Computational variants of the CGS and BiCGstab methods*

Victor Eijkhout
University of Tennessee
Department of Computer Science
107 Ayres Hall, Knoxville, TN 37996-1301
eijkhout@cs.utk.edu

August 4, 1994

Abstract
The CGS and BiCGstab algorithms were derived by their authors from the bi-conjugate gradient iteration. In this paper a derivation from the (unnormalized) Lanczos algorithm is given. Experimental results are given, showing that these methods offer no immediate advantage over their earlier counterparts.

1 Introduction

A number of recent conjugate gradient-type methods are based on the idea of polynomial multiplication. Ordinary cg-like methods generate a sequence of polynomials $\{P_i\}$ of successively higher degree, and they compute residuals that satisfy $r_i = P_i(A)r_1$ where r_1 is the initial residual. Polynomial multiplication methods generate a second sequence of polynomials, $\{Q_i\}$, and they compute residuals satisfying $r_i = Q_i(A)P_i(A)r_1$.

The first such method was CGS [3], conjugate gradients squared, which is based on $Q_i \equiv P_i$. This method's motivation comes from the bi-conjugate gradient or Lanczos algorithm, where two sequences $s_i = P_i(A^t)r_1$ and $r_i = P_i(A)r_1$ are computed. Since in the ideal case both sequences converge to zero, and

$$r_1^t P_i^2(A) r_i = \big(P_i(A^t)r_1\big)^2 \big(P_i(A)r_1\big) = s_i^t r_i,$$

it makes sense to try and compute the sequence $\hat{r}_i = P_i^2(A)r_1$.

In practice, CGS inherits, and even amplifies, the irregularities in the convergence of the bi-conjugate gradient method. Therefore in [4] a method was proposed that smooths the convergence behaviour by letting $\hat{r}_i = Q_i(A)P_i(A)r_1$, where the P_i polynomials are still derived from the biconjugate gradient method, but the Q_is are derived from a sequence of steepest descent steps.

*. This work was supported by DARPA under contract number DAAL03-91-C-0047

In the above, the rationale for computing polynomial multiplying sequences is given. The rest of this paper is devoted to explaining why it is computationally feasible to construct such sequences. In particular, a construction will be given that is not based on the bi-conjugate gradient method (as was the case in [3] and [4]), but on the three-term recurrences of the Lanczos method.

2 Standard presentation of conjugate gradient squaring methods

Denote in preconditioned bi-conjugate gradient-type methods left and right residuals by s_i, r_i and left and right search directions by q_i, p_i, and let M be the preconditioning matrix.

For the derivation of a preconditioned conjugate gradients squared method we use a slightly different version of the (bi)conjugate gradient method than is used ordinarily. The residuals and search directions are computed as

$$r_{i+1} = r_i - AMp_i\alpha_i^{(r)}, \qquad p_{i+1} = r_{i+1} + p_i\beta_i,$$

and the left sequences of residuals and search directions are computed as

$$s_{i+1} = s_i - A^tM^tq_i\alpha_i^{\ell}, \qquad q_{i+1} = s_{i+1} + q_i\beta_i,$$

where α_i^{ℓ} and $\alpha_i^{(r)}$ maybe different for the left and right sequences. The coefficients for the right sequence are computed in the traditional manner

$$\alpha_i^{(r)} = s_i^tr_i/q_i^tAp_i, \qquad \beta_i = s_{i+1}^tr_{i+1}/s_i^tr_i.$$

There exist polynomials ρ_i and π_i of degree $i-1$ such that

$$r_i = \rho_i(AM)r_1, \qquad p_i = \pi_i(AM)r_1.$$

We find relations for the polynomials

$$\rho_{i+1} = \rho_i - AM\pi_i(AM)\alpha_i, \qquad \pi_{i+1} = \rho_{i+1} + \pi_i\beta_i.$$

Similarly the left sequences satisfy

$$s_i = \tilde{\rho}_i(A^tM^t)s_1, \qquad q_i = \tilde{\pi}_i(A^tM^t)s_1$$

for certain polynomials $\tilde{\rho}_i$ and $\tilde{\pi}_i$.

In the following derivation of the cgs and bcgs methods, the polynomials ρ_i and π_i will be taken to refer to the left sequences s_i and q_i. For cgs they are identical to the polynomials for the r_i and p_i sequences.

The conjugate gradient squared method aims at computing the sequences of squared residuals $\rho_i^2(AM)r_1$ and squared search directions $\pi_i^2(MA)p_1$. The scalars of the biconjugate gradient method can be computed as

$$s_i^tMr_i = s_1^t\rho_i(MA)M\rho_i(AM)r_1 = s_1^tM\left[\rho_i^2(AM)r_1\right],$$

and

$$q_i^tAp_i = q_1^t\pi_i(AM)A\pi_i(MA)p_1 = q_1^tA\left[\pi_i^2(MA)p_1\right].$$

First an auxiliary quantity:

$$\boxed{1}_{i+1} \stackrel{D}{=} \pi_i r_{i+1} = \pi_i r_i - \pi_i AMp_i\alpha_i^{(r)} = \pi_i r_i - AM\pi_i p_i \alpha_i^{(r)}$$
$$= \begin{cases} \boxed{3}_i - AM\boxed{4}_i\alpha_i^{(r)} & \text{cgs} \\ \boxed{2}_i - AM\boxed{4}_i\alpha_i^{(r)} & \text{bcgs} \end{cases}$$

Next the squared residual:

$$\boxed{2}_{i+1} \stackrel{D}{=} \rho_{i+1}r_{i+1} = (\rho_i - AM\pi_i\alpha_i^\ell)r_{i+1}$$
$$= \begin{cases} \rho_i r_i - \rho_i AMp_i\alpha_i - AM\pi_i r_{i+1}\alpha_i & \\ \qquad = \boxed{2}_i - AM(\boxed{3}_i + \boxed{1}_{i+1})\alpha_i & \text{cgs} \\ \pi_i r_{i+1} - AM\pi_i r_{i+1}\alpha_i^\ell & \\ \qquad = \boxed{1}_{i+1} - AM\boxed{1}_{i+1}\alpha_i^\ell & \text{bcgs} \end{cases}$$

The next quantity is not needed for bcgs, since it is equal to the previous:

$$\boxed{3}_{i+1} \stackrel{D}{=} \pi_{i+1}r_{i+1}$$
$$= \pi_{i+1}\rho_{i+1}r_1 = \rho_{i+1}\pi_{i+1}r_1 = \rho_{i+1}r_{i+1} + \rho_{i+1}p_i\beta_i$$
$$= \rho_{i+1}r_{i+1} + \pi_i r_{i+1}\beta_i = \boxed{2}_{i+1} + \boxed{1}_{i+1}\beta_i \qquad \text{cgs only}$$

For the squared search directions we find

$$\boxed{4}_{i+1} \stackrel{D}{=} \pi_{i+1}p_{i+1}$$
$$= \begin{cases} \rho_{i+1}p_{i+1} + \pi_i p_{i+1}\beta_i & \\ \qquad = \rho_{i+1}p_{i+1} + (\pi_i r_{i+1} + \pi_i p_i\beta_i)\beta_i & \\ \qquad = \boxed{3}_{i+1} + \beta_i(\boxed{1}_{i+1} + \boxed{4}_i\beta_i) & \text{cgs} \\ \rho_{i+1}p_{i+1} = \rho_{i+1}r_{i+1} + \beta_i\rho_{i+1}p_i & \\ \qquad = \rho_{i+1}r_{i+1} + \beta_i(\rho_i p_i - \alpha_i^\ell AM\pi_i p_i) & \\ \qquad = \boxed{2}_{i+1} + \beta_i(\boxed{4}_i - \alpha_i^\ell AM\boxed{4}_i) & \text{bcgs} \end{cases}$$

The amount of work per iteration for both methods consists of two matrix-vector products and preconditioner solves; additionally, for cgs there are 6 vector additions or vector-plus-scalar-times-vector operations, for bcgs there are 4 such operations. Updating the iterate takes one extra vector operation (analogous to updating $\boxed{2}_i$) for cgs and two (corresponding to update $\boxed{1}_i$ and $\boxed{2}_i$) for bcgs.

3 Polynomial squaring methods without search directions

Polynomial squaring methods, such as conjugate gradients squared (cgs) or bi-conjugate gradients stabilized (bcgs) can be formulated without search directions. For this, consider a generalized Lanczos method with two series of vectors $R = (\ldots, r_i, \ldots)$ and $S = (\ldots, s_i, \ldots)$ and upper Hessenberg matrices H and K such that

$$AR = RH \qquad \text{and} \qquad A^tS = SK.$$

It is easy to see that for these sequences there are corresponding sequences of polynomials $\{\rho_i\}$ and $\{\sigma_i\}$ so that $r_i = \rho_i(A)r_1$ and $s_i = \sigma_i(A^t)s_1$, where the i-th polynomials have degree $i-1$. The relations $AR = RH$, $A^tS = SK$, or

$$Ar_j = \sum_{i\leq j+1} r_i h_{ij}, \qquad A^t s_j = \sum_{i\leq j+1} s_i k_{ij}, \tag{1}$$

translate into corresponding relations

$$t\rho_j(t) = \sum_{i\leq j+1} \rho_i(t)h_{ij}, \qquad t\sigma_j(t) = \sum_{i\leq j+1} \sigma_i(t)k_{ij} \tag{2}$$

for the polynomials.

The Lanczos method uses the same polynomials for both sequences, so we will introduce a reference sequence $\tilde{R}$ satisfying $A^t\tilde{R} = \tilde{R}H$ and $\tilde{r}_1 = s_1$. Equivalently, the elements of this sequences are generated as

$$\tilde{r}_i = \rho_i(A^t)\tilde{r}_1.$$

Since the degrees of corresponding polynomials σ_i and ρ_i are the same, the sequences $\tilde{R}$ and S are related by

$$s_j = \sum_{i\leq j} \tilde{r}_i v_{ij}, \qquad \tilde{r}_j = \sum_{i\leq j} s_i u_{ij}$$

for certain sets of coefficients u_{ij}, v_{ij}. This fact will be needed later in the bcgs method.

Polynomials squaring methods now compute vectors $r_i^{(j)} = \sigma_i(A)\rho_j(A)r_1$, in particular the sequence $\{r_i^{(i)}\} \equiv \{\sigma_i(A)\rho_i(A)r_1\}$. The relations (1), (2) then translate into

$$r_i^{(j+1)}h_{i+1i} + r_i^{(j)}h_{ii} + r_i^{(j-1)}h_{i-1i} = Ar_i^{(j)} \tag{3}$$

and

$$r_{i+1}^{(j)}k_{i+1i} + r_i^{(j)}k_{ii} + r_{i-1}^{(j)}k_{i-1i} = Ar_i^{(j)}. \tag{4}$$

In all polynomial squaring methods, H is the Hessenberg matrix of the Lanczos method, that is, its elements can be computed from the inner products $s_i^tAr_j$, $s_i^tr_j$ directly or indirectly. For the computation of these inner products the sequence S is not explicitly needed except for its first element. For instance, $s_i^tr_j = s_1\rho_i(A)\rho_j(A)r_1 = s_1^tr_i^{(j)}$.

The cgs method corresponds to the choice $\sigma_i \equiv \rho_i$ for the polynomials. Thus $K = H$, and the elements of the Hessenberg matrix can be computed directly from inner products $s_i^tAr_j = s_1Ar_i^{(j)}$, $\tilde{r}_i^tr_j = s_1r_i^{(j)}$. Because of the commutativity of polynomials in A we have $r_i^{(j)} = r_j^{(i)}$ for all i and j.

For the bcgs method the Hessenberg matrix K is reduced to a lower bidiagonal matrix; the coefficients k_{i+1i}, k_{ii} are chosen in a steepest descent fashion in order to minimize $\|r_{i+1}^{(i+1)}\|$.

Using the above recurrences in i and j direction through the two-parameter family of $r_i^{(j)}$ vectors, we can now construct the single-parameter sequence $\{r_i^{(i)}\}$ in an efficient way.

Suppose that $r_{i-1}^{(i-1)}$, $Ar_{i-1}^{(i-1)}$, $r_{i-1}^{(i)}$, $Ar_{i-1}^{(i)}$, $r_i^{(i)}$, and $Ar_i^{(i)}$ have already been computed. Now perform the following steps:

- Compute $r_i^{(i-1)}$. For cgs this is equal to $r_{i-1}^{(i)}$; for bcgs compute

$$r_i^{(i-1)}k_{ii-1} + r_{i-1}^{(i-1)}k_{i-1i-1} = Ar_{i-1}^{(i-1)}.$$

- Compute $r_i^{(i+1)}$ from

$$r_i^{(i+1)}h_{i+1i} + r_i^{(i)}h_{ii} + r_i^{(i-1)}h_{i-1i} = Ar_i^{(i)}.$$

- Compute $r_{i+1}^{(i+1)}$

$$r_{i+1}^{(i+1)}k_{i+1i} + r_i^{(i+1)}k_{ii} + r_{i-1}^{(i+1)}k_{i-1i} = Ar_i^{(i+1)},$$

  where for bcgs the coefficient k_{i-1i} is zero. This requires having computed $Ar_i^{(i+1)}$, and for cgs additionally $r_{i-1}^{(i+1)}$ needs to be computed from

$$r_i^{(i+1)}h_{i+1i} + r_i^{(i)}h_{ii} + r_i^{(i-1)}h_{i-1i} = Ar_i^{(i)}.$$

The total cost, as in the previous derivation with search directions, is seen to comprise 6 vector operations for cgs and 4 for bcgs. However, unlike in the case of the methods using search directions, for the computation of the iterate there is now an update relation corresponding to each update for the residual. Thus computing the iterate doubles the number of vector operations.

4 Computation of coefficients in CGS

In the conjugate gradient squared method, the choices $\rho_i \equiv \tilde{\rho}_i$ and $H = K$ are made, where the tridiagonal matrix H is chosen as the one generated by the Lanczos method. Denoting the left and right residuals of the Lanczos method by s_i, r_i, we need the values of

$$s_i^t r_i = \left(\rho_i(A^t)r_1\right)^t \left(\rho_i(A)r_1\right) = r_1^t \rho_i^2(A) r_1 = r_1^t r_i^{(i)}, \tag{5}$$

and

$$s_i^t A r_i = \left(\rho_i(A^t)r_1\right)^t A\left(\rho_i(A)r_1\right) = r_1^t A \rho_i^2(A) r_1 = r_1^t A r_i^{(i)}. \tag{6}$$

The coefficients of H then follow from

$$h_{ii} = \frac{s_i A r_i}{s_i^t r_i}, \qquad h_{i-1i} = \frac{s_i r_i}{s_{i-1}^t r_{i-1}} h_{ii-1}, \qquad h_{i+1i} = -h_{ii} - h_{i-1i}. \tag{7}$$

(For a survey of such identities, see [1].)

5 Computation of coefficients in BiCGstab

The BiCGstab algorithm is based on taking a different recurrence in i direction from the one in j direction. While the latter one is still based on the matrix H from the Lanczos method, the former recurrence takes steepest descent steps. Thus, we perform the steepest descent update

$$r_{i+1}^{(i+1)} = r_i^{(i+1)} + \omega_i A r_i^{(i+1)}$$

with

$$\omega_i = -\frac{r_i^{(i+1)^t} A r_i^{(i+1)}}{(A r_i^{(i+1)})^t (A r_i^{(i+1)})}.$$

In order to write this as a three-term recurrence

$$r_{i+1}^{(i+1)} k_{i+1i} + r_i^{(i+1)} k_{ii} + r_{i-1}^{(i+1)} k_{i-1i} = A r_i^{(i+1)}$$

we choose $k_{i-1i} = 0$, and

$$k_{i+1i} = -k_{ii} = 1/\omega_i.$$

Computing the elements of H has become slightly more difficult, however. While we are interested in having for instance the Lanczos coefficient

$$\tilde{r}_i^t r_i = \big(\rho_i(A^t)s_1\big)^t \big(\rho_i(A)r_1\big) = s_1^t \rho_i^2(A) r_1, \tag{8}$$

the only thing we can reasonably compute is

$$s_1^t r_i^{(i)} = s_1^t \sigma_i(A)\rho_i(A) r_1 = \big(\sigma_i(A^t)s_1\big)^t \big(\rho_i(A)r_1\big) = s_i^t r_i. \tag{9}$$

The key to retrieving the Lanczos coefficient is in comparing the left sequence $s_i = \sigma_i(A^t)s_1$ to the reference Lanczos left sequence $\tilde{r}_i = \rho_i(A^t)s_1$.

Since both are polynomial sequences, there are upper triangular matrices U and V such that

$$\tilde{r}_i = \sum_{j=i}^{i} u_{ji}(A^t)^i s_1, \qquad s_i = \sum_{j=i}^{i} v_{ji}(A^t)^i s_1.$$

We find that the sequences are related by $S = \tilde{R}U^{-1}V$. This relation can be carried over to the coefficients. Below are two ways of using the computed quantities to retrieve the original Lanczos coefficients.

Defining the upper triangular matrix $W = V^{-1}U$, we find that

$$\tilde{R}^t R = W^t S^t R \qquad \text{and} \qquad \tilde{R}^t A R = W^t S^t A R,$$

in which $S^t R$ is lower triangular and $S^t A R$ is of lower Hessenberg form. This implies that we can retrieve the inner products in equations (5) and (6) as

$$\tilde{r}_i^t r_i = w_{ii} s_i^t r_i$$

and

$$\tilde{r}_i^t A r_i = w_{ii} s_i^t A r_i + w_{i-1i} s_{i-1}^t A r_i.$$

The only remaining problem is computing the numbers w_{ii} and w_{i-1i}. Note that the original implementation of BiCGstab needs only w_{ii}, since it is based on computing search directions; the inner product $s_{i-1} A r_i$ also has no counterpart in the original algorithm.

The necessary elements of W follow from the equation $VW = U$:

$$w_{ii} = u_{ii}/v_{ii} \qquad \text{and} \qquad w_{i-1i} = (u_{i-1i} - v_{i-1i} w_{ii})/v_{i-1i-1} \tag{10}$$

where we compute the elements of U from H by the recurrences

$$u_{i+1i+1}h_{i+1i} = u_{ii}$$
$$u_{ii+1}h_{i+1i} = u_{i-1i} - u_{ii}h_{ii}$$

and similar formulas for computing V from K.

Using these formulas directly gives to a method that has a danger of leading quickly to floating point underflow. Therefore we note that we only need the ratios

$$\frac{\tilde{r}^t_{i+1}r_{i+1}}{\tilde{r}^t_i r_i} = \frac{s^t_{i+1}r_{i+1}}{s^t_i r_i} \cdot \frac{k_{i+1i}}{h_{i+1i}} \tag{11}$$

and

$$\frac{\tilde{r}^t_{i+1}Ar_{i+1}}{\tilde{r}^t_{i+1}r_{i+1}} = \frac{s^t_{i+1}Ar_{i+1}}{s^t_{i+1}r_{i+1}} + \frac{w_{ii+1}}{w_{i+1i+1}} \cdot \frac{s^t_i Ar_{i+1}}{s^t_{i+1}r_{i+1}} \tag{12}$$

in order to compute the Hessenberg matrix of the bi-conjugate gradient algorithm; see equation (7).

Similar formulas are arrived at by taking for W the inverse of the above choice.

Define then the upper triangular matrix $W = V^{-1}U$. This leads to

$$S^t R = W^t \tilde{R}^t R \qquad \text{and} \qquad S^t AR = W^t \tilde{R}^t AR.$$

Now we find for the inner products in equations (5) and (6) that

$$w_{ii}\tilde{r}^t_i r_i = s^t_i r_i$$

which leads again to formula (11), and

$$w_{ii}\tilde{r}^t_i Ar_i = s^t_i Ar_i - w_{i-1i}\tilde{r}^t_{i-1}Ar_i.$$

which gives, analogous to by slightly differing from (12)

$$\frac{\tilde{r}^t_{i+1}Ar_{i+1}}{\tilde{r}^t_{i+1}r_{i+1}} = \frac{s^t_{i+1}Ar_{i+1}}{s^t_{i+1}r_{i+1}} + \frac{w_{ii+1}}{w_{i+1i+1}} \cdot \frac{\tilde{r}^t_i Ar_{i+1}}{\tilde{r}^t_{i+1}r_{i+1}}. \tag{13}$$

However, computing according to equations (12) and (13) gives methods that first of all take an extra inner product, and secondly, turn out to be extremely numerically unstable[1]. Therefore we substitute for the first method

$$\frac{w_{ii+1}}{w_{i+1i+1}} = \frac{1}{k_{i+1i}}\left(\frac{u_{ii+1}}{u_{i+1i+1}} - \frac{v_{ii+1}}{v_{i+1i+1}}\right)$$

and interchanging the roles of u and v for the second choice of w

$$\frac{w_{ii+1}}{w_{i+1i+1}} = \frac{1}{h_{i+1i}}\left(\frac{v_{ii+1}}{v_{i+1i+1}} - \frac{u_{ii+1}}{u_{i+1i+1}}\right).$$

Using some elementary relation about the Hessenberg matrices H and K, for instance

$$\tilde{r}^t_i Ar_{i+1} = \tilde{r}^t_{i+1}r_{i+1}h_{i+1i}$$

we then find for both choices of W

$$\frac{\tilde{r}^t_{i+1}Ar_{i+1}}{\tilde{r}^t_{i+1}r_{i+1}} = \frac{s^t_{i+1}Ar_{i+1}}{s^t_{i+1}r_{i+1}} + \frac{u_{ii+1}}{u_{i+1i+1}} - \frac{v_{ii+1}}{v_{i+1i+1}}. \tag{14}$$

1. In fact, we don't report tests on such variants, since even on small, well-conditioned problems they stall or diverge within a few iterations.

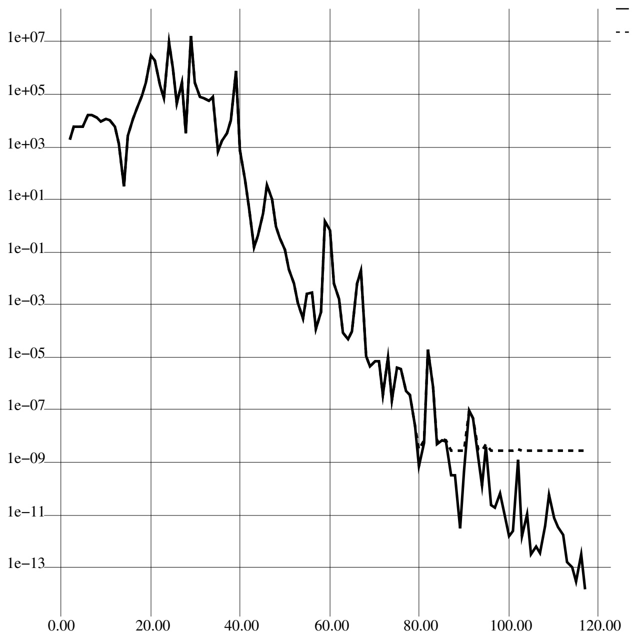

Figure 1: Updated (solid line) and true (dashed line) residual for Classical CGS on model problem (16)

The normalized polynomial coefficients u_{ii+1}/u_{i+1i+1}, v_{ii+1}/v_{i+1i+1} can be calculated recursively. For instance,

$$\frac{u_{ii+1}}{u_{i+1i+1}} = \frac{u_{i-1i}}{u_{ii}} - h_{ii}. \tag{15}$$

but in practice we see no difference in numerical stability between using equation (14) directly, and modifying it with (15).

6 Stability

Since the variants with and without search directions of any given method perform different computations, their results will differ numerically. For recursive methods such as discussed in this paper this means that at some points variants may give completely different results. This phenomenon can be seen by comparing figures 1 and 3 for the classical methods to figures 2 and 4 for the three-term ones.

It can be observed that at some point the residual sizes of computational variants become unrelated, even though in both cases they still converge.

There is another, more interesting, phenomenon related to numerical accuracy in these methods. For this, we consider the difference between the recursively updated residual and the true residual found by computing $Ax_k - b$. In [2] an argument was

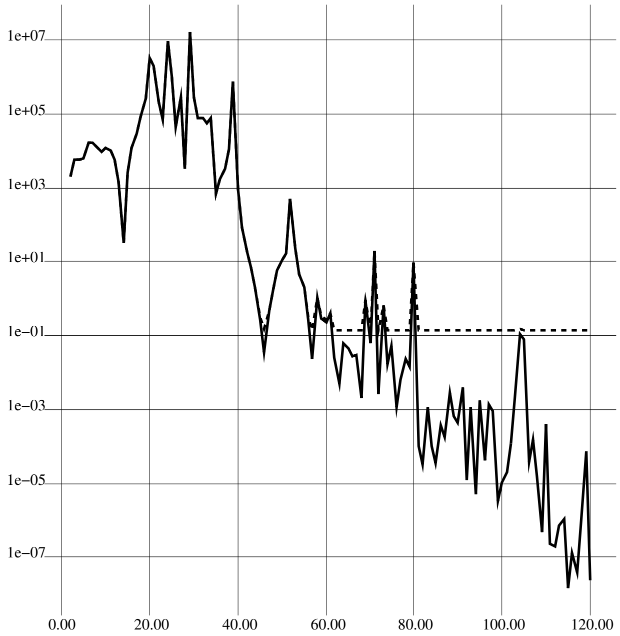

Figure 2: Updated (solid line) and true (dashed line) residual for CGS without search directions on model problem (16)

given why the norm of the true residual cannot get smaller than the maximal residual norm during the iteration times a constant consisting mostly of the machine precision. This means that any increase in the residual size during the iteration will reduce the amount to which the true residual can be reduced.

For the three-term methods presented here (and in fact for ordinary methods based on three-term recurrences) basically the same argument holds. From numerical tests we see that although the qualitative behaviour is the same as for methods using search directions, in a quantitative sense less accuracy is attainable.

7 Experimental results

We solved the following test problem:

$$-\Delta u + 90(u_x + u_y) = f \tag{16}$$

on the unit square, discretized with central differences and $h = 1/31$. The graphs present both the updated residual (solid line) and the true residual (dotted line). One sees that all methods converge, although the three-term recurrences take slightly longer, and have a less regular convergence history. The norm of the true residual is seen to level

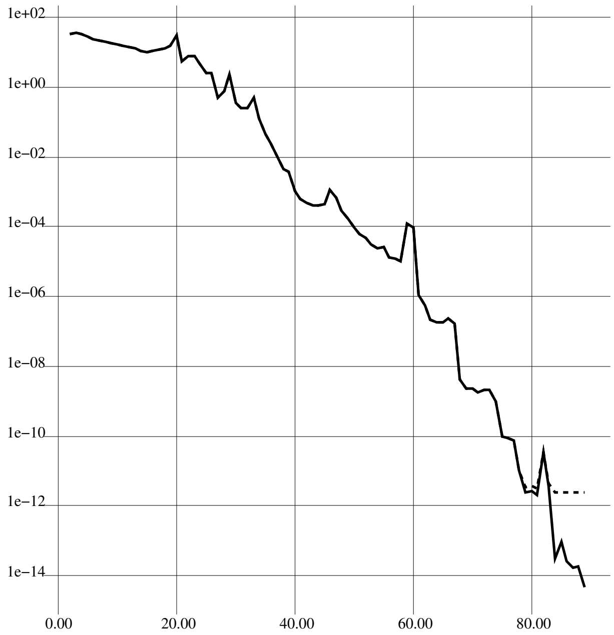

Figure 3: Updated (solid line) and true (dashed line) residual for Classical BiCGstab on model problem (16)

off at some point. We observe that this point lies appreciably higher for the three-term methods than for the classical methods.

8 Conclusion

Polynomial squaring methods based on the three-term recurrences of the Lanczos method can be derived. However, they involve slightly more work than the classical variants of such methods, and they are seen to be numerically less stable.

References

[1] Victor Eijkhout. Lapack working note 51: Qualitative properties of the conjugate gradient and lanczos methods in a matrix framework. Technical Report CS 92-170, Computer Science Department, University of Tennessee, 1992.

[2] G.L.G. Sleijpen, H.A. Van der Vorst, and D.R. Fokkema. Bicgstab(ℓ) and other hybrid Bi-cg methods. submitted.

[3] Peter Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

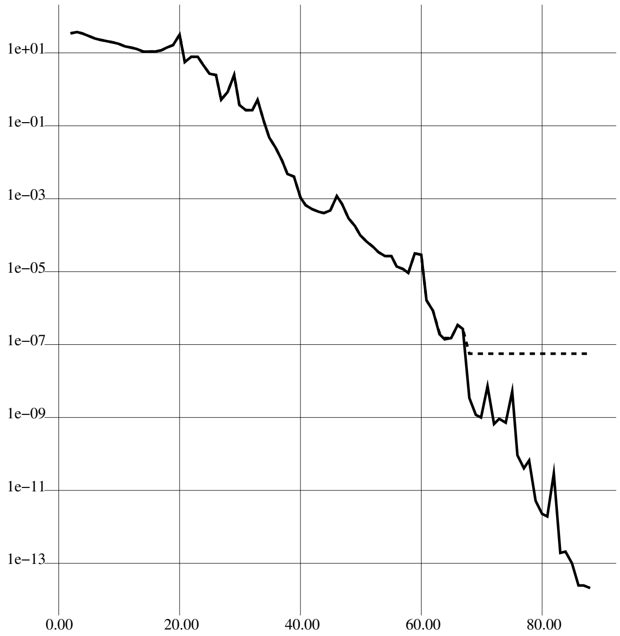

Figure 4: Updated (solid line) and true (dashed line) residual for BiCGstab without search directions on model problem (16)

[4] Henk van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.