MATRICES, MOMENTS AND QUADRATURE II OR HOW TO COMPUTE THE NORM OF THE ERROR IN ITERATIVE METHODS *

G.H. GOLUB[1] AND G. MEURANT[2] †

[1] *Computer Science Department, Stanford University, Stanford CA 94305, USA. email: golub@sccm.stanford.edu*

[2] *CEA/Limeil–Valenton*

94195 Villeneuve St Georges cedex, France. email: meurant@limeil.cea.fr

Abstract. In this paper, we study the numerical computation of the errors in linear systems when using iterative methods. This is done by using methods to obtain bounds or approximations of quadratic forms $u^T A^{-1} u$ where A is a symmetric positive definite matrix and u is a given vector. Numerical examples are given for the Gauss–Seidel algorithm.

Moreover, we show that using a formula for the A–norm of the error from [2], very good bounds of the error can be computed almost for free during the iterations of the conjugate gradient method leading to a reliable stopping criteria.

Key words. Iterative methods, Error computation, Conjugate gradient.

1. Introduction.. Let A be a large, sparse symmetric positive definite matrix of order n and suppose an iterative method is used to compute an approximate solution $\tilde{x}$ of the linear system

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

where b is a given vector. The residual r is defined as,

$$r = b - A\tilde{x}.$$

The error e being $e = x - \tilde{x}$, we obviously have,

$$e = A^{-1} r.$$

Therefore, if we consider the A–norm of the error,

$$\|e\|_A^2 = e^T A e = r^T A^{-1} A A^{-1} r = r^T A^{-1} r.$$

It is sometimes also of interest to study or compute the l_2–norm, for which $\|e\|^2 = r^T A^{-2} r$.

In order to bound or estimate $\|e\|_A$, we must obtain bounds or estimates of $r^T A^{-1} r$, see [1] and [2]. Notice that r is something we can compute but, of course, we do not want to compute A^{-1}.

Therefore, our task is to obtain computable bounds for quadratic forms

$$u^T A^{-1} u, \tag{2}$$

without computing A^{-1}. This problem has been considered at length in [5] and [6], see also [4]. In [5], algorithms combining quadrature formulas and the Lanczos

*Received ?? 1996.

†The work of the first author was partially supported by NSF Grant CCR–950539.

algorithm have been defined that allows us to compute bounds for quadratic forms such as (2) and more generally for $u^T A^{-1} v$. These algorithms are described in Section 2. Then, in Section 3, we see how to compute approximations of the A–norm of the error for iterative methods. As an example we consider the Gauss–Seidel method and give numerical examples. Section 4 is devoted to the particular case of the conjugate gradient (CG) algorithm. Using the connection of CG with the Lanczos algorithm, and a formula from [2], we derive a method to compute reliable bounds of the A–norm of the error during the CG iterations. This improves on the results in [6]. This method adds only a few floating point operations to CG meaning that one can estimate the norm of the error almost for free. Numerical examples are given in Section 5.

2. Quadrature algorithms.. In [5], the more general problem of finding bounds for

$$u^T f(A) v, \tag{3}$$

where u and v are given vectors and f is some smooth (possibly C^∞) function on a given interval of the real line was considered. In the applications we are interested here, we have $u = v$ and $f(x) = \frac{1}{x}$.

The first step of the method is to express the bilinear form in (3) as a Stieljtes integral. Since $A = A^T$, we write A as

$$A = Q \Lambda Q^T,$$

where Q is the orthonormal matrix whose columns are the normalized eigenvectors of A and Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i of A, which we order as

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

By definition, we have

$$A^{-1} = Q \Lambda^{-1} Q^T.$$

Therefore,

$$\begin{aligned} u^T A^{-1} u &= u^T Q \Lambda^{-1} Q^T u \\ &= \alpha^T \Lambda^{-1} \alpha, \\ &= \sum_{i=1}^{n} \lambda_i^{-1} \alpha_i^2. \end{aligned}$$

This last sum can be considered as a Riemann–Stieltjes integral

$$I[A, u] = u^T A^{-1} u = \int_a^b \lambda^{-1} \, d\alpha(\lambda),$$

where the measure α is piecewise constant and defined by

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a = \lambda_1 \\ \sum_{j=1}^{i} \alpha_j^2 & \text{if } \lambda_i \leq \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} \alpha_j^2 & \text{if } b = \lambda_n \leq \lambda \end{cases}$$

The idea is to use quadrature formulas to approximate the Riemann–Stieltjes integral. We use the Gauss, Gauss–Radau and Gauss–Lobatto quadrature formulas. The general formula we will use for a function f is

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f],$$

where the weights $[w_j]_{j=1}^N, [v_k]_{k=1}^M$ and the nodes $[t_j]_{j=1}^N$ are unknowns and the nodes $[z_k]_{k=1}^M$ are prescribed. If $M = 0$, that is no prescribed nodes, this leads to the Gauss rule. If $M = 1$ and $z_1 = a$ or $z_1 = b$ we have the Gauss–Radau formula. If $M = 2$ and $z_1 = a, z_2 = b$, this is the Gauss–Lobatto formula. It is known that the remainder is

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \prod_{k=1}^{M}(\lambda - z_k) \left[\prod_{j=1}^{N}(\lambda - t_j)\right]^2 d\alpha(\lambda), \quad a < \eta < b.$$

The nodes and weights are obtained by considering the sequence of orthonormal polynomials $p_0(\lambda), p_1(\lambda), \ldots$ that are associated to the measure α. This set of orthonormal polynomials satisfies a three term recurrence relationship:

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j) p_{j-1}(\lambda) - \gamma_{j-1} p_{j-2}(\lambda), \quad j = 1, 2, \ldots, N$$

$$p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1,$$

if $\int d\alpha = 1$.

In matrix form, this recurrence can be written as

$$\lambda p(\lambda) = J_N p(\lambda) + \gamma_N p_N(\lambda) e_N,$$

where

$$p(\lambda)^T = [p_0(\lambda)\ p_1(\lambda) \cdots\ p_{N-1}(\lambda)],$$

$$e_N^T = (0\ 0\ \cdots\ 0\ 1),$$

$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \gamma_{N-1} & \omega_N \end{pmatrix}.$$

The eigenvalues of J_N (which are the zeroes of p_N) are the nodes of the Gauss quadrature rule (i. e. $M = 0$). The weights are the squares of the first elements of the normalized eigenvectors of J_N. To obtain the Gauss–Radau and Gauss–Lobatto rules, we must extend the matrix J_N in such a way that is has the prescribed eigenvalues. Details are given in [5].

We note that

$$\sum_{l=1}^{N} w_l f(t_l) = (e_1)^T f(J_N) e_1,$$

where e_1 is the first unit vector, see [5]. Therefore, as the sign of the remainder is known, it is enough to compute the (1,1) element of the inverse of the tridiagonal matrix J_N to obtain a bound for the integral. The same statement is true for the Gauss–Radau and Gauss–Lobatto rules. The Gauss rule gives a lower bound, the Gauss–Radau rule gives both a lower and an upper bound and the Gauss–Lobatto rule gives an upper bound.

The last ingredient of the algorithm is to obtain the coefficients of the recurrences for the orthonormal polynomials. This is done through the use of the Lanczos algorithm.

Let $h_{-1} = 0$ and h_0 be given such that $\|h_0\| = 1$. The Lanczos algorithm is defined by the following relations,

$$\gamma_j h_j = \tilde{h}_j = (A - \omega_j I)h_{j-1} - \gamma_{j-1}h_{j-2}, \quad j = 1, \ldots$$

where

$$\omega_j = h_{j-1}^T A h_{j-1},$$

and

$$\gamma_j = \|\tilde{h}_j\|.$$

The sequence $\{h_j\}_{j=0}^l$ is an orthonormal basis of the Krylov space

$$\text{span}\{h_0, Ah_0, \ldots, A^l h_0\}.$$

Obviously, the vector h_j is given by

$$h_j = p_j(A)h_0,$$

where p_j is a polynomial of degree j defined by the three term recurrence

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}j(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \quad p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1.$$

The (1,1) element of the inverse of the tridiagonal matrix J_N can be computed incrementally as we go through the Lanczos algorithm (see [5]) and the following algorithm GQL (Gauss Quadrature and Lanczos) is finally obtained. **Algorithm GQL**$[A, u, l]$

Suppose $\|u\| = 1$, the following formulas yield a lower bound b_j of $u^T A^{-1} u$ by the Gauss quadrature rule, a lower bound $\underline{b}_j$ and an upper bound $\bar{b}_j$ through the Gauss–Radau quadrature rule and an upper bound $\breve{b}_j$ through the Gauss–Lobatto rule.

Let $h_{-1} = 0$ and $h_0 = u$, $\omega_1 = u^T A u$, $\gamma_1 = \|(A - \omega_1 I)u\|$, $b_1 = \omega_1^{-1}$, $d_1 = \omega_1$, $c_1 = 1$, $\bar{d}_1 = \omega_1 - a$, $\underline{d}_1 = \omega_1 - b$, $h_1 = (A - \omega_1 I)u/\gamma_1$.

Then for $j = 2, \ldots, l$ we compute

$$\omega_j = h_{j-1}^T A h_{j-1},$$

$$\tilde{h}_j = (A - \omega_j I)h_{j-1} - \gamma_{j-1}h_{j-2},$$

$$\gamma_j = \|\tilde{h}_j\|,$$

$$h_j = \frac{\tilde{h}_j}{\gamma_j},$$

$$b_j = b_{j-1} + \frac{\gamma_{j-1}^2 c_{j-1}^2}{d_{j-1}(\omega_j d_{j-1} - \gamma_{j-1}^2)}, \tag{4}$$

$$d_j = \omega_j - \frac{\gamma_{j-1}^2}{d_{j-1}},$$

$$c_j = c_{j-1}\frac{\gamma_{j-1}}{d_{j-1}},$$

$$\bar{d}_j = \omega_j - a - \frac{\gamma_{j-1}^2}{\bar{d}_{j-1}},$$

$$\underline{d}_j = \omega_j - b - \frac{\gamma_{j-1}^2}{\underline{d}_{j-1}},$$

$$\bar{\omega}_j = a + \frac{\gamma_j^2}{\bar{d}_j},$$

$$\underline{\omega}_j = b + \frac{\gamma_j^2}{\underline{d}_j},$$

$$\bar{b}_j = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\bar{\omega}_j d_j - \gamma_j^2)},$$

$$\underline{b}_j = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\underline{\omega}_j d_j - \gamma_j^2)},$$

$$\breve{\omega}_j = \frac{\bar{d}_j \underline{d}_j}{\underline{d}_j - \bar{d}_j}\left(\frac{b}{\bar{d}_j} - \frac{a}{\underline{d}_j}\right),$$

$$\breve{\gamma}_j^2 = \frac{\bar{d}_j \underline{d}_j}{\underline{d}_j - \bar{d}_j}(b-a),$$

$$\breve{b}_j = b_j + \frac{\breve{\gamma}_j^2 c_j^2}{d_j(\breve{\omega}_j d_j - \breve{\gamma}_j{}^2)}.$$

Notice that the bulk of the computations in Algorithm GQL comes about from the matrix vector product Ah_{j-1}.

3. Numerical computation of the A–norm of the error.. As an example for solving $Ax = b$, we consider one of the simplest methods: the Gauss–Seidel algorithm. Let x^0 be given and $A = D + L + L^T$, where D is diagonal and L is strictly lower triangular. Then, the iterates x^k are computed by

$$(D + L)x^k = b - L^T x^{k-1}.$$

Suppose we want to compute bounds for the A–norm of the error at iteration k. The algorithm is the following:
1) Compute the residual $r^k = b - Ax^k = L^T(x^{k-1} - x^k)$. Note that $L^T x^k$ is computed in the algorithm.
2) To compute bounds at iteration k, we set $u = r^k/\|r^k\|$ and run GQL$[A, u, l]$ for a given l.

It is of interest to know how many iterations l of Lanczos we need to obtain at least the order of magnitude of the error. We consider the following simple numerical example.
Example 1: the matrix arises from the 5–point finite difference of the Poisson equation in a unit square. This gives a linear system $Ax = b$ of order $n = m^2$, where

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

each block being of order m and

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

We choose $n = 900$, b such that the exact solution x_{ex} is $x_{ex} = (1, \ldots, 1)^T$ and a zero initial guess x^0. We use $a = 0.02$, $b = 8$ when the "exact" eigenvalues are $\lambda_1 = 0.0205227, \lambda_n = 7.979472$. The computations were done using Matlab 4.1 on an Apple Macintosh Quadra 650.

Although we do not recommend this procedure, at each iteration k of Gauss–Seidel we computed the residual and ran Lanczos to compute bounds for $\|e^k\|_A$. Figure 1 shows the relative differences between the bounds and the exact value of the A–norm of the error as a function of k for $l = 2$. Figure 2 shows the relative differences for Gauss–Seidel iteration $k = 300$ as a function of l. This shows that good estimates (less than 2% can be obtained for only 2 Lanczos iterations. Figure 3 shows the sensitivity of the bounds to the given value of a for 2 iterations of Lanczos and $k = 10$. Remember that the lower bound from the Gauss rule is independent of a and b. Notice that the lower bound from Gauss–Radau seems independent of a. The upper bounds do depend on a but only a rough estimate is needed. When a becomes too large the estimates are not upper bounds anymore and for very large values of a they converge to the lower bounds. The bounds are almost independent of b.

Remarks:

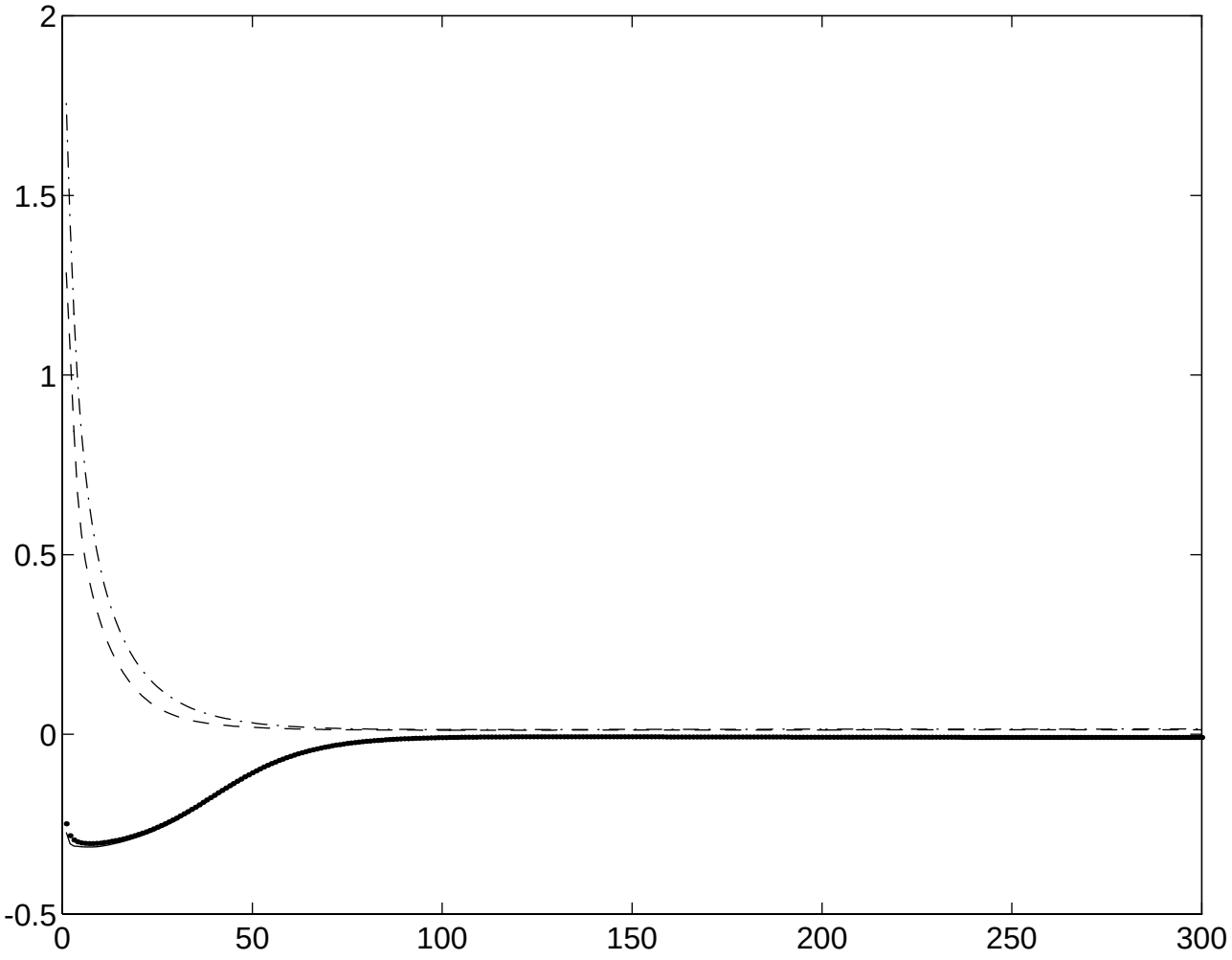

FIG. 1. *Gauss–Seidel,* $\frac{estimate - \|e_{ex}^k\|}{\|e_{ex}^k\|}$ *for* $l = 2$ *as a function of* k*, solid line: Gauss, dots and dashed line: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

i) The bounds are only very slightly dependent on the eigenvalue estimates.
ii) The same method can be used with any iterative method and also with any process that computes an approximation of the solution of $Ax = b$.
iii) Notice we have the following relationship between the A–norm of the error and the l_2–norm,

$$\lambda_1 \|e^k\|^2 \leq \|e^k\|_A^2 \leq \lambda_n \|e^k\|^2.$$

Therefore, if we have $\|e^k\|_A \leq \epsilon$ then, $\|e^k\| \leq \epsilon/\sqrt{\lambda_1}$. We have also

$$\frac{\|r^k\|}{\sqrt{\lambda_n}} \leq \|e^k\|_A \leq \frac{\|r^k\|}{\sqrt{\lambda_1}}.$$

Therefore, if λ_1 is not too much different from λ_n, the l_2 norm of the residual can be a good approximation of the norm of the error.

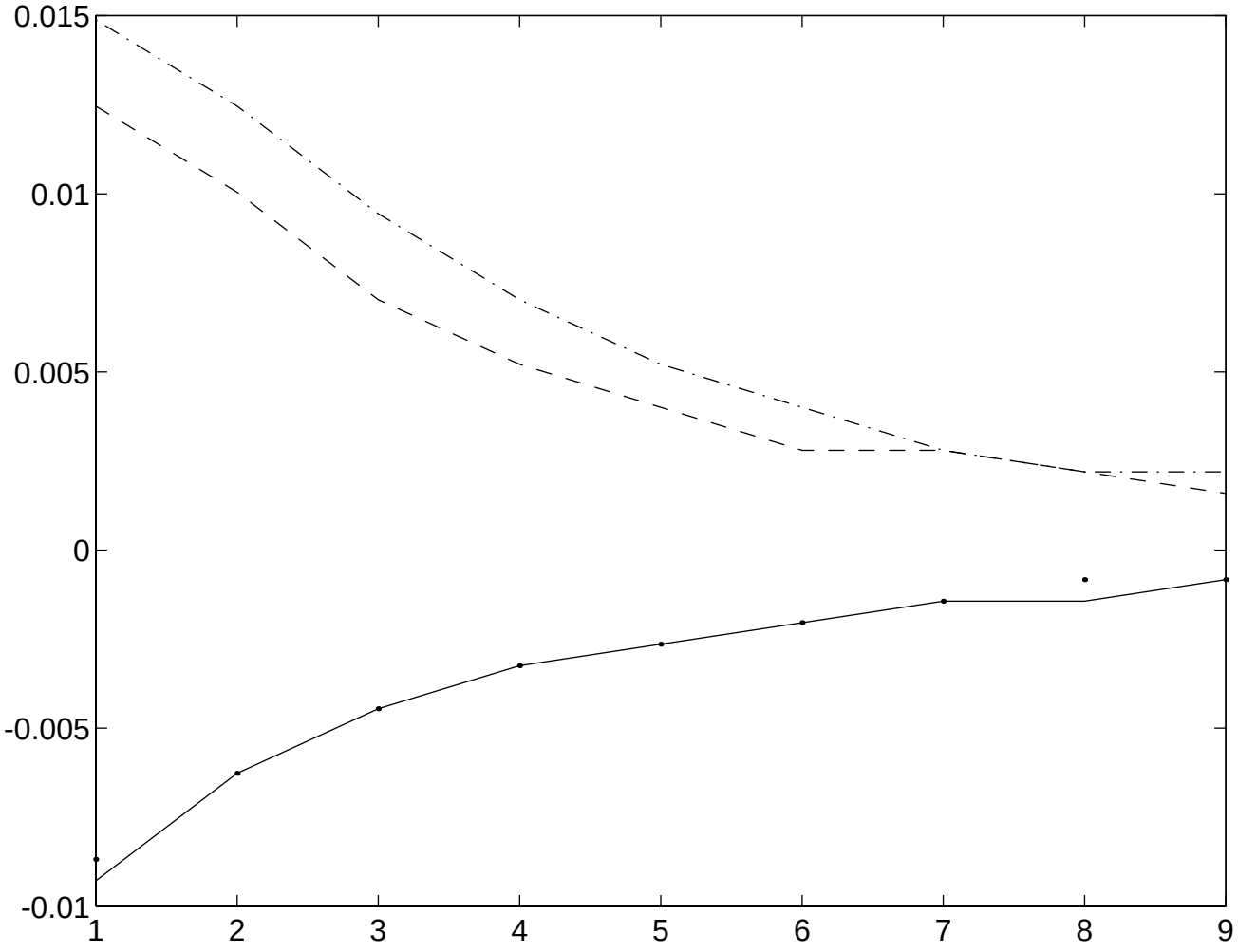

FIG. 2. *Gauss–Seidel,* $\frac{estimate - \|e^k_{ex}\|}{\|e^k_{ex}\|}$ *for* $k = 300$ *as a function of* l*, solid line: Gauss, dots and dashed line: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

iv) The l_2 norm of the error can also be estimated directly. In that case we have to deal with J_k^{-2}. This will be considered in details in a future paper.
v) We note that the Gauss–Seidel iterates can be improved by the Lanczos steps that we ran for computing the error as an approximate solution y of $Ay = r^k$ can be computed from the Lanczos iterates and added to the current approximation x^k. In this way the work that is done to estimate the error is not lost.

4. Computation of the A–norm of the error for CG.. Regarding the computation of the A–norm of the error, the situation of CG is different from other iterative methods. The most common form of CG is the following:

Algorithm CG

Let x^0 be given, $r^0 = b - Ax^0$, $p^0 = r^0$, for $k = 1, \ldots$ until convergence

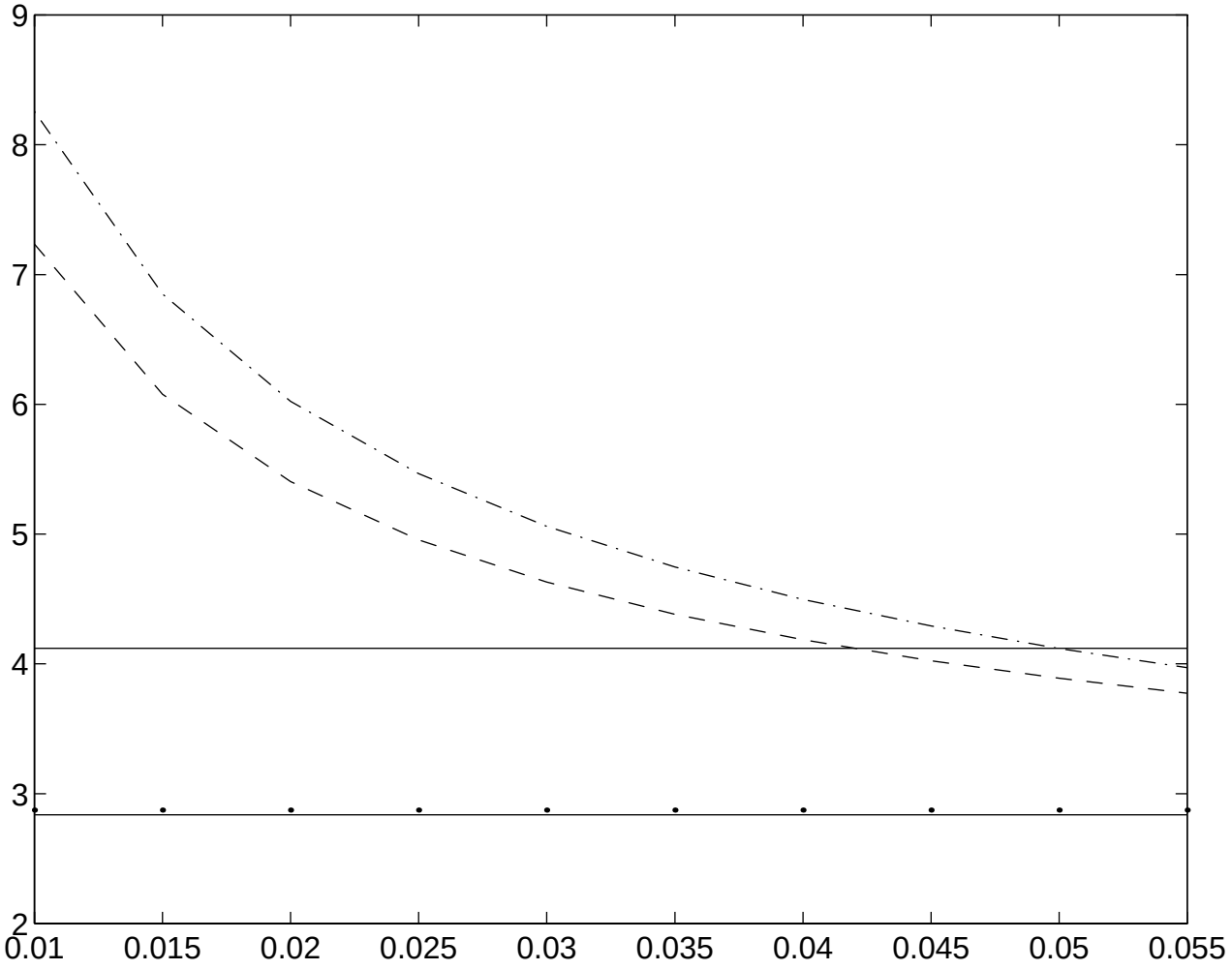

FIG. 3. *Gauss–Seidel, estimate as a function of a for* $k = 10$, $l = 2$, $b = 8$, *solid line: Gauss, dots and dashed line: Gauss–Radau, dot–dashed line: Gauss–Lobatto, The exact value* 4.1208 *is indicated by a solid line*

$$\alpha_{k-1} = \frac{{r^{k-1}}^T r^{k-1}}{{p^{k-1}}^T A p^{k-1}},$$

$$x^k = x^{k-1} + \alpha_{k-1} p^{k-1},$$

$$r^k = r^{k-1} - \alpha_{k-1} A p^{k-1},$$

$$\beta_k = \frac{{r^k}^T r^k}{{r^{k-1}}^T r^{k-1}},$$

$$p^k = r^k + \beta_k p^{k-1}.$$

It is well known that, in some sense, CG is equivalent to Lanczos. In fact, if we start Lanczos from $r^0/\|r^0\|$, then

$$h^{k+1} = (-1)^k \frac{r^k}{\|r^k\|},$$

and we have the following relationship between the Lanczos and CG coefficients, for $k = 1, \ldots$

$$\omega_k = \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-1}}{\alpha_{k-2}}, \quad \beta_0 = 0, \quad \alpha_{-1} = 1$$
$$\gamma_k = \frac{\sqrt{\beta_k}}{\alpha_{k-1}}.$$

Therefore, it is "the same" to run CG for l iterations or GQL$[A, r^0/\|r^0\|, l]$. This means that when running CG we can compute bounds for ${r^0}^T A^{-1} r^0$ that will improve when k grows. So, it would be wasteful to run GQL$[A, r^k/\|r^k\|, l]$ starting from CG results to get bounds of the error at step k.

How can we estimate ${r^k}^T A^{-1} r^k$? We can use a formula from [2] (see [6] for a proof) that relates the A–norm of the error at step k and the inverse of matrix J_k,

$$\|e^k\|_A^2 = \|x - x^k\|_A^2 = {r^0}^T A^{-1} r^0 - \|r^0\|^2 (J_k^{-1})_{(1,1)} \tag{5}$$
$$= \|r^0\|^2 ((J_n^{-1})_{(1,1)} - (J_k^{-1})_{(1,1)}). \tag{6}$$

Notice, we also have

$$\|e^k\|_A^2 = {r^0}^T A^{-1} r^0 - \sum_{j=0}^{k-1} \alpha_j \|r^j\|^2,$$

and hence,

$$\sum_{j=0}^{k-1} \alpha_j \|r^j\|^2 = \|r^0\|^2 (J_k^{-1})_{(1,1)}.$$

Formula (6) has been used in [6] for reconstructing the A–norm of the error but the $(1,1)$ element of the inverse was computed by means of continued fractions. A round off error analysis given in [6] shows that below a certain value of $\|e^k\|_A^2$, then no more useful information can be obtained from this algorithm.

Formula (6) has also been used in [3] but, the computations of $\|e^k\|_A$ were not calculated below 10^{-5}. We will show below that these difficulties can be overcome and that reliable estimates of $\|e^k\|_A$ can be computed.

For the sake of simplicity, let us just consider the lower bound b_k computed by the Gauss rule and let s_k be the estimate of $\|e^k\|_A^2$. Let d be a positive integer, then the idea is to use the following formula at CG iteration k,

$$s_{k-d} = \|r^0\|^2 (b_k - b_{k-d}), \tag{7}$$

where b_k is defined as in (4) to get an estimate of the error at iteration $k - d$. The larger is d, the better will be the estimate.

If we use a naive approach by computing b_k by formula (4), then we run into some difficulties which are similar to the ones described in [6]. Roughly speaking what we see in the numerical experiments is that, for k sufficiently large, $\gamma_{k-1}/d_{k-1} < 1$, and therefore $c_k \to 0$. Let us denote

$$f_{k-1} = \frac{\gamma_{k-1}^2 c_{k-1}^2}{d_{k-1}(\omega_k d_{k-1} - \gamma_{k-1}^2)} > 0,$$

then since the denominator is bounded, $f_k \to 0$ (f_k being a decreasing sequence).

Then computing b_k, we start by summing the largest terms of the sequence. It happens that when $k > \tilde{k}$, f_k adds no significant digit to b_k. Therefore if $k > \tilde{k}$, $b_k = b_{\tilde{k}}$ and $s_{k-d} = 0$ up to working precision when $k > \tilde{k} + d$, and no useful information can be gotten from (7).

But the solution of this problem is very simple. As $b_k = b_{k-1} + f_{k-1}$ and we are not interested in b_k itself but only in $b_k - b_{k-d}$, we can reliably compute the difference by only summing up some of the f_k's. Notice it is likely that they are going to be of the same order of magnitude. The algorithm is the following: **Algorithm CGQL**

Let x^0 be given, $r^0 = b - Ax^0$, $p^0 = r^0$, $\beta_0 = 0$, $\alpha_{-1} = 1$, $c_1 = 1$.

For $k = 1, \ldots$ until convergence

$$\alpha_{k-1} = \frac{{r^{k-1}}^T r^{k-1}}{{p^{k-1}}^T A p^{k-1}},$$

$$\omega_k = \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-1}}{\alpha_{k-2}},$$

if $k = 1$ ———————————————————

$$f_1 = \frac{1}{\omega_1},$$

$$d_1 = \omega_1,$$

$$\bar{d}_1 = \omega_1 - a,$$

$$\underline{d}_1 = \omega_1 - b,$$

else ———————————————————

$$f_k = \frac{\gamma_{k-1}^2 c_{k-1}^2}{d_{k-1}(\omega_k d_{k-1} - \gamma_{k-1}^2)},$$

$$c_k = c_{k-1}\frac{\gamma_{k-1}}{d_{k-1}},$$

$$d_k = \omega_k - \frac{\gamma_{k-1}^2}{d_{k-1}},$$

$$\bar{d}_k = \omega_k - a - \frac{\gamma_{k-1}^2}{\bar{d}_{k-1}},$$

$$\underline{d}_k = \omega_k - b - \frac{\gamma_{k-1}^2}{\underline{d}_{k-1}}$$

end ———————————————————

$$x^k = x^{k-1} + \alpha_{k-1} p^{k-1},$$

$$r^k = r^{k-1} - \alpha_{k-1} A p^{k-1},$$

$$\beta_k = \frac{{r^k}^T r^k}{{r^{k-1}}^T r^{k-1}},$$

$$\gamma_k = \frac{\sqrt{\beta_k}}{\alpha_{k-1}},$$

$$p^k = r^k + \beta_k p^{k-1},$$

$$\bar{\omega}_k = a + \frac{\gamma_k^2}{\bar{d}_k},$$

$$\underline{\omega}_k = b + \frac{\gamma_k^2}{\underline{d}_k},$$

$$\breve{\omega}_k = \frac{\bar{d}_k \underline{d}_k}{\underline{d}_k - \bar{d}_k} \left(\frac{b}{\bar{d}_k} - \frac{a}{\underline{d}_k} \right),$$

$$\breve{\gamma}_k^2 = \frac{\bar{d}_k \underline{d}_k}{\underline{d}_k - \bar{d}_k} (b - a),$$

$$\bar{f}_k = \frac{\gamma_k^2 c_k^2}{d_k(\bar{\omega}_k d_k - \gamma_k^2)},$$

$$\underline{f}_k = \frac{\gamma_k^2 c_k^2}{d_k(\underline{\omega}_k d_k - \gamma_k^2)},$$

$$\breve{f}_k = \frac{\breve{\gamma}_k^2 c_k^2}{d_k(\breve{\omega}_k d_k - {\breve{\gamma}_k}^2)},$$

if $k > d$ ———————————————

$$t_k = \sum_{j=k-d+1}^{k} f_j,$$

$$s_{k-d} = \|r^0\|^2 t_k,$$

$$\bar{s}_{k-d} = \|r^0\|^2 (t_k + \bar{f}_k),$$

$$\underline{s}_{k-d} = \|r^0\|^2 (t_k + \underline{f}_k),$$

$$\breve{s}_{k-d} = \|r^0\|^2 (t_k + \breve{f}_k)$$

end ———————————————

This algorithm gives lower bounds s_{k-d}, $\underline{s}_{k-d}$ and upper bounds $\bar{s}_{k-d}$, $\breve{s}_{k-d}$ of $\|e^{k-d}\|_A^2$.

Notice that in the practical implementation we do not need to store all the f_ks but only the last d. We can also compute only some of the estimates. Remember that the lower bound s_{k-d} does not depend on the eigenvalue bounds a and b.

The additional number of operations is approximately $50 + d$ if we compute the four estimates, which is almost nothing compared to the 10 n operations plus the matrix–vector product of CG.

An interesting question is to know how large d has to be to get a reliable estimate of the error. We are going to see this in the numerical experiments in the next section.

5. Numerical experiments.. Example 1 is the Poisson problem we considered in Section 3. We will consider three more examples. Example 2 arises from the 5–point finite difference approximation of a diffusion equation in a unit square,

$$-\mathrm{div}(a\nabla u)) = f,$$

with Dirichlet boundary conditions. $a(x, y)$ is a diagonal matrix with equal diagonal elements. This element is equal to 1000 in a square $]1/4, 3/4[\times]1/4, 3/4[$, 1 otherwise. Example 3 is the same with different diffusion coefficients. The coefficient in the x direction is 100 if $x \in [1/4, 3/4]$, 1 otherwise. The coefficient in the y direction is constant and equal to 1. For the first three problems, we choose $n = 900$, b such that the exact solution x_{ex} is $x_{ex} = (1, \ldots, 1)^T$ and a random initial guess x^0.

Example 4 is taken from [6]. The matrix A is diagonal. The diagonal elements are defined as

$$\mu_i = a + \frac{i-1}{n-1}(b-a)\rho^{n-i}, \quad i = 2, \ldots, n-1 \quad \mu_1 = a,\ \mu_n = b$$

As in [6], we take $n = 48$, $a = 0.1$, $b = 100$ and $\rho = 0.875$.

5.1. Results for Example 1.. As before $a = 0.02$ and $b = 8$. The naive algorithm gives a zero estimate after 85 iterations of CG. Figure 4 shows the exact error and the bound from the Gauss rule with $d = 2$ on a logarithmic scale. After 60 iterations the two curves are indistinguishable. Figure 5 is a plot of the relative differences between the error and the lower bounds from the Gauss rule for different values of d. We see that when d increases, the bounds get significantly better. Note it is not difficult to take a large value of d. The only drawbacks are:
1) the number of operations increase, but not very significantly,
2) we only get bounds for the error d iterations before the current one. If we use this to stop the iterations, we are loosing d iterations as the actual error will be smaller.

Figure 6 is a picture of the relatives differences for $d = 10$ as a function of k. We see that the lower bounds are better than the upper ones. But this improves when d increases. Moreover, the bounds improve when k gets larger.

We see that the bounds depend only slightly on the estimates of the smallest eigenvalue. However, it is well known that the smallest and largest eigenvalues of J_k approximate the smallest and largest eigenvalues of A. Therefore, we can devise an adaptive algorithm for estimating a. For some CG iterations, we compute the smallest eigenvalue of J_k by an inverse power iteration. This is cheap as we have at hand the LDL^T decomposition of the tridiagonal matrix J_k. In fact, the d_k that are computed in CGQL are the diagonal elements of the decomposition. When, the smallest eigenvalue has converged it replaces the initial estimate. In this way, even though the bounds can be very crude in the first CG iterations, as soon as we switch, we recover very good upper bounds.

5.2. Results for Example 2.. In the next two examples, we scale the matrix by its diagonal. Therefore, $b = 2$. For example 2, we take $a = 10^{-5}$ when the exact eigenvalue is $1.022\ 10^{-5}$. Figure 7 shows the logarithm of the error and the bounds for $d = 20$. Note that the results are not as good as for example 1, especially for the upper bounds. But, after 50 CG iterations the bounds are very good. This is exemplified even more in Figure 8. We see that when $k > 50$, the percentage of error is around 10%.

5.3. Results for Example 3.. This example is more difficult than the preceding ones as the error does not decrease too much for 300 iterations. We take $a = 1.49\ 10^{-4}$ which is a little bit less than the exact value. Figure 9 shows the logarithm of the error and the bounds for $d = 10$. The percentage of error is given on Figure 10. Once again, the lower bounds are better than the upper ones. But the later ones improve when k gets large. A larger value of d will give better results.

5.4. Results for Example 4.. This example was devised by Z. Strakoš to produce large rounding errors in CG. Note that we need 100 iterations to reach an error of 10^{-10} when the order of the matrix is $n = 48$. However CGQL gives good bounds for the error even with a small value of d. In Figure 11, we choose $d = 2$. Figure 12 gives the relative differences for $d = 10$. The results are quite satisfactory.

6. Conclusion.. In this paper, we have shown that bounds for the A–norm of the error can be easily computed by using Lanczos and quadrature rules. Moreover in the case of CG, bounds can be computed during the CG iterations almost for free. These bounds can be much better than some obtained by other ways when a sufficiently large value of the delay d is used. In CG, the estimate of the smallest eigenvalue can be obtained also during the iterations leading to an algorithm which

FIG. 4. *CG, Example 1, Log*$_{10}$ *of* $\|e^k\|_A$ *(dotted line) and of the square root of* s_k *(solid line) for the Gauss rule,* $d = 2$

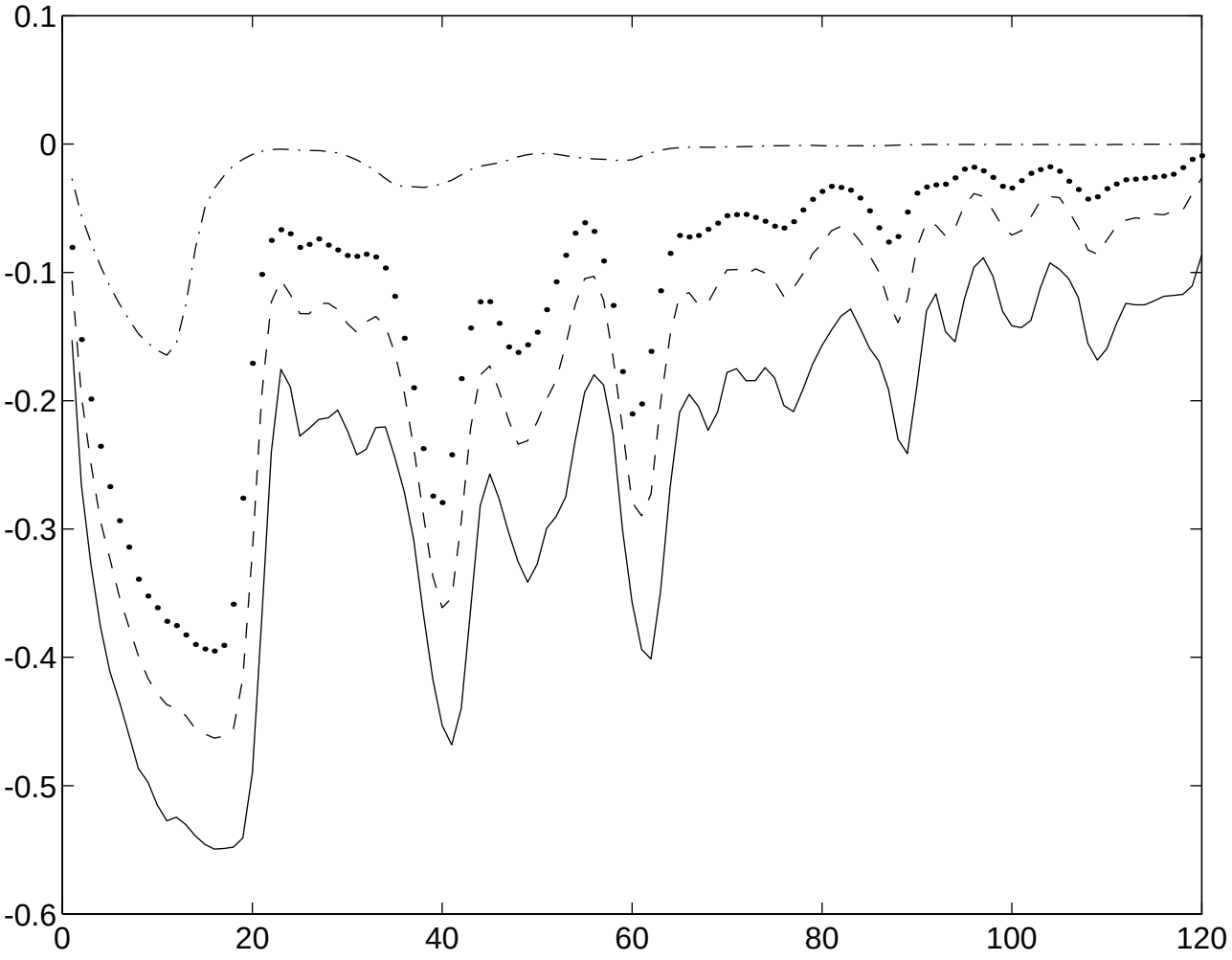

FIG. 5. *CG, Example 1,* $\frac{estimate - \|e^k\|_A}{\|e^k\|_A}$ *as a function of* k *for the Gauss rule, solid line:* $d = 2$*, dashed line:* $d = 3$*, dotted line:* $d = 4$*, dot–dashed line:* $d = 10$

is only slightly dependent on the eigenvalues estimates. This gives a more reliable stopping criteria for CG.

REFERENCES

[1] G. Dahlquist, S.C. Eisenstat and G.H. Golub. *Bounds for the error of linear systems of equations using the theory of moments* J. Math. Anal. Appl. 37 (1972) pp. 151–166.

[2] G. Dahlquist, G.H. Golub and S.G. Nash. *Bounds for the error in linear systems.* In Proc. of the Workshop on Semi–Infinite Programming, R. Hettich ed, Springer (1978), pp. 154–172.

[3] B. Fischer and G.H. Golub. *On the error computation for polynomial based iteration methods.* Report NA 92–21, Stanford University (1992).

[4] G.H. Golub. *Matrix computation and the theory of moments.* In Proceedings of the International Congress of Mathematicians, Birkhäuser, (1995).

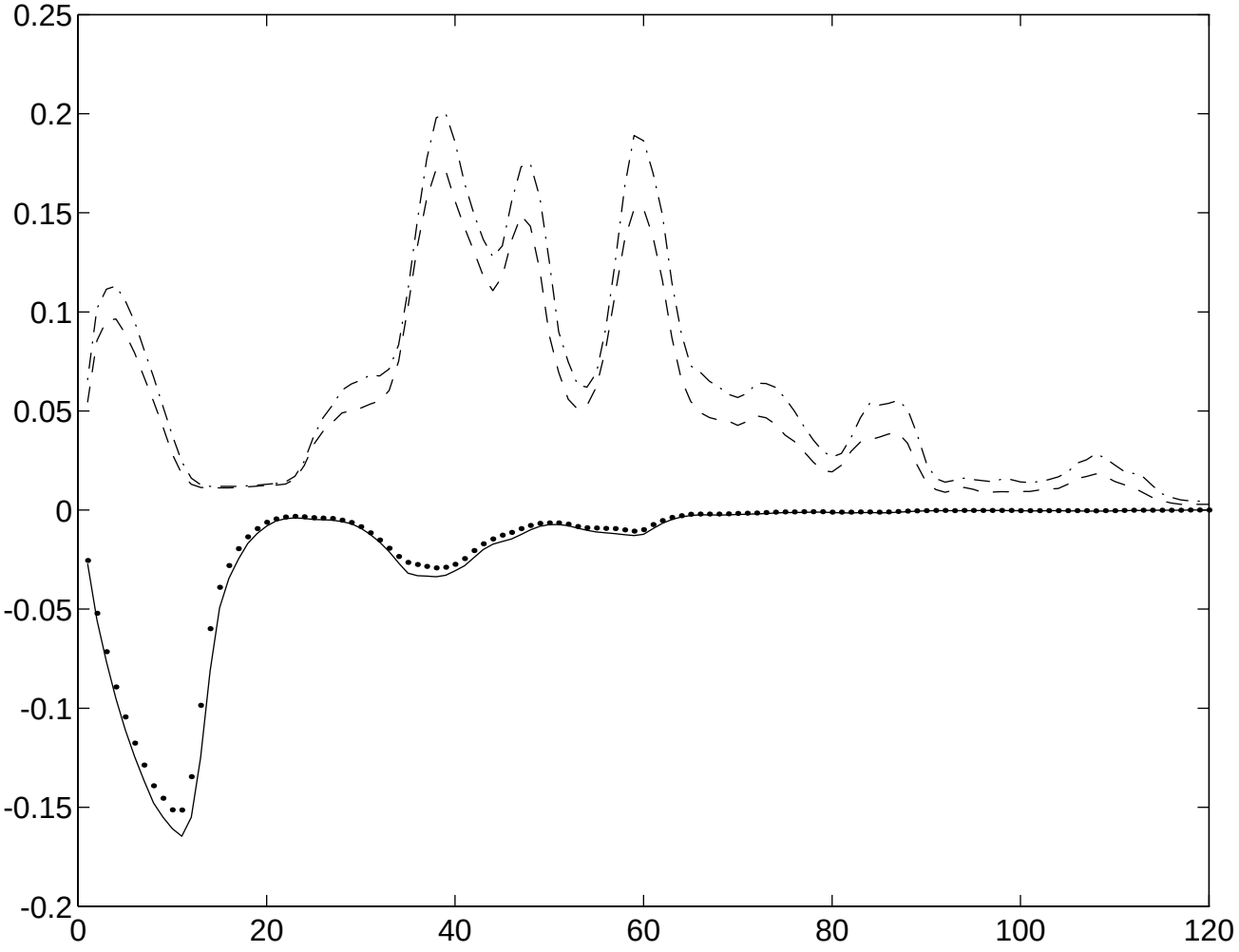

FIG. 6. *CG, Example 1,* $\frac{estimate - \|e^k\|_A}{\|e^k\|_A}$, $d = 10$ *as a function of* k*, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

[5] G.H. Golub and G. Meurant. *Matrices, moments and quadrature.* In Numerical Analysis 1993, D.F. Griffiths & G.A. Watson, Eds. Pitman Research Notes in Mathematics, v 303, (1994), pp. 105–156.

[6] G.H. Golub and Z. Strakoš. *Estimates in quadratic formulas.* Report SCCM–93–08, Stanford University, accepted for publication in Numerical Algorithms.

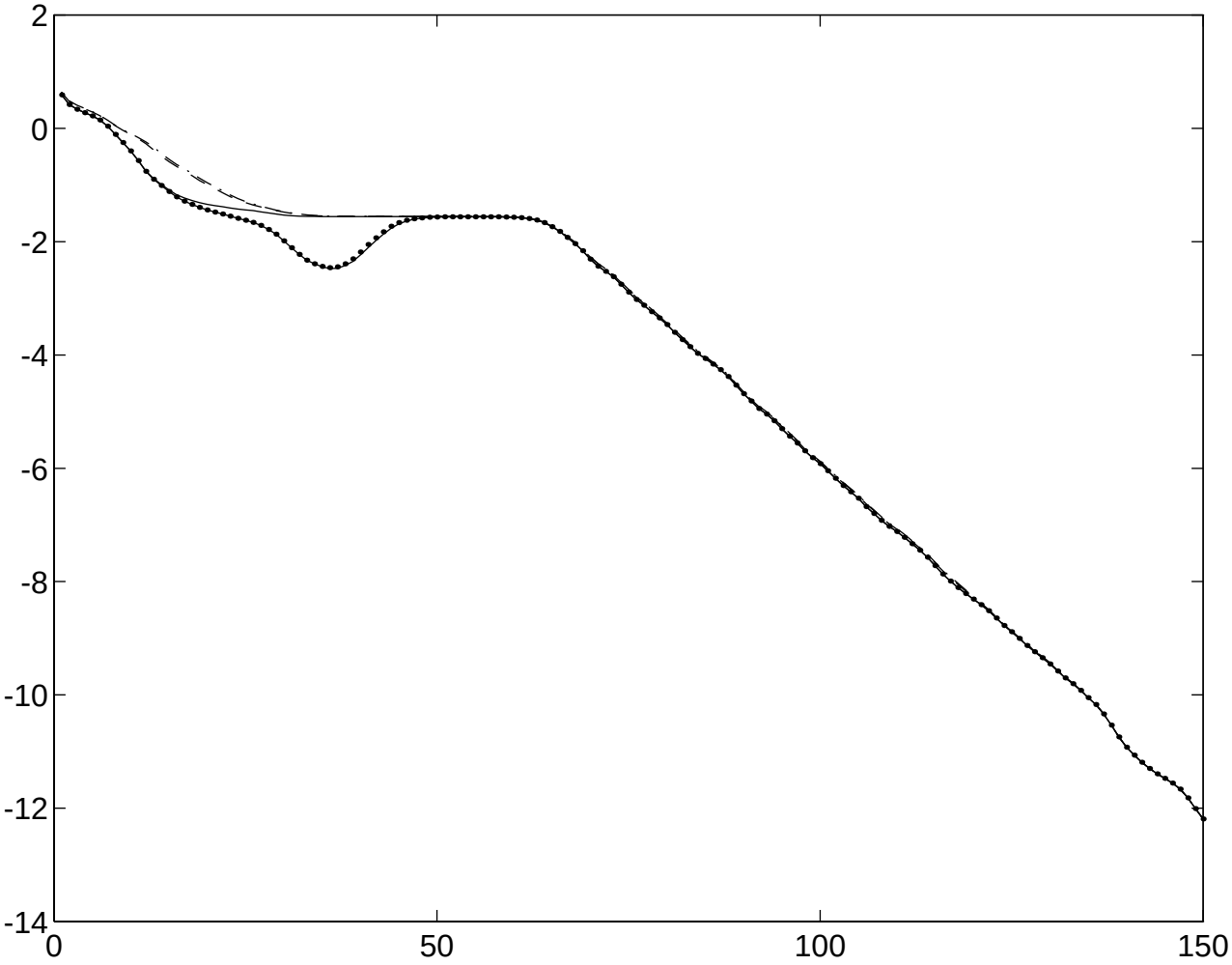

FIG. 7. *CG, Example 2, Log_{10} of $\|e^k\|_A$ (solid line) and of estimates, $d = 20$, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

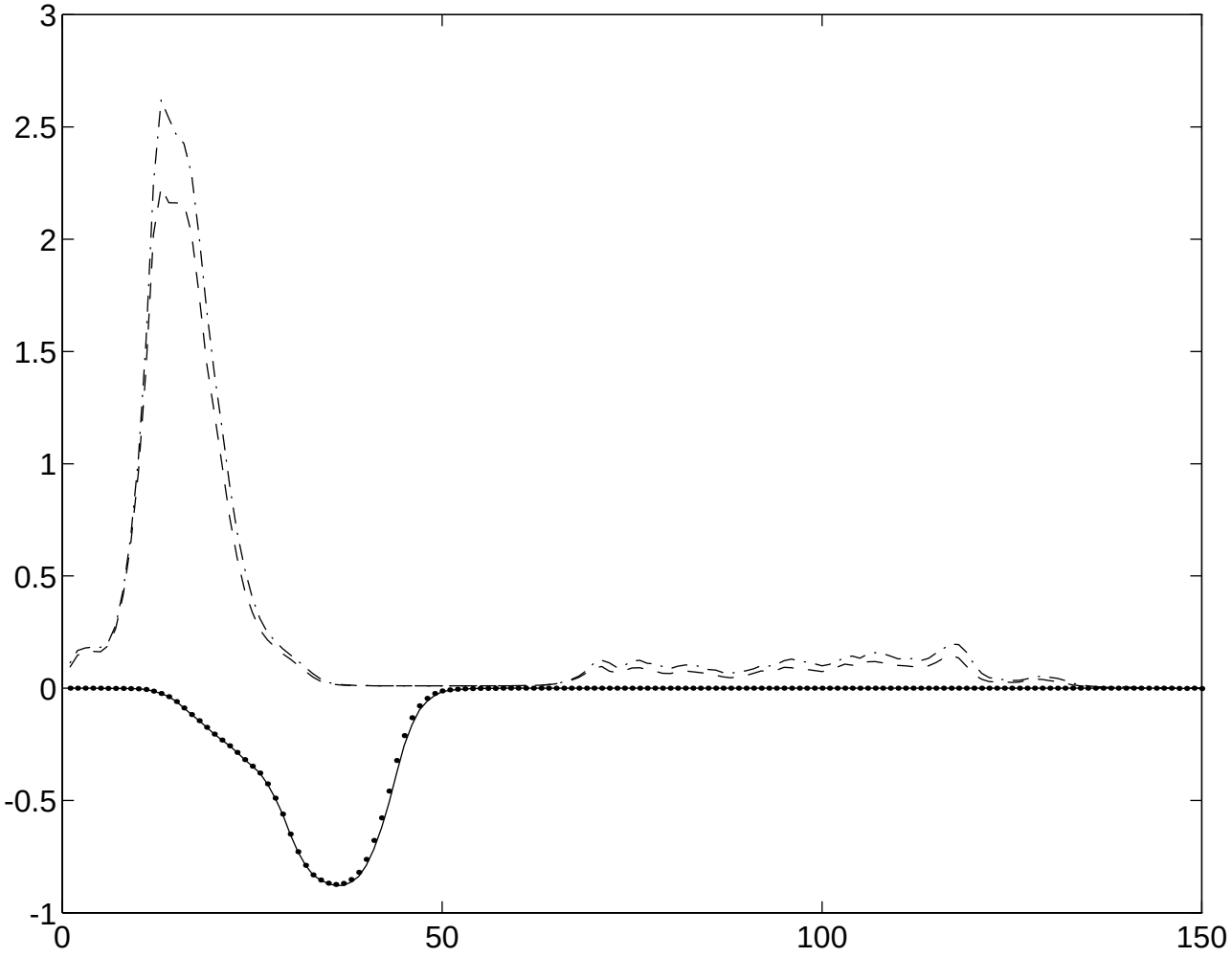

FIG. 8. *CG, Example 2,* $\frac{estimate - \|e^k\|_A}{\|e^k\|_A}$, $d = 20$ *as a function of* k*, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

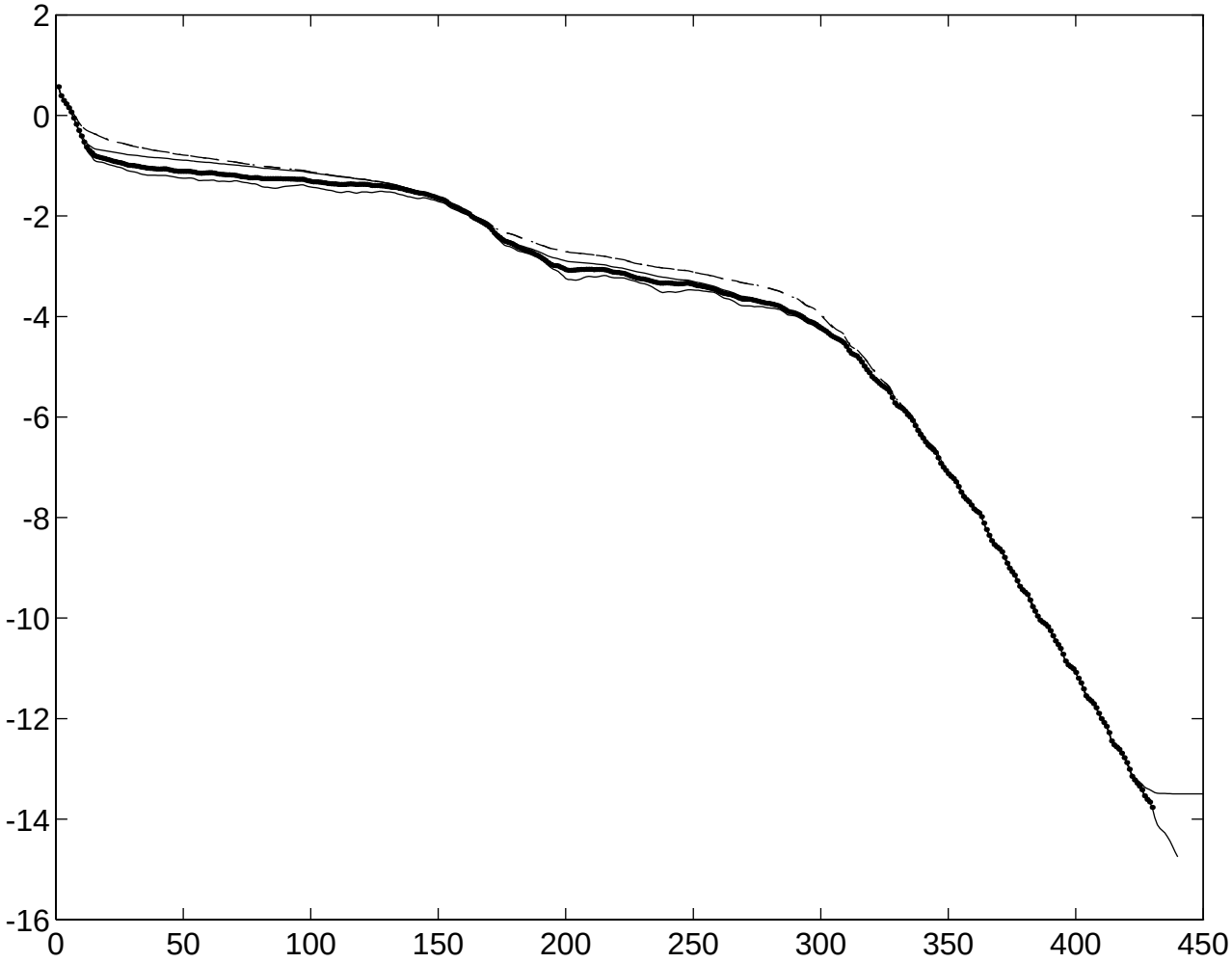

FIG. 9. *CG, Example 3, Log_{10} of $\|e^k\|_A$ (solid line) and of estimates, $d = 10$, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

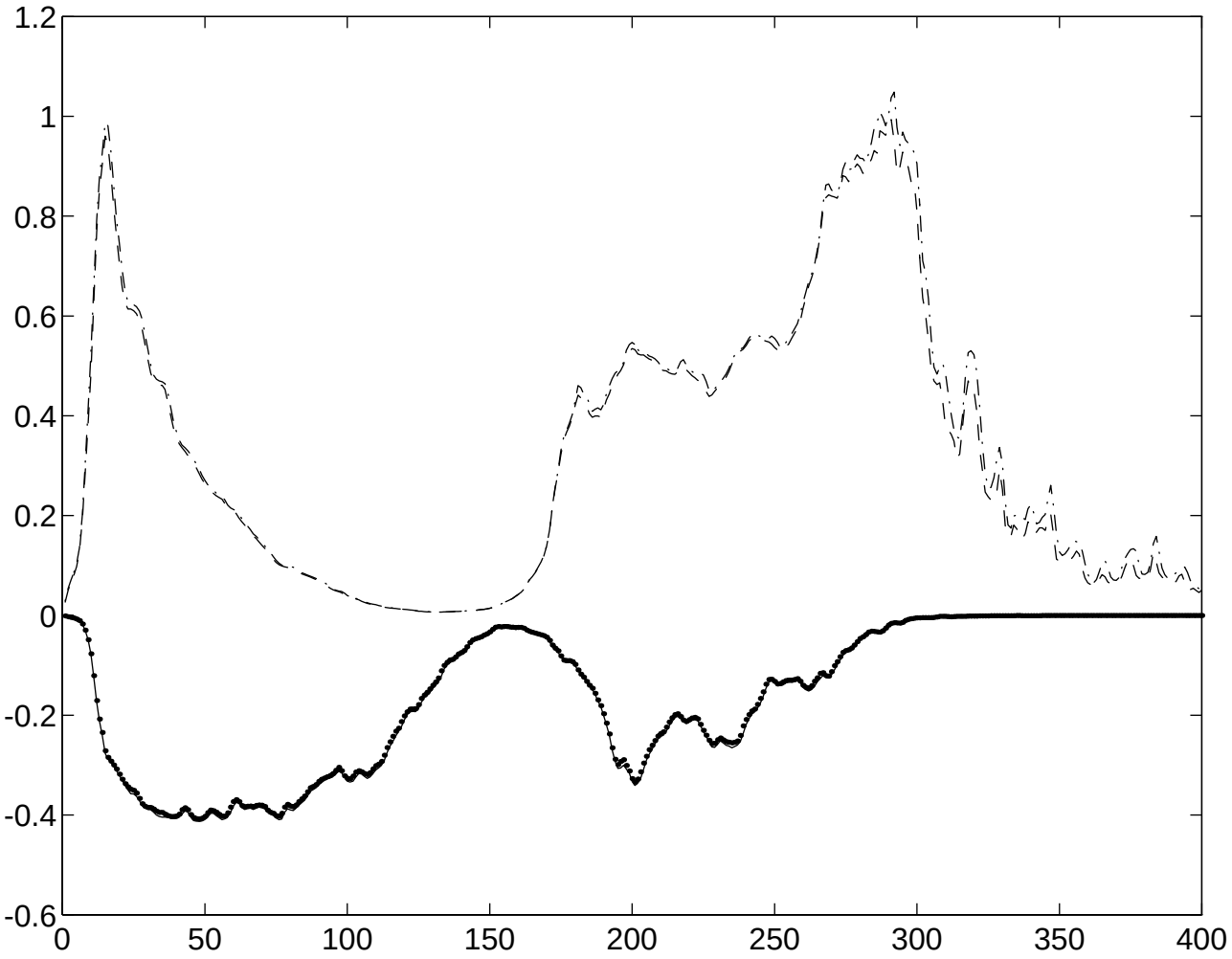

FIG. 10. *CG, Example 3,* $\frac{estimate - \|e^k\|_A}{\|e^k\|_A}$, $d = 10$ *as a function of* k*, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

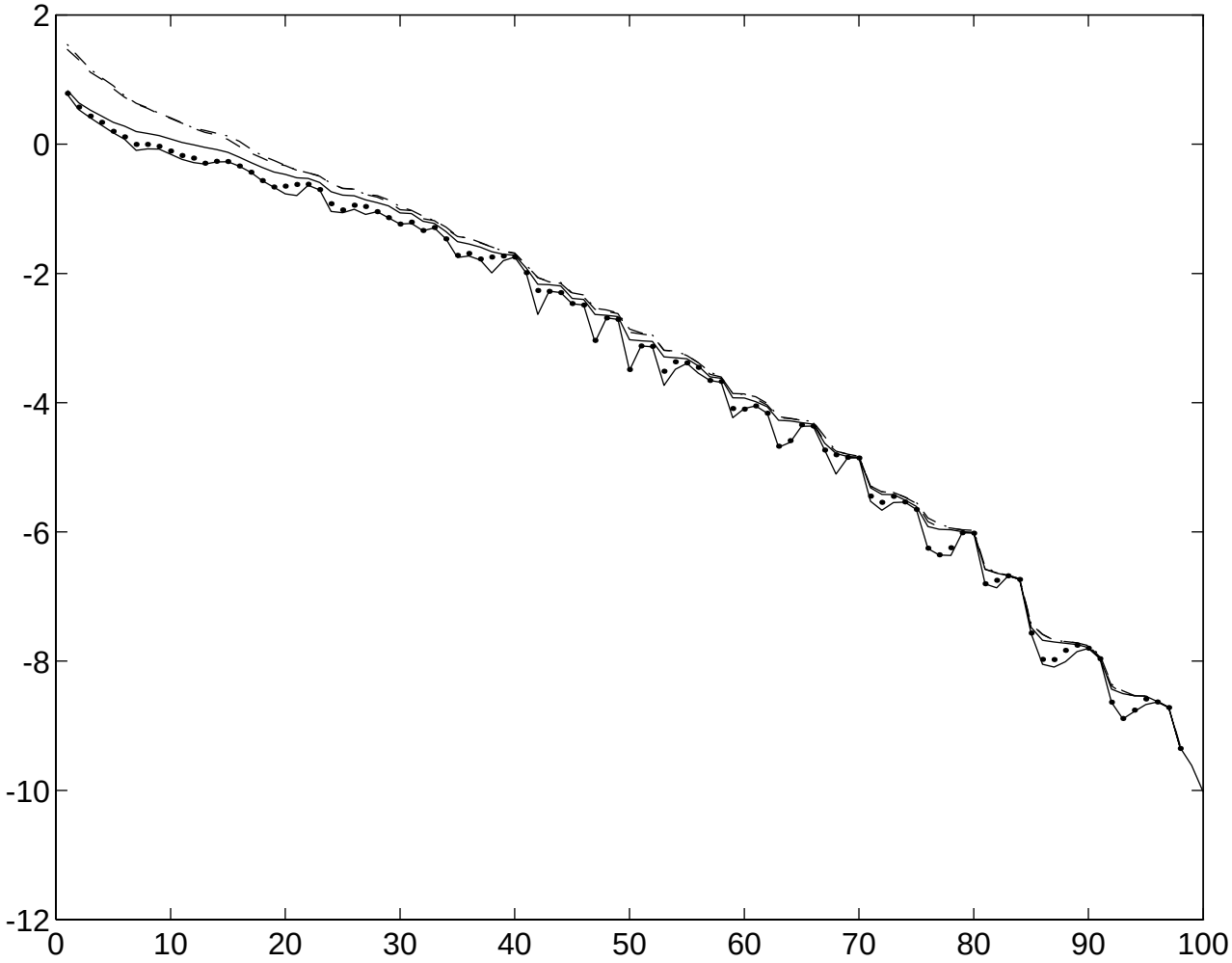

FIG. 11. *CG, Example 4, Log_{10} of $\|e^k\|_A$ (solid line) and of estimates, $d = 2$, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*

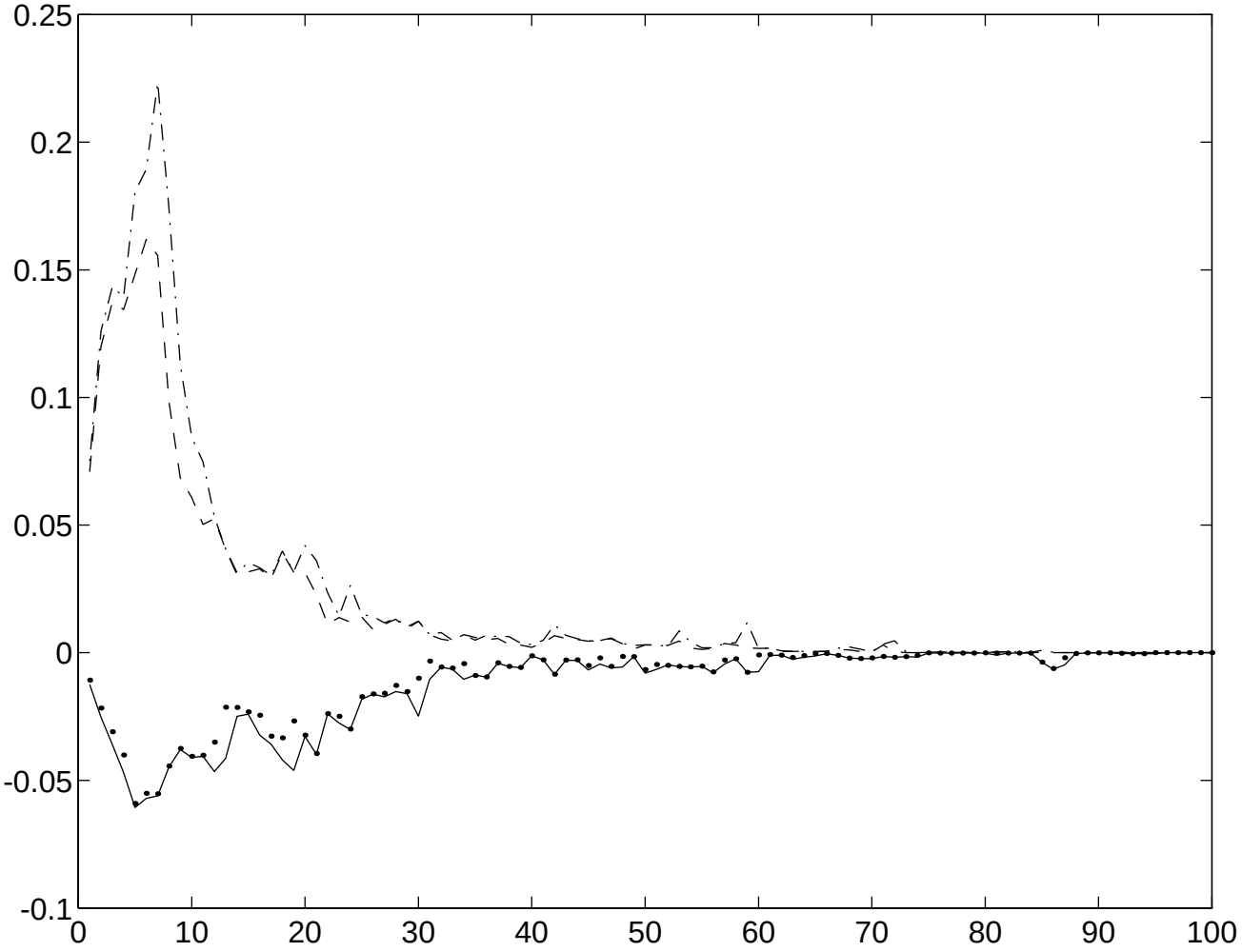

FIG. 12. *CG, Example 4,* $\frac{estimate - \|e^k\|_A}{\|e^k\|_A}$, $d = 10$ *as a function of* k*, solid line: Gauss, dashed and dotted lines: Gauss–Radau, dot–dashed line: Gauss–Lobatto*