Krylov subspace method (R3-CKS) was proposed in [36]. The method fulfils a minimization property and can be formulated with a short recurrence. The method is beyond the CG frame and can be realized by generalizing equation (16) to

$$r_k^T Z_k r_{k-i+1} = 0, \tag{78}$$

with $Z_k = Z + D_k E_k^T$. $Z_k \in \mathbb{R}^{n\times n}$ is a rank-3 update of $Z \in \mathbb{R}^{n\times n}$ with $D_k, E_k \in \mathbb{R}^{n\times 3}$. In each iteration step two matrix-vector multiplications with the matrices A, A^T, respectively, have to be performed. The work counted in matrix-vector multiplications as essential operations corresponds therefore to the work of the biconjugate gradients.

4.6. Work and Storage.

The essential operations for the here discussed methods are matrix-vector multiplications, dot products and triadic operations. For realistic problems the work for the matrix-vector multiplication dominates the work for other operations. The required storage is depending on the recurrence length and is a crucial point for large systems, i. e. the value of p in (11) may not be arbitrary large. In table 1 the different methods are classified with respect to work and storage requirements. The

method	mvm	st
classical CG	1	5
OR(p)	1	2p+1
PRES20	1	13
GMRES(p)	1	2p+3
GMERR(p)	1	2p+3
CGNR-ATPRES	2	8
CGNE	2	6

method	mvm	st
BCG	2	6
BICO	2	8
QMR	2	8
CGS	2	6
TFQMR	2	8
BiCGSTAB	2	6
R3-CKS	2	12

mvm = matrix-vector multiplications per iteration, st = number of auxiliary vectors of length n

TABLE 1
Comparison of work and storage

later investigated truncated and restarted methods, OR(5), PRES20, GMRES(5) and GMERR(5), require approximately the same amount of storage and work.

4.7. A Sample Problem.

All here presented methods will be analysed by numerical experiments for a sample problem. Clearly, we cannot judge the different methods just by applying them to one particular example. But we get a feeling how the methods work.

We will solve the following partial differential model equation (PDE) for the velocity

$$\begin{aligned} u_{xx} + u_{yy} + u_{zz} + u + R(uu_x + vu_y + wu_z) - h_1 = 0 \ , \\ v_{xx} + v_{yy} + v_{zz} + v + R(uv_x + vv_y + wv_z) - h_2 = 0 \ , \\ w_{xx} + w_{yy} + w_{zz} + w + R(uw_x + vw_y + ww_z) - h_3 = 0 \ . \end{aligned} \tag{79}$$

The h_i are determined that the solution is

$$u = \sin\alpha \ \cos\beta \ cos\gamma \ , \ v = \cos\alpha \ \sin\beta \ cos\gamma \ , \ w = \cos\alpha \ \cos\beta \ sin\gamma \tag{80}$$

with $\alpha = 2\pi x$, $\beta = 2\pi y$, $\gamma = 2\pi z$. We prescribe Dirichlet boundary conditions, i.e. u, v, w on the unit cube. This example is a coarse model of the Navier-Stokes equations. The parameter R has the character of a Reynolds number. For small R we have strong ellipticity, for large R we have a more hyperbolic character. Thus we can change the type of problem by the value of R.

The linear system arises from a finite difference discretization by the FIDISOL program package [23] with consistency order 4 and from the linearization in the first Newton step. As starting solution for the Newton method the boundary conditions have been interpolated by a weighted 1-D interpolation for the whole domain, see [23]. The calculations have been performed on a $20 \times 20 \times 20$ grid for different Reynolds numbers. The matrix is already normalized according to equation (2), the dimension of the system is n=24000.

We have tested the methods that have been discussed in the previous section for the Reynolds numbers R=1, 10, 100, 500 and 10^5. The methods are grouped together according to

I Classical CG (denoted by CG), GMRES(5), GMERR(5), OR(5), PRES20, R3-CKS,

II BCG-based methods:
BCG, BICO, QMR, CGS, TFQMR, BiCGSTAB (denoted by CGSTAB),

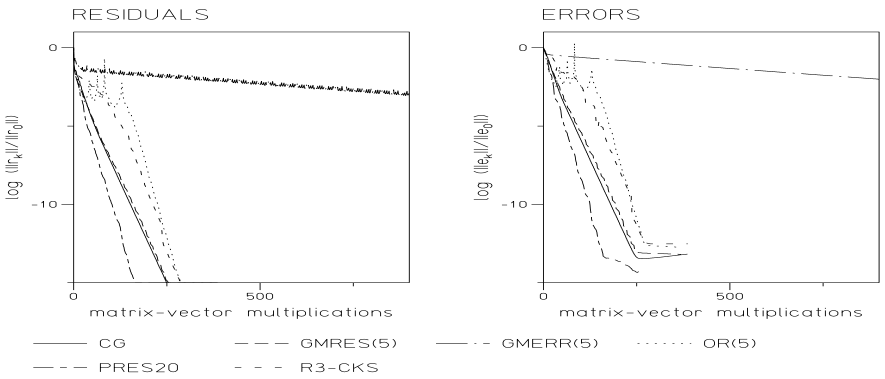

FIG. 1. *Methods of group I for $R = 1$*

III Methods based on the normal equations:
CGNE, CGNR-ATPRES.

In the figures the logarithm of the relative norms of the residuals and errors is plotted against the number of matrix-vector multiplications (that are the essential operations).

Group I. OR(5), PRES20, GMRES(5) and GMERR(5) require approximately the same amount of storage and work per iteration step, see table 1. Therefore they are grouped together. The GMERR(5) method performs poorly for all Reynolds numbers, see figures 1–5. For $R = 1$ and $R = 10$ PRES20 is the best method, GMRES(5), R3-CKS and surprisingly the classical CG, that was introduced for symmetric and positive definite matrices, perform well. For $R = 10$ OR(5) diverges heavily oscillating, but for $R = 100$ it is the best method, smoothly converging. For this Reynolds number PRES20 and GMRES(5) are comparable good, while classical CG becomes slower. For the Reynolds numbers $R = 500$ and $R = 10^5$ all methods fail.

Group II. The methods of group II have the same origin: BCG. The figures 6–10 show that they perform quite similar. BCG and CGS oscillate heavily, but the reduction of the residuals up to 10 orders of magnitude is not essentially worse than that of the more smooth variants BICO, QMR, TFQMR and BiCGSTAB. For the Reynolds number $R = 500$ CGS and TFQMR do not converge, while the CGS-based BiCGSTAB still performs well. For $R = 10^5$ all methods fail. For the Reynolds numbers R=1, 10 and 100 BiCGSTAB is the best method. It is even better than all methods of group I.

Group III. CGNE and CGNR-ATPRES perform quite similar, see figures 11–15. The figures show clearly that CGNR-ATPRES minimizes the residuals. Its residual norm is always better than that of CGNE. As CGNE minimizes the errors it is always better in the error norm. Both methods converge for low Reynolds numbers rather slowly compared to the best methods of group I and II. But for a Reynolds number of 10^5 they are the only methods of all tested techniques that converge at all. Thus they are rather robust. Note that the relative reduction of the residual is for $R = 10^5$ of 2 orders of magnitude better than the reduction of the error.

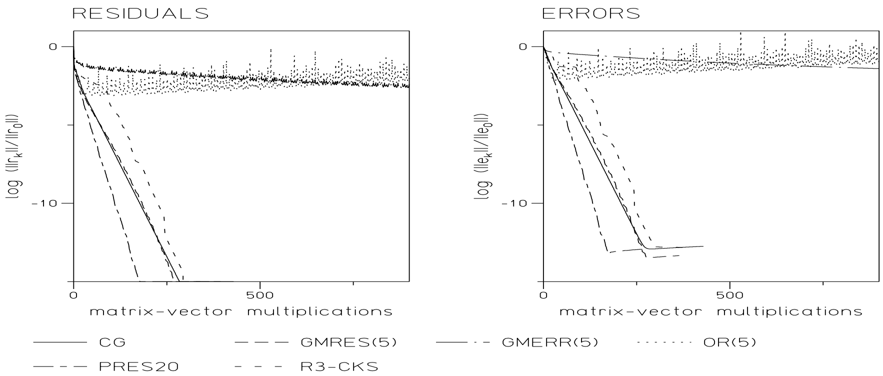

FIG. 2. *Methods of group I for* $R = 10$

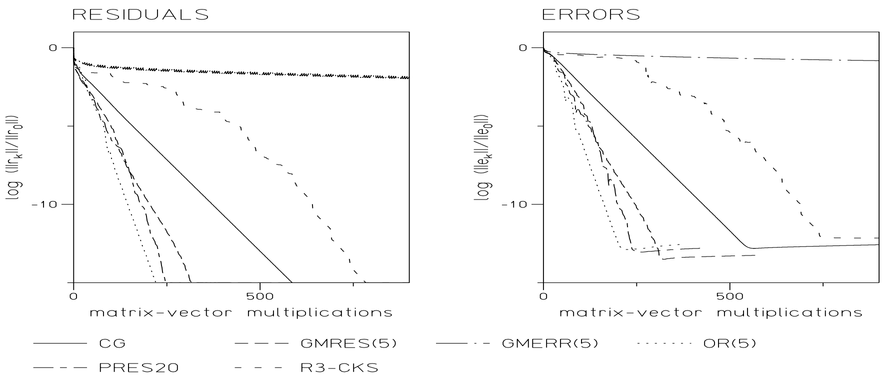

FIG. 3. *Methods of group I for* $R = 100$

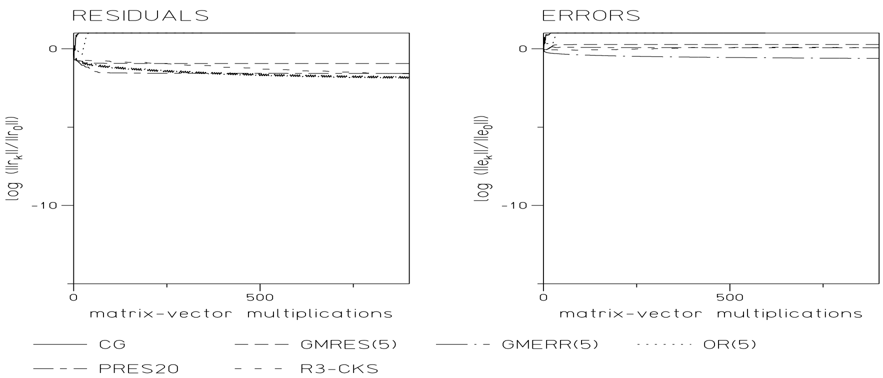

FIG. 4. *Methods of group I for* $R = 500$

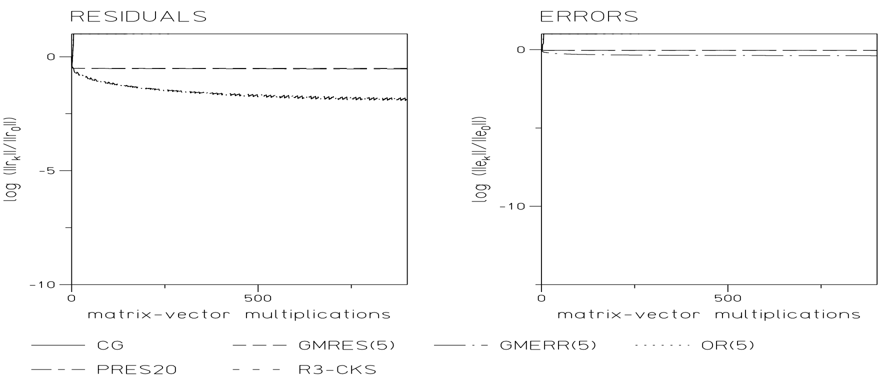

FIG. 5. *Methods of group I for* $R = 10^5$

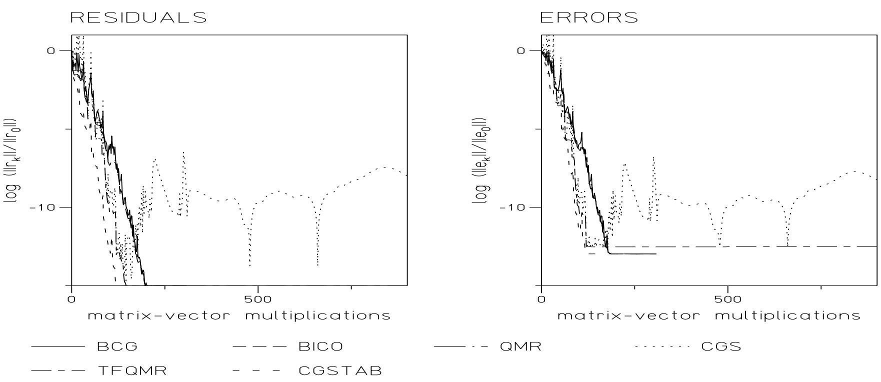

FIG. 6. *Methods of group II for* $R = 1$

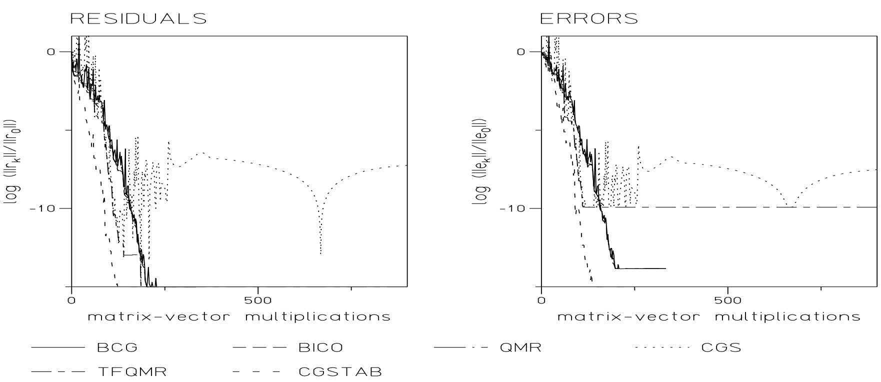

FIG. 7. *Methods of group II for* $R = 10$

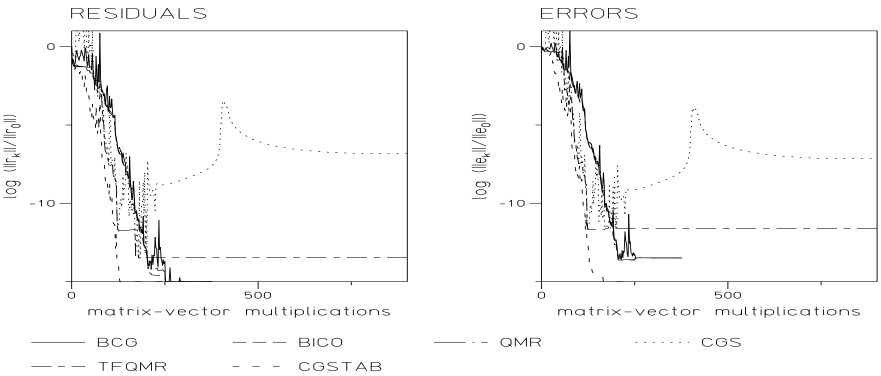

Fig. 8. *Methods of group II for* $R = 100$

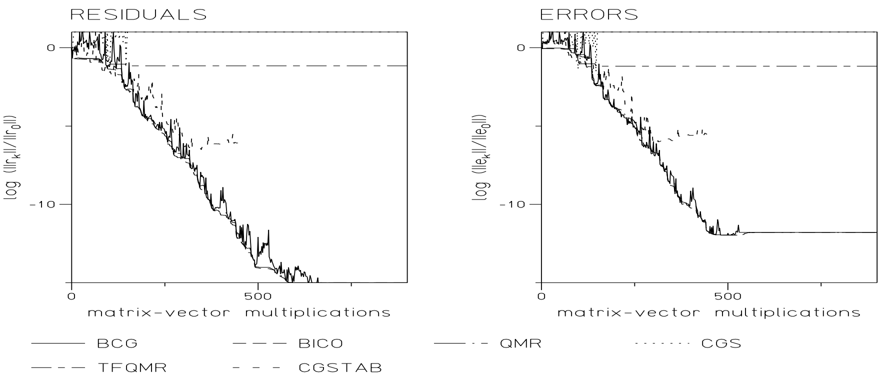

Fig. 9. *Methods of group II for* $R = 500$

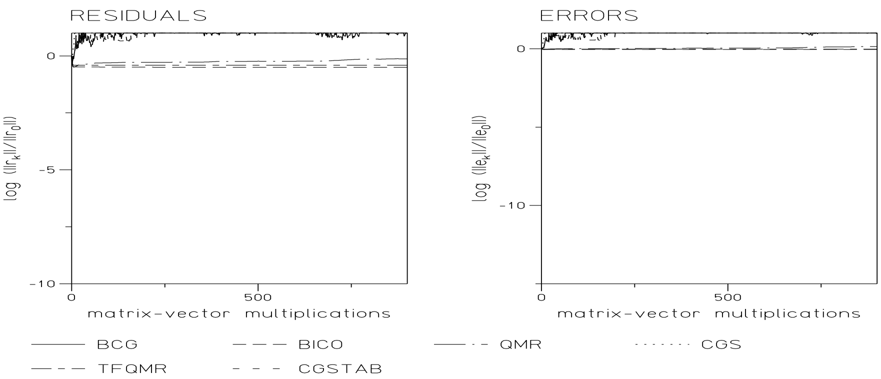

Fig. 10. *Methods of group II for* $R = 10^5$

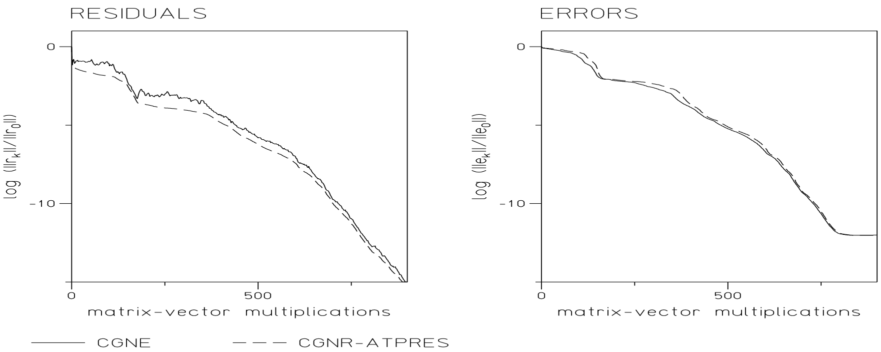

Fig. 11. *Methods of group III for* $R = 1$

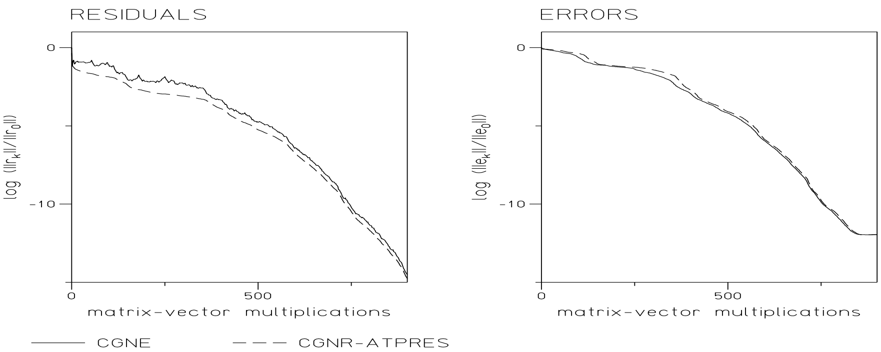

Fig. 12. *Methods of group III for* $R = 10$

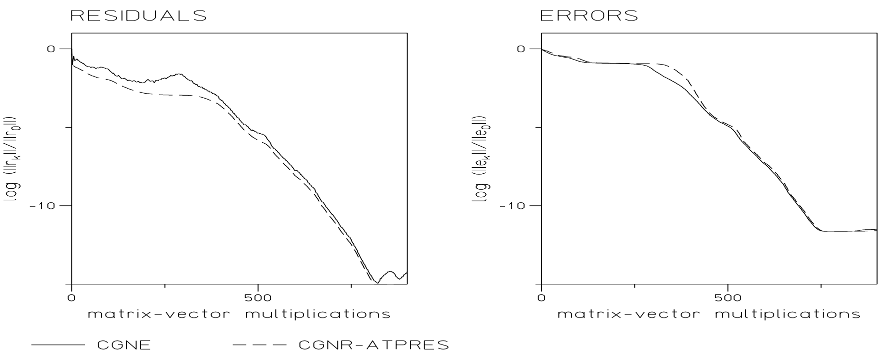

Fig. 13. *Methods of group III for* $R = 100$

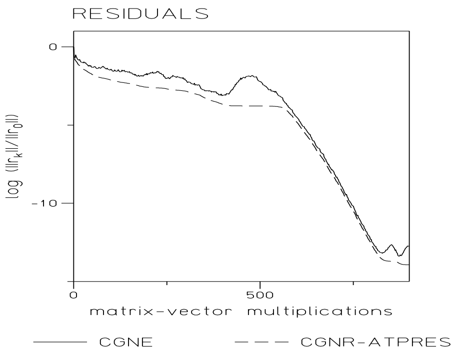

FIG. 14. *Methods of group III for* $R = 500$

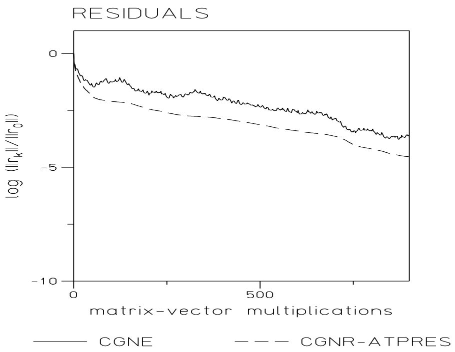

FIG. 15. *Methods of group III for* $R = 10^5$

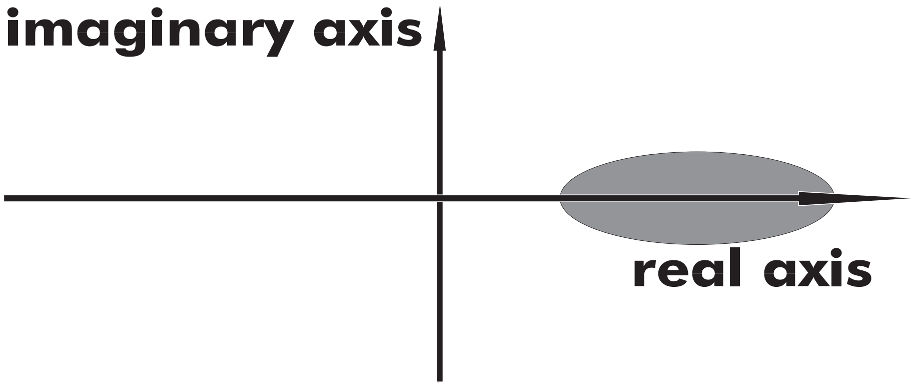

FIG. 16. *Domain of eigenvalues, where restarted or truncated ORTHORES and GMRES methods are efficient*

The eigenvalues of the matrix on a coarser grid were computed in [35]. They are enclosed by the unit circle as result of the normalization (2). For a small R the eigenvalues are close to the real axis and are in the positive half plane. For increasing R the imaginary part of the eigenvalues increases. Up to a certain R the real part of the eigenvalues is positive. For a large R the eigenvalues are scattered in the whole complex domain.

Note that the matrix of our example was normalized according to equation (2). Thus the system matrix cannot be arbitrarily far from a normal matrix. If the matrix A is not too far from a normal matrix, then we can derive from practical experience the following rules of thumb. ORTHORES(p) and GMRES(p) are similar methods and converge for small values of p if the eigenvalues of A are close to the real axis, see figure 16. A special optimized version for this case is PRES20. BCG or the smoothed versions BICO and QMR converge if the eigenvalues are in the positive half plane, see figure 17. Similar results hold for CGS and TFQMR. Then they are superior to ORTHORES and GMRES for small values of p. If we admit high values of p, then BCG, BICO, QMR, CGS or TFQMR lose. The BiCGSTAB versions behave similar, but still perform very well when ORTHORES and GMRES are better than BCG, BICO, QMR, CGS or TFQMR. CGNR-ATPRES and CGNE are the best methods for scattered eigenvalues, however they are rather slow for systems where the eigenvalues are close to the real axis, see figure 18.

Note that ORTHORES and GMRES methods can be computed in parallel without additional costs by smoothing, see [34], and that GMRES minimizes the residuals in the Euclidean norm while ORTHORES minimizes the error in the energy norm. GMRES guarantees monotonicly decreasing residuals. ORTHORES may be better as regards the decrease of the error. For both methods the convergence is only guaranteed if they are used as exact methods. Truncated and restarted versions are very efficient for matrices A with a single cluster of eigenvalues, but not very robust.

ORTHORES, GMRES and GMERR use long recurrences for large values of p. PRES20, R3-CKS, the BCG-based methods, BCG, BICO, QMR, CGS, TFQMR and BiCGSTAB, and CGNE and CGNR-ATPRES use short recurrences.

GMERR and R3-CKS cannot be classified up to now. For the here presented example they do not perform well. There are other examples where they work fine.

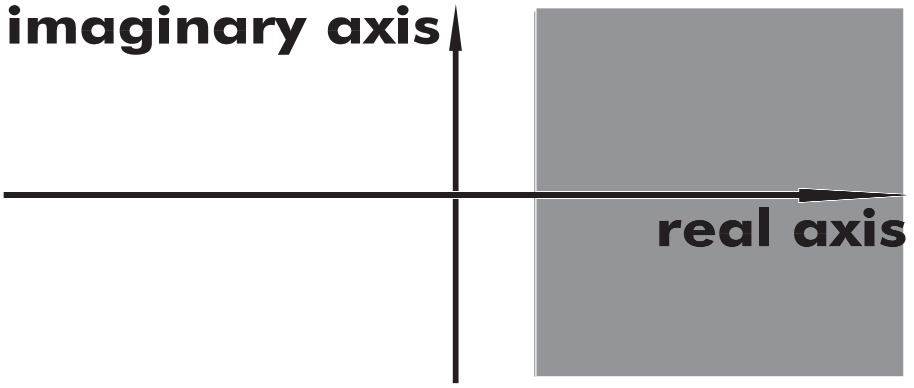

FIG. 17. *Domain of eigenvalues, where methods resulting from BCG are efficient*

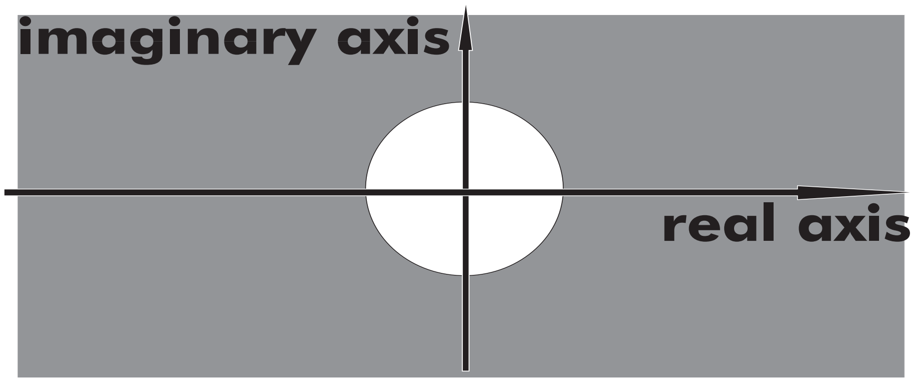

FIG. 18. *Domain of eigenvalues, where methods based on the normal equations have to be applied*

5. Some implementation details.

Polyalgorithm: If we use an iterative linear solver within a black box solver for PDEs, e.g. our LINSOL in FIDISOL/CADSOL [23, 24] or in VECFEM [12], we must make a compromise between efficiency and robustness. Therefore we have designed LINSOL as a polyalgorithm with automatic method switching [23, 27]: We start with PRES20, if it does not converge "sufficiently well" we switch to BICO and, if this fails also, to ATPRES. At a possible breakdown of BICO we make a restart from the smoothed sequence.

The algorithm is a typical "numerical engineering" algorithm that makes use of experience: The polyalgorithm starts with PRES20 and controls at the restart point after every 20 iterations if the L_2-norm $||r_k||_2$ has decreased by a factor of 0.5 since the last restart. If this is not the case, we switch to BICO. Then we check after every 1000 iteration steps if the residual $||r_k||_2$ has decreased at least by a factor of 0.1 since the last check. If this does not hold, we switch to the "emergency exit" ATPRES. The reason why we check only after 1000 iterations is that BICO often converges "in steps" where it falls "down the cliff" after a plateau phase. This strategy means that even in a single Newton step the iteration method of the linear solver may be changed.

In order to demonstrate how this polyalgorithm works we execute it for the solution of the model PDE (79) with solution (80). We prescribe Dirichlet boundary conditions, i.e. u, v, w on the unit cube. We generate A' and b' of (2) by FIDISOL for the consistency orders 4 in x, y, z-direction on a grid 40 x 40 x 40 ($n = 192\ 000$). The initial values for u, v, w are mean values of interpolations between the bondary values in x, y, z-direction. For these values the matrix A' of the first Newton step is used as test-matrix. We force LINSOL to reduce the L_2-norm of the relative residual $||r_k||/||r_0||$ of the normalized equations (1) to be reduced to 10^{-20}. In Fig. 19 we depict the log of the relative residual as a function of the MVMs (matrix-vector multiplications). Note that PRES20 needs one MVM, but BICO and ATPRES need two MVMs per iteration step. The MVMs are a good measure for the computation. The parameter R is indicated as r on the top of the figures. For $R = 1$ and $R = 10$ the polyalgorithm follows PRES20, for $R = 100$ the polyalgorithm is between PRES20 and BICO, for $R = 1000$ the polyalgorithm follows BICO and for $R = 10000$ it follows nearly ATPRES. Thus the polyalgorithm with the above mentioned strategy follows nearly the best method. This is what we can call making "intelligent" software by numerical engineering.

Parallelization: The kernel operation of all these methods are matrix-vector multiplications (MVMs) with A or A^T. We use storing of the sparse matrices by (packed) diagonals or rows or columns or even group of single elements. The data structures are selected for optimal vectorization. For the parallelization row blocks are distributed over the processors [39]. The row blocks are subdivided into "logical" column blocks. Full parallelization with maximal latency hiding has been obtained. The remaining operations are scalar products (cascade sum for parallel computer) and vector triads. Thus CG-type methods can be fully vectorized and parallelized.

It should be mentioned that CG-type methods are very sensitive to implementation details. Different implementations of the same method may have quite different convergence properties due to the sensibility of the algorithms to rounding errors.

6. Preconditioning.

If the basic method fails we know that the eigenvalues of A are not sufficiently clustered in the complex plane. Then we can try to precondition the linear system

FIG. 19. *Reduction of the relative residual for different algorithms over the number of matrix-vector multiplications (MVM)*

.

either "from the left" by premultiplication by a preconditioning matrix P_k:

$$P_k A x = P_k b \ , \tag{81}$$

or by choosing the search direction

$$d_k = P_k r_k \ , \ then \ in \ (11) \ A d_k = A P_k r_k \ , \tag{82}$$

which is called preconditioning "from the right". This corresponds to a change in the unknowns from x to y in the following way:

$$A x = b \ , \ A P_k y = b \ , \ x = P_k y \ , \tag{83}$$

so that AP_k is the iteration matrix. In both cases $P_k A_k = I$, $AP_k = I$ would yield the solution in one step, thus $P_k = A^{-1}$ would be the optimal preconditioning matrix. Quite naturally A^{-1} is not available. However, all preconditioning techniques aim at some "substitute" for A^{-1} that is "practicable" and clusters the eigenvalues of the iteration matrix without too high expenses.

One possible method is ILU (incomplete LU decomposition) with many variants, but unfortunately totally recursive and not suited for vectorization and parallelization. With $P_k = A^{-1}$, $A = LU$ in (82) we had in (11)

$$A d_k = A A^{-1} r_k = A(U^{-1}(L^{-1} r_k)) \tag{84}$$

which means elimination $v = L^{-1} r_k$, substitution $w = U^{-1} v$, MVM Aw. Up to now no sufficiently robust, efficient, vectorizable and parallelizable preconditioning is known. This is due to the recursive character of ILU which was not fatal for scalar computers, but is fatal for vector or pipelined arithmetic and for parallel computation, especially for sparse matrices. It is misleading to count the number of iterations which may be reduced by ILU preconditioning. Unfortunately, the computation time may nevertheless increase significantly, depending on the architecture of the used (super)computer. Corresponding examples are reported in [26]. There are many modifications of ILU, e.g. threshold ILU, where elements in the LU process are dropped only if they are below a certain threshold value. Thus a certain fill-in is admitted [21].

Another possibility is to use as P_k an "approximate inverse", i.e. some sparse approximation for the full matrix A^{-1}. This procedure turns out to be not robust if the "pattern" of nonzero elements is not optimally selected. An example for an incomplete inverse is to prescribe a pattern for P_k, e.g. that of A, and to determine P_k as Frobenius inverse of A [4].

7. Beyond CG-type methods.

Many of the here dicussed methods have been already beyond the CG-type. The reason for the large variety in iterative solvers is that methods should converge rapidly. Importantly, there is no single optimal method, but each method performs differently, depending on the matrix property, on the used computer architecture and on the storage pattern of the matrix. Moreover, a theoretically under certain circumstances fast method like exact GMRES cannot be used for large and sparse systems because the computing time and the required storage increase with the iteration. The proper goal for the devopment of new methods is to obtain more efficient and robust techniques. As there is no recipe the CG approach has been generalized in different ways. For QMR for example the minimization property is varied step by step in order to obtain a minimization property at all and to get the possibility to formulate the method with a short recurrence. This approach results in step depending orthogonalities. For BiCGSTAB the search directions are changed in order to obtain a faster and smoother recurrence.

A very general approach is to consider methods that calculate the approximations according to

$$x_{k+1} = \sum_{i=1}^{p} \varphi_{i,k} d_{k-i+1} + \sum_{i=1}^{p} \alpha_{i,k} x_{k-i+1} \tag{85}$$

instead of (14) with $\alpha_{i,k}$ fixed, and by determining the coefficients $\varphi_{i,k}$ by orthogonality relations

$$r_{k+1}^T Z_k d_{k-i+1} = 0 \tag{86}$$

instead of (16), with arbitrary non-singular matrices Z_k. These methods are called orthogonalization methods [37]. We get (14) as special case by setting d_k as in (14), $d_{k-i+1} = x_{k-i+1} - x_{k-i}$ for i = 1, ..., p-1, $\sum_{i=1}^{p} \alpha_{i,k} x_{k-i+1} = x_{k-p+1}$ and for example the orthogonalities (21) and (22) by $Z_k = A$. By this approach the behavior of methods that are beyond the CG concept can be explained. For example the step-depending matrices Z_k offer the possibility to describe the step-depending orthogonalization property of QMR [9]. GMERR [35] can be explained by using particular search directions d_{k-i+1}, to name only a few methods. By this approach well-known methods can be classified and convergence results can be obtained [37], but, more importantly, it is so general that new techniques can be incorporated and the the convergence results can be applied.

8. Concluding remarks.

It is a long way from the elementary method of steepest descent to the sophisticated generalized conjugate gradient methods and beyond to generalized orthogonalization methods. It was not the intention (and it would not be possible in the available space) to give an exhaustive survey over these methods. The prime intention was to point out to the reader a way to "understand" the design of these methods and to stimulate and encourage him/her to fertilize this area with new and unconventional ideas. It is only now that we begin to discover the relations between many of the discussed methods, e.g. that a PRES method with smoothing is a GMRES method, – theoretically, but practically it may behave quite differently by the influence of rounding errors.

We have seen in section 3 that there is no "universal" method. This has led us in section 4 to the design of a polyalgorithm. This situation means that generalized CG methods need much "experience" in their application and a lot of "numerical engineering". Because these methods are used for the solution of extremely large problems on modern vector and parallel computers, there is a significant influence of the computer architecture on the efficiency of the method, e.g. on all types of recursive preconditioners, – again numerical engineering.

It is just this situation that makes the subject so fascinating. What we want is an iterative method that finds from the starting solution x_0 the "cheapest" way to the solution x. This is then the most efficient method. But is this method also "robust" in that sense that it is also "cheap" for a quite different type of problem or on a quite different computer architecture? Or, what is then the "cheapest" method? These questions demonstrate that this area is open to interesting future research, in the domain of theoretical investigations as well as in the domain of practical numerical engineering. Often the latter has been the pathfinder for new and better methods.

In the following paper of this Special Issue R. Weiss will give some details of the theory of generalized CG methods.

REFERENCES

[1] C. Brezinski, M. Redivo-Zaglia, and H. Sadok. Avoiding breakdown and near-breakdown in Lanczos type algorithms. *Numerical Algorithms*, 1(3):261–284, 1991.

[2] C. Brezinski, M. Redivo-Zaglia, and H. Sadok. A breakdown-free Lanczos type algorithm for solving linear systems. *Numer. Math.*, 63:29–38, 1992.

[3] C. Brezinski and H. Sadok. Avoiding breakdown in the CGS algorithm. *Numerical Algorithms*, 1(2):199–206, 1991.

[4] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. Research Report UMSI 94/101, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, 1994.

[5] E.J. Craig. The n-step iteration procedures. *Math. Phys.* , 34:64–73, 1955.

[6] R. Fletcher. Conjugate gradient methods for indefinite systems. In G.A. Watson, editor, *Proc. Dundee Biennial Conf. on Num. Anal.*, pages 73–89. Springer-Verlag, 1975.

[7] R.W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2):470–482, March 1993.

[8] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. An implementation of the Look-Ahead Lanczos algorithm for non-Hermitian matrices, part 1. RIACS Technical Report 90/45, Research Institute for Advanced Computer Science, NASA Ames Research Center, November 1990.

[9] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[10] V.M. Fridman. The method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetrical matrix. *USSR Comput. Math. and Math. Phys.* , 2:362–363, 1963.

[11] S. Fujino, S. Zhang, and M. Mori. Visualization of convergence behavior of Bi-CGSTAB method. *Supercomputer*, 8(6):127–135, 1992.

[12] L. Groß, C. Roll, and W. Schönauer. Chances and limits for the application of the FEM-solver VECFEM to general partial differential equations. In R. Vichnevetsky, Knight, and Richter, editors, *Advances in Computer Methods for Partial Differential Equations VII*, pages 306–312. IMACS, 1992.

[13] M.H. Gutknecht. Variants of BICGSTAB for matrices with complex spectrum. Research Report 91-14, Eidgenössische Technische Hochschule Zürich, Interdisziplinäres Projektzentrum für Supercomputing, ETH-Zentrum, CH-8092 Zürich, August 1991.

[14] M.H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, part 1. *SIAM J. Matrix. Anal.*, 13(2):594–639, April 1992.

[15] L.A. Hageman and D.M. Young. *Applied Iterative Methods.* Academic Press, New York, London, Toronto, Sydney, San Francisco, 1981.

[16] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–435, 1952.

[17] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45:255–282, 1950.

[18] B.N. Parlett, D. Taylor, and Z. S. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[19] M. Rozložník. Variants of the generalized minimal error method. In *Proceedings of Mathematical Modelling and Computational Methods "Modelling 94", Prague*, 1994. submitted.

[20] M. Rozložník and Z. Strakoš. Variants of the Krylov space methods. Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží2, 18207 Prague 8, Czech Republic, March 1994.

[21] Y. Saad. ILUT: A dual threshold incomplete LU factorization. Research Report UMSI 92/38, University of Minnesota, Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, Minnesota 55415, March 1992.

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

[23] W. Schönauer. *Scientific Computing on Vector Computers.* North-Holland, Amsterdam, New

York, Oxford, Tokyo, 1987.

[24] W. Schönauer, H. Müller, and E. Schnepf. Pseudo-residual type methods for the iterative solution of large linear systems on vector computers. In M. Feilmeier, J. Joubert, and U. Schendel, editors, *Parallel Computing 85*, pages 193–198. Elsevier Science Publishers B.V., North-Holland, 1986.

[25] W. Schönauer, M. Schlichte, and R. Weiss. Wrong ways and promising ways towards robust and efficient iterative linear solvers. In *Advances in Computer Methods for Partial Differential Equations VI*, pages 7–14. Publ. IMACS, 1987.

[26] W. Schönauer, E. Schnepf, and H. Müller. The FIDISOL program package. Internal report 27/85, University of Karlsruhe, Computing Center, 1985.

[27] W. Schönauer, R. Weiss, and P. Sternecker. Polyalgorithms with automatic method selection for the iterative solution of linear equations and eigenproblems. In P.W. Gaffney and E.N. Houstis, editors, *Programming Environments for High-Level Problem Solving*, pages 57–67. North-Holland, Amsterdam, 1992.

[28] J. R. Shewchuk. Conjugate gradient method without the agonizing pain. Internal Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1994.

[29] G.L.G. Sleijpen and D.R. Fokkema. BiCGStab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numer. Anal. (ETNA)*, 1:11–32, September 1993.

[30] G.L.G. Sleijpen, H.A. van der Vorst, and D.R. Fokkema. BiCGSTAB(l) and other hybrid Bi-CG methods. *Numerical Algorithms*, 1993. to appear.

[31] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 10(1):36–52, 1989.

[32] H.A. van der Vorst and P. Sonneveld. CGSTAB: A more smoothly converging variant of CGS. Report, Delft University of Technology, Department of Mathematics and Informatics, 1990.

[33] H.F. Walker. Implementations of the GMRES method. *Computer Physics Communications*, 53:311–320, 1989.

[34] R. Weiss. Convergence behavior of generalized conjugate gradient methods. Internal report 43/90, University of Karlsruhe, Computing Center, 1990.

[35] R. Weiss. Error-minimizing Krylov subspace methods. *SIAM J. Sci. Comput.*, 15(3):511–527, 1994.

[36] R. Weiss. Minimization properties and short recurrences for Krylov subspace methods. *Electronic Transactions on Numerical Analysis (ETNA)*, 2:57–75, 1994.

[37] R. Weiss. Orthogonalization methods. Internal report 52/94, University of Karlsruhe, Computing Center, 1994.

[38] R. Weiss. Relations between smoothing and QMR. *Numer. Math.*, 1994. submitted to.

[39] R. Weiss, C. Roll, H. Häfner, L. Groß, M. Schmauder, and W. Schönauer. Parallelisierung von ODIN-Software für iterative Gleichungslöser. In A. Schreiner and E. Schnepf, editors, *Tagungsband zum 3. ODIN-Symposium Karlsruhe: Softwareentwicklung für Supercomputer*, pages 117–128, 1994.

[40] D.M. Young. *Iterative Solution of Large linear Systems.* Academic Press, New York, San Francisco, London, 1971.

[41] L. Zhou and H.F. Walker. Residual smoothing techniques for iterative methods. *SIAM J. Sci. Comput.*, 15(2):297–312, 1994.