ON COMPUTING QUADRATURE-BASED BOUNDS FOR THE A-NORM OF THE ERROR IN CONJUGATE GRADIENTS*

GÉRARD MEURANT† AND PETR TICHÝ‡

Abstract. In their original paper, Golub and Meurant [BIT, 37 (1997), pp. 687–705] suggest to compute bounds for the A-norm of the error in the conjugate gradient (CG) method using Gauss, Gauss-Radau and Gauss-Lobatto quadratures. The quadratures are computed using the $(1, 1)$-entry of the inverse of the corresponding Jacobi matrix (or its rank-one or rank-two modifications). The resulting algorithm called CGQL computes explicitly the entries of the Jacobi matrix and its modifications from the CG coefficients. In this paper, we use the fact that CG computes the Cholesky decomposition of the Jacobi matrix which is given implicitly. For Gauss-Radau and Gauss-Lobatto quadratures, instead of computing the entries of the modified Jacobi matrices, we directly compute the entries of the Cholesky decompositions of the (modified) Jacobi matrices. This leads to simpler formulas in comparison to those used in CGQL.

Key words. Conjugate Gradients, norm of the error, bounds for the error norm.

AMS subject classifications. 15A06, 65F10.

1. Introduction. Today the Conjugate Gradient (CG) algorithm is the iterative method of choice for solving linear systems with a real positive definite symmetric matrix. It is almost always used with a preconditioner to speed up convergence. CG was introduced in the beginning of the 1950s by Magnus Hestenes and Eduard Stiefel [17]. It can be derived from several different perspectives, as an orthogonalization algorithm or as a minimization process. It can also be obtained from the Lanczos algorithm [19] that was published almost at the same time.

When using CG for solving a linear system $Ax = b$ an important question is when to stop the iterations. Ideally, one would like to stop the iterations when the norm of the error $\varepsilon_k = x - x_k$, where x_k are the CG iterates, is small enough. However, the error is unknown and most CG implementations rely on stopping criteria like $\|r_k\| \leq \epsilon\|b\|$ where $r_k = b - Ax_k$ is the residual vector, which is computed in CG or even $\|r_k\| \leq \epsilon\|r_0\|$. These types of stopping criteria can be misleading depending on the norm of A or the choice of the initial approximation. This was already noticed in the Hestenes and Stiefel paper [17, p. 410]. It can stop the iterations too early when the norm of the error is still too large, or too late in which case too many floating point operations have been done for obtaining the required accuracy. This motivated some researchers to look for ways to compute estimates of some norms of the error during CG iterations. The norm of the error which is particularly interesting for CG is the A-norm (also called the energy norm) which is minimized at each iteration. The A-norm of the error has an important meaning in physics and mechanics, and plays a fundamental role in evaluating convergence [1, 18]. It is defined as

$$\|\varepsilon_k\|_A \equiv (\varepsilon_k^T A \varepsilon_k)^{1/2}. \tag{1.1}$$

Inspired by the connection of CG with Riemann-Stieltjes integrals (already noticed in [17]), a way of research on this topic was started by Gene Golub in the 1970s

†30 rue du sergent Bauchat, 75012 Paris, France, email: gerard.meurant@gmail.com

‡Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 18207 Prague, Czech Republic, email: mail.tichy@gmail.com

*This work was supported by the project No. IAA100300802 of the Grant Agency of the Academy of Sciences of the Czech Republic. It was finalized in 2011 during a visit of G. Meurant to the Institute of Computer Science of the Academy of Sciences of the Czech Republic.

and continued throughout the years with several collaborators ([6, 7, 8, 11, 14, 12]). In particular, it was known that the A-norm of the error can be written as a Riemann-Stieltjes integral for an unknown stepwise constant measure depending on the eigenvalues of A. The main idea of Golub and his collaborators was to obtain bounds for this integral by using Gauss quadrature rules. It turns out that these bounds can be computed without the knowledge of the stepwise constant measure and at almost no cost during the CG iterations as we will see in the next sections.

In [11], these techniques were used for providing lower and upper bounds for quadratic forms $u^T f(A)u$ where f is a smooth function, A is a symmetric matrix and u is a given vector. The algorithm GQL (Gauss Quadrature and Lanczos) was based on the Lanczos algorithm and on computing functions of Jacobi matrices (and their rank-one or rank-two modifications). Later [12, 21], these techniques were adapted to the CG algorithm to compute lower and upper bounds on the A-norm of the error for which the function is $f(x) = 1/x$. The idea was to use CG instead of the Lanczos algorithm, to compute explicitly the entries of the corresponding Jacobi matrices and their modifications from the CG coefficients, and then to use the same formulas as in GQL. The formulas were summarized in the CGQL algorithm (QL standing again for Quadrature and Lanczos). Extensions to preconditioned CG were given in [22, 29]. This research is summarized in the books [23, 13]. The formula for the Gauss rule was analyzed for finite precision arithmetic in [28] where it is shown that it is still valid in finite precision up to small terms proportional to the unit roundoff.

The CGQL algorithm, whose most recent version is described in [13], may seem complicated, particularly for computing bounds with the Gauss-Radau or Gauss-Lobatto quadrature rules. It uses the tridiagonal Jacobi matrix obtained by translating the coefficients computed in CG into the Lanczos coefficients. Therefore the analysis of the formulas is difficult. Our aim in this paper is to show that these formulas can be considerably simplified by working with the LDL^T factorizations of the Jacobi matrices and their modifications instead of computing the Lanczos coefficients explicitly. In other words, one can obtain the bounds from the CG coefficients without computing the Lanczos coefficients. Therefore we hope that with the simpler new formulas the computation of upper bounds for the A-norm of the error can be incorporated more easily into existing CG codes.

It is fair to note that there exist other ways to compute estimates of the norms of the error; see [2, 3]. The paper [3] uses extrapolation techniques. However, this only gives estimates of the norm of the error and not bounds.

The outline of the paper is as follows. Section 2 recalls some basic facts about the Lanczos and CG algorithms, their connection to the approximation of the Riemann-Stieltjes integral using various quadrature rules, about computing quadratures using a convenient modification of the corresponding Jacobi matrix, and finally about estimating the A-norm of the error in CG. Section 3 describes the algebraic background for the new formulas; it is shown how to compute efficiently the entries of the LDL^T factorizations of modified Jacobi matrices. These results are then used in Section 4 in the formulation of the new algorithm called CGQ. Section 5 shows how to modify these new formulas when using preconditioning and finally, Section 6 presents numerical experiments which show that the new formulas are not only simpler but also slightly more accurate than the previous ones.

Throughout the paper e_k denotes the kth column of the identity matrix of appropriate order.

2. Conjugate Gradient, Lanczos and quadratures. In this section we briefly recall the Lanczos and Conjugate Gradient algorithms as well as their relationships; see, for instance, [15, 23].

Algorithm 1 Lanczos algorithm

input A, v
$\beta_0 = 0$, $v_0 = 0$
$v_1 = v/\|v\|$
for $k = 1, \ldots$ **do**
  $w = Av_k - \beta_{k-1} v_{k-1}$
  $\alpha_k = v_k^T w$
  $w = w - \alpha_k v_k$
  $\beta_k = \|w\|$
  $v_{k+1} = w/\beta_k$
end for

2.1. The Lanczos and CG algorithms. Given a starting vector v and a symmetric matrix $A \in \mathbb{R}^{N\times N}$, one can consider a sequence of nested subspaces

$$\mathcal{K}_k(A, v) \equiv \text{span}\{v, Av, \ldots, A^{k-1}v\}, \qquad k = 1, 2, \ldots,$$

called Krylov subspaces. The dimension of these subspaces is increasing up to an index n called *the grade of v with respect to A*, at which the maximal dimension is attained, and $\mathcal{K}_n(A, v)$ is invariant under multiplication with A. Assuming that $k < n$, the Lanczos algorithm (Algorithm 1) computes an orthonormal basis $v_1, \ldots, v_{k+1}$ of the Krylov subspace $\mathcal{K}_{k+1}(A, v)$. In Algorithm 1 we have used the modified Gram-Schmidt form of the algorithm. The basis vectors v_j satisfy the matrix relation

$$AV_k = V_k T_k + \beta_k v_{k+1} e_k^T$$

where $V_k = [v_1 \cdots v_k]$ and T_k is the $k \times k$ symmetric tridiagonal matrix of the recurrence coefficients computed in Algorithm 1:

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

The coefficients β_j being positive, T_k is a Jacobi matrix. The Lanczos algorithm works for any symmetric matrix, but if A is positive definite, then T_k is positive definite as well.

When solving a system of linear algebraic equations $Ax = b$ with symmetric and positive definite matrix A, the CG method (Algorithm 2) can be used. CG computes iterates x_k that are optimal since the A-norm of the error defined in (1.1) is minimized over the manifold $x_0 + \mathcal{K}_k(A, r_0)$,

$$\|x - x_k\|_A = \min_{y \in x_0 + \mathcal{K}_k(A, r_0)} \|x - y\|_A.$$

The residual vectors $r_k = b - Ax_k$ are proportional to the Lanczos basis vectors v_j and hence mutually orthogonal,

$$v_{j+1} = (-1)^j \frac{r_j}{\|r_j\|}, \qquad j = 0, \ldots, k.$$

Algorithm 2 Conjugate gradient algorithm

input A, b, x_0
$r_0 = b - Ax_0$
$p_0 = r_0$
for $k = 1, \ldots, n$ until convergence **do**
  $\gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$
  $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
  $r_k = r_{k-1} - \gamma_{k-1} A p_{k-1}$
  $\delta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$
  $p_k = r_k + \delta_k p_{k-1}$
end for

Therefore, the residual vectors r_j yield an orthogonal basis of the Krylov subspaces $\mathcal{K}_{k+1}(A, r_0)$. In this sense, CG can be seen as an algorithm for computing an orthogonal basis of the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$ and there is a close relationship between the CG and Lanczos algorithms. It is well-known (see, for instance [23]) that the recurrence coefficients computed in both algorithms are connected via

$$\beta_k = \frac{\sqrt{\delta_k}}{\gamma_{k-1}}, \quad \alpha_k = \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}. \quad \delta_0 = 0, \quad \gamma_{-1} = 1.$$

Writing these formulas in a matrix form, we get

$$T_k = L_k D_k L_k^T \tag{2.1}$$

where T_k is the Jacobi matrix resulting from the Lanczos algorithm and

$$L_k \equiv \begin{bmatrix} 1 & & & \\ \sqrt{\delta_1} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\delta_{k-1}} & 1 \end{bmatrix}, \qquad D_k \equiv \begin{bmatrix} \gamma_0^{-1} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \gamma_{k-1}^{-1} \end{bmatrix}. \tag{2.2}$$

In other words, CG implicitly computes an LDL^T factorization of the Jacobi matrix T_k generated by the Lanczos algorithm. In this paper we are interested in computing bounds for the A-norm of the error. Noticing that the error ε_k and the residual r_k are related through $A\varepsilon_k = r_k$, we have

$$\|\varepsilon_k\|_A^2 = \varepsilon_k^T A \varepsilon_k = r_k^T A^{-1} r_k.$$

The quantity on the right-hand side is a quadratic form. In the next subsection we briefly recall how quadratic forms are related to Riemann-Stieltjes integrals. This will allow us to compute bounds for the norm of the error.

2.2. Connection with Riemann-Stieltjes integrals. Let

$$A = U \Lambda U^T, \quad UU^T = U^T U = I, \tag{2.3}$$

be the eigendecomposition of the symmetric matrix A where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ and $U = [u_1, \ldots, u_N]$. For simplicity of notation we assume that all the eigenvalues

of A are distinct and ordered as $\lambda_1 < \lambda_2 < \ldots < \lambda_N$ (the generalization of the below defined function $\omega(\lambda)$ to the case of multiple eigenvalues is straightforward). Let v_1 be a given unit vector. Define the weights ω_i by

$$\omega_i \equiv (v_1, u_i)^2 \qquad \text{so that} \qquad \sum_{i=1}^{N} \omega_i = 1\,, \tag{2.4}$$

and the (nondecreasing) distribution function $\omega(\lambda)$ with a finite number of points of increase $\lambda_1, \lambda_2, \ldots, \lambda_N$,

$$\omega(\lambda) \equiv \begin{cases} 0 & \text{for} \quad \lambda < \lambda_1\,, \\ \sum_{j=1}^{i} \omega_j & \text{for} \quad \lambda_i \le \lambda < \lambda_{i+1}\,, \quad 1 \le i \le N-1\,, \\ 1 & \text{for} \quad \lambda_N \le \lambda\,. \end{cases} \tag{2.5}$$

Having the distribution function $\omega(\lambda)$ and an interval $\langle \zeta, \xi \rangle$ such that $\zeta < \lambda_1 < \lambda_2 < \ldots < \lambda_N < \xi$, for any continuous function f, one can define the Riemann-Stieltjes integral (see, for instance [13])

$$\int_{\zeta}^{\xi} f(\lambda)\, d\omega(\lambda). \tag{2.6}$$

Since $\omega(\lambda)$ is a stepwise constant function and all points of increase lie in the open interval (ζ, ξ), the integral (2.6) is a finite sum and it holds that

$$\int_{\zeta}^{\xi} f(\lambda)\, d\omega(\lambda) = \sum_{i=1}^{N} \omega_i f(\lambda_i) \;=\; v_1^T f(A) v_1. \tag{2.7}$$

The quantity $v_1^T f(A) v_1$ can be expressed using the tridiagonal matrix T_n stemming from the Lanczos algorithm (note that n is the grade of v_1 with respect to A). In the nth step of the Lanczos algorithm we get the full orthonormal basis of $\mathcal{K}_n(A, v_1)$ and we have

$$AV_n = V_n T_n \qquad \Rightarrow \qquad f(A) V_n = V_n f(T_n)$$

and, therefore,

$$v_1^T f(A) v_1 = v_1^T f(A) V_n e_1 = v_1^T V_n f(T_n) e_1 = e_1^T f(T_n) e_1.$$

From this it is clear that the quadratic form we are interested in, $r_k^T A^{-1} r_k$, can be written as a Riemann-Stieltjes integral for the function $f(\lambda) = 1/\lambda$.

2.3. Quadrature formulas. The integral (2.7), i.e. the quantity $v_1^T f(A) v_1$, can be approximated by quadrature formulas, for example the Gauss, Gauss-Radau and Gauss-Lobatto rules; see, for instance, [9, 13]. The general quadrature formula we use has the form

$$\int_{\zeta}^{\xi} f\, d\omega(\lambda) \;=\; \sum_{i=1}^{k} w_i f\,(\nu_i) + \sum_{j=1}^{m} \widetilde{w}_j f(\widetilde{\nu}_j) + \mathcal{R}_k[f],$$

where the weights $[w_i]_{i=1}^{k}$, $[\widetilde{w}_j]_{j=1}^{m}$ and the nodes $[\nu_i]_{i=1}^{k}$ are *unknowns* and the nodes $[\widetilde{\nu}_j]_{j=1}^{m}$ are *prescribed* outside the open integration interval. In our case it is sufficient

when the prescribed nodes are strictly smaller than λ_1 or strictly larger than λ_N. The unknown nodes and weights are chosen to maximize the degree of exactness of the quadrature rule. If $m = 0$, there are no prescribed nodes, and we obtain the Gauss rule. If $m = 1$ we have the Gauss-Radau rule and if $m = 2$, this is the Gauss-Lobatto rule. It is known (see, for instance, [27]) that if $f \in C^{2k+m}$, then the remainder is

$$\mathcal{R}_k[f] = \frac{f^{(2k+m)}(\upsilon)}{(2k+m)!} \int_\zeta^\xi \prod_{j=1}^{m} (\lambda - \widetilde{\nu}_j) \left[\prod_{i=1}^{k} (\lambda - \nu_i)\right]^2 d\omega(\lambda)\,, \qquad \upsilon \in (\zeta, \xi).$$

For some functions f of interest the sign of the remainder term is known.

Consider first the Gauss rule, i.e. $m = 0$. The nodes ν_i and the weights w_i of the kth Gauss quadrature approximation are implicitly determined by the Lanczos algorithm; the nodes are the eigenvalues of T_k generated by the Lanczos algorithm started from v_1 and the weights are the squares of the first components of the normalized eigenvectors of T_k; see [30, 16].

To obtain the Gauss-Radau and Gauss-Lobatto rules, we must extend the matrix T_k in such a way that it has the prescribed nodes as eigenvalues; see [10]. Suppose that μ is a prescribed node. For the Gauss-Radau quadrature rule, we have to determine the coefficient $\tilde{\alpha}_{k+1}^{(\mu)}$ so that μ is an eigenvalue of the extended matrix

$$\tilde{T}_{k+1}^{(\mu)} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{k-1} & \\ & & \beta_{k-1} & \alpha_k & \beta_k \\ & & & \beta_k & \tilde{\alpha}_{k+1}^{(\mu)} \end{bmatrix}. \tag{2.8}$$

Given two prescribed nodes μ and η, for the Gauss-Lobatto quadrature rule we have to find the coefficients $\tilde{\alpha}_{k+1}^{(\mu,\eta)}$ and $\tilde{\beta}_k^{(\mu,\eta)}$ such that μ and η are eigenvalues of the extended matrix

$$\tilde{T}_{k+1}^{(\mu,\eta)} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{k-1} & \\ & & \beta_{k-1} & \alpha_k & \tilde{\beta}_k^{(\mu,\eta)} \\ & & & \tilde{\beta}_k^{(\mu,\eta)} & \tilde{\alpha}_{k+1}^{(\mu,\eta)} \end{bmatrix}. \tag{2.9}$$

Having the matrices T_k, $\tilde{T}_{k+1}^{(\mu)}$ and $\tilde{T}_{k+1}^{(\mu,\eta)}$, the Gauss, Gauss-Radau and Gauss-Lobatto quadrature rules can be respectively written in the form (see [13])

$$\begin{aligned} e_1^T f(T_n) e_1 &= e_1^T f(T_k) e_1 + \mathcal{R}_k^{(G)}[f]\,, \\ e_1^T f(T_n) e_1 &= e_1^T f\left(\tilde{T}_{k+1}^{(\mu)}\right) e_1 + \mathcal{R}_k^{(R)}[f]\,, \\ e_1^T f(T_n) e_1 &= e_1^T f\left(\tilde{T}_{k+1}^{(\mu,\eta)}\right) e_1 + \mathcal{R}_k^{(L)}[f]\,. \end{aligned}$$

These rules can provide lower and upper bounds on the integral (2.7), based on the following implications (see, e.g., [13, Theorem 6.4 and 6.5]):

(2.10) If $f^{(2k)}(\lambda) > 0$ for all $\lambda \in \langle \zeta, \xi \rangle$, then $\mathcal{R}_k^{(G)}[f] > 0$.

(2.11) If $f^{(2k+1)}(\lambda) < 0$ for all $\lambda \in \langle \zeta, \xi \rangle$, and $\mu \leq \lambda_1$, then $\mathcal{R}_k^{(R)}[f] < 0$.

(2.12) If $f^{(2k+2)}(\lambda) > 0$ for all $\lambda \in \langle \zeta, \xi \rangle$, and $\mu \leq \lambda_1$ and $\lambda_N \leq \eta$, then $\mathcal{R}_k^{(L)}[f] < 0$.

If the derivatives of the function f satisfy the assumptions in (2.10)–(2.12), then the Gauss rule gives a lower bound and the Gauss-Radau and Gauss-Lobatto rules give upper bounds for the integral (2.7). Note that the assumptions on the sign of the derivatives of f in (2.10)–(2.12) are satisfied for the function $f(\lambda) = 1/\lambda$.

2.4. CG and Gauss quadrature. For the quadratic form involved in CG we are interested in the function $f(\lambda) = 1/\lambda$. Previous results imply that we can express the Gauss quadrature rule using the Lanczos-related quantities as

$$\left(T_n^{-1}\right)_{1,1} = \left(T_k^{-1}\right)_{1,1} + \mathcal{R}_k^{(G)}[\lambda^{-1}].$$

In [28] the authors show that the same equation multiplied by $\|r_0\|^2$ can be written using the CG-related quantities

$$\|x - x_0\|_A^2 = \sum_{j=0}^{k-1} \gamma_j \|r_j\|^2 + \|x - x_k\|_A^2 .$$

In other words, CG can be see as a procedure that implicitly determines weights and nodes of the Gauss quadrature rule applied to the Riemann-Stieltjes integral

$$\int_\zeta^\xi \lambda^{-1} \, d\omega(\lambda) = \frac{\|x - x_0\|_A^2}{\|r_0\|^2}$$

for which the Gauss quadrature approximation is given by

$$\left(T_k^{-1}\right)_{1,1} = \frac{1}{\|r_0\|^2} \sum_{j=0}^{k-1} \gamma_j \|r_j\|^2. \tag{2.13}$$

The remainder is nothing but the scaled and squared A-norm of the kth error,

$$\mathcal{R}_k^{(G)}\left[\lambda^{-1}\right] = \frac{\|x - x_k\|_A^2}{\|r_0\|^2}.$$

For more information on this topic see, e.g., [14], [28, Section 3] or [24, Subsection 3.3].

2.5. Estimating the A-norm of the error in CG. Of course, at CG iteration k we do not know $(T_n^{-1})_{1,1}$ or $\|x - x_0\|_A$. For estimating the A-norm of the error in CG we consider the Gauss quadrature rule at step k,

$$\|x - x_0\|_A^2 = \|r_0\|^2 \left(T_k^{-1}\right)_{1,1} + \|x - x_k\|_A^2, \tag{2.14}$$

and a (eventually modified) quadrature rule at step $k + d$, $d > 0$,

$$\|x - x_0\|_A^2 = \|r_0\|^2 \left(\hat{T}_{k+d}^{-1}\right)_{1,1} + \hat{\mathcal{R}}_{k+d}\left[\lambda^{-1}\right], \tag{2.15}$$

where $\hat{T}_{k+d}$ stands for the matrix T_{k+d} (in the case of using Gauss rule) or a suitable modification of T_{k+d} (in the case of using Gauss-Radau or Gauss-Lobatto rules). From the equations (2.14) and (2.15) we get

$$\|x - x_k\|_A^2 = \hat{Q}_{k,d} + \hat{\mathcal{R}}_{k+d}\left[\lambda^{-1}\right] \ , \ \hat{Q}_{k,d} \equiv \|r_0\|^2 \left(\left(\hat{T}_{k+d}^{-1}\right)_{1,1} - \left(T_k^{-1}\right)_{1,1} \right). \tag{2.16}$$

$\hat{Q}_{k,d}$ represents either a lower bound on $\|x - x_k\|_A^2$ if $\hat{\mathcal{R}}_{k+d}\left[\lambda^{-1}\right] > 0$, or an upper bound in the case $\hat{\mathcal{R}}_{k+d}\left[\lambda^{-1}\right] < 0$. It means that at CG iteration $k+d$, by computing $\hat{Q}_{k,d}$, we can obtain a bound for the A-norm of the error at iteration k.

From the computational point of view, as noted in [14] and [21], it is not convenient to compute $\hat{Q}_{k,d}$ by first computing $\left(T_k^{-1}\right)_{1,1}$, $\left(\hat{T}_{k+d}^{-1}\right)_{1,1}$, and then taking the difference. By subtracting both quantities we loose accuracy and, as a result, the use of the estimate is limited by the square root of machine precision. Instead of subtracting, it is better to use the following identity

$$\left(\hat{T}_{k+d}^{-1}\right)_{1,1} - \left(T_k^{-1}\right)_{1,1} = \left(\hat{T}_{k+d}^{-1}\right)_{1,1} - \left(T_{k+d-1}^{-1}\right)_{1,1} + \sum_{j=k}^{k+d-2} \left[\left(T_{j+1}^{-1}\right)_{1,1} - \left(T_j^{-1}\right)_{1,1} \right].$$

From (2.13) we have

$$\|r_0\|^2 \left[\left(T_{j+1}^{-1}\right)_{1,1} - \left(T_j^{-1}\right)_{1,1} \right] = \gamma_j \|r_j\|^2 \tag{2.17}$$

so that $\hat{Q}_{k,d}$ takes the form

$$\hat{Q}_{k,d} = \|r_0\|^2 \left[\left(\hat{T}_{k+d}^{-1}\right)_{1,1} - \left(T_{k+d-1}^{-1}\right)_{1,1} \right] + \sum_{j=k}^{k+d-2} \gamma_j \|r_j\|^2.$$

Therefore, the problem of computing $\hat{Q}_{k,d}$ reduces to the problem of computing efficiently the difference

$$\|r_0\|^2 \left[\left(\hat{T}_{j+1}^{-1}\right)_{1,1} - \left(T_j^{-1}\right)_{1,1} \right], \qquad j = k + d - 1, \tag{2.18}$$

using the CG-related quantities that are available during the CG iterations. This is easy for the Gauss rule since it is given by (2.17) but, for the Gauss-Radau and Gauss-Lobatto rules we need to use results about factorizations of tridiagonal matrices. They are recalled in the next section.

3. Factorizations of tridiagonal matrices. In this section our aim is to show how to compute the quantities we need for the quadrature rules by relying only on the LDL^T factorizations of the tridiagonal matrices. For doing this we will use variants of the `qd` algorithm; see [25]. Although the matrices in our problem are positive definite in theory, it can happen during finite precision computations that the computed tridiagonal matrices are indefinite. Therefore, we will assume that our symmetric tridiagonal matrices are indefinite. However, we also assume that their LDL^T factorizations exist.

3.1. LDL^T factorization of T_k and of its extension $\hat{T}_{k+1}$. Consider a symmetric tridiagonal matrix T_k with diagonal entries α_j and subdiagonal entries $\beta_j \neq 0$,

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{3.1}$$

and its LDL^T factorization, $T_k = L_k D_k L_k^T$, denoted as

$$T_k = \begin{bmatrix} 1 & & & \\ \ell_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ell_{k-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_k \end{bmatrix} \begin{bmatrix} 1 & \ell_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \ell_{k-1} \\ & & & 1 \end{bmatrix}. \tag{3.2}$$

To compute this factorization, one can use the following recurrence relations, see, e.g., [13, p.25],

$$d_1 = \alpha_1, \quad \ell_j = \frac{\beta_j}{d_j}, \quad d_{j+1} = \alpha_{j+1} - \beta_j \ell_j, \qquad j = 1, \ldots, k-1. \tag{3.3}$$

If T_k is extended by one row and one column to the matrix $\hat{T}_{k+1}$,

$$\hat{T}_{k+1} = \begin{bmatrix} T_k & \hat{\beta}_k e_k \\ \hat{\beta}_k e_k^T & \hat{\alpha}_{k+1} \end{bmatrix}, \tag{3.4}$$

the LDL^T factorization of $\hat{T}_{k+1}$ is just a straightforward extension of the LDL^T factorization of T_k,

$$\hat{T}_{k+1} = \begin{bmatrix} 1 & & & & \\ \ell_1 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ell_{k-1} & 1 & \\ & & & \hat{\ell}_k & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & d_k & \\ & & & & \hat{d}_{k+1} \end{bmatrix} \begin{bmatrix} 1 & \ell_1 & & & \\ & \ddots & & & \\ & & \ddots & \ell_{k-1} & \\ & & & 1 & \hat{\ell}_k \\ & & & & 1 \end{bmatrix}$$

where the additional entries are given by

$$\hat{\ell}_k = \frac{\hat{\beta}_k}{d_k}, \qquad \hat{d}_{k+1} = \hat{\alpha}_{k+1} - \hat{\beta}_k \hat{\ell}_k = \hat{\alpha}_{k+1} - \frac{\hat{\beta}_k^2}{d_k}.$$

3.2. The difference between $(1,1)$ entries of $\hat{T}_{k+1}^{-1}$ and T_k^{-1}. To compute various types of quadratures, we need to compute efficiently the difference between $(1,1)$ entries of inverses of some tridiagonal matrices; see (2.18). This can be done using the following formula, see Theorem 3.9 in [13, p. 31],

$$\left(\hat{T}_{k+1}^{-1}\right)_{1,1} - \left(T_k^{-1}\right)_{1,1} = \hat{d}_{k+1}^{-1} \frac{\left(\beta_1 \ldots \beta_{k-1} \hat{\beta}_k\right)^2}{\left(d_1 \ldots d_k\right)^2} = \frac{\hat{\ell}_k^2}{\hat{d}_{k+1}} \left(\ell_1 \ldots \ell_{k-1}\right)^2, \tag{3.5}$$

where we have used that $\beta_j = \ell_j d_j$, $j = 1, \ldots, k-1$ and $\hat{\beta}_k = \hat{\ell}_k \hat{d}_k$. In other words, having the LDL^T factorizations of T_k and $\hat{T}_{k+1}$, one can compute the required difference without subtraction.

3.3. LDL^T factorization of a shifted tridiagonal matrix. We will see that for prescribing some eigenvalues we have to deal with shifted tridiagonal matrices. Let the shift μ be given such that it is different from any eigenvalue of T_k so that $T_k - \mu I$ is nonsingular. In the application μ will be smaller (resp. larger) than the smallest (resp. largest) eigenvalue of A. We denote the LDL^T factorization of $T_k - \mu I$ (when it exists) as,

$$T_k - \mu I = \bar{L}_k^{(\mu)} \bar{D}_k^{(\mu)} \left(\bar{L}_k^{(\mu)} \right)^T. \tag{3.6}$$

The entries of the LDL^T factorization of $T_k - \mu I$ are denoted with a bar, the dependence on the parameter μ is denoted by the superscript within parentheses. This factorization can be computed from scratch using (3.3) since $T_k - \mu I$ differs from T_k only in diagonal entries by

$$\bar{d}_1^{(\mu)} = \alpha_1 - \mu, \qquad \bar{\ell}_j^{(\mu)} = \frac{\beta_j}{\bar{d}_j^{(\mu)}}, \qquad \bar{d}_{j+1}^{(\mu)} = \alpha_{j+1} - \mu - \beta_j \bar{\ell}_j^{(\mu)}, \qquad j = 1, \ldots, k-1.$$

Algorithm 3 `stqds`

input μ, $d_1, \ldots, d_k, \ell_1, \ldots, \ell_{k-1}$
$\bar{d}_1^{(\mu)} = d_1 - \mu$
for $j = 1, \ldots, k-1$ **do**
  $\bar{\ell}_j^{(\mu)} = \frac{d_j \ell_j}{\bar{d}_j^{(\mu)}}$
  $\bar{d}_{j+1}^{(\mu)} = (d_{j+1} - \mu) + d_j \ell_j^2 - d_j \ell_j \bar{\ell}_j^{(\mu)}$
end for
output $\bar{d}_1^{(\mu)}, \ldots, \bar{d}_k^{(\mu)}, \bar{\ell}_1^{(\mu)}, \ldots, \bar{\ell}_{k-1}^{(\mu)}$

However, suppose now that T_k is given in the form of its LDL^T factorization, i.e. we know the entries $d_1, \ldots, d_k$ and $\ell_1, \ldots, \ell_{k-1}$ and want to compute the factorization (3.6) directly from these entries. This can be done using the `stqds` algorithm (Algorithm 3) which is a variant of the Rutishauser `qd` algorithm, see [26], [25] or [13, p. 35]. This can be further improved by introducing the difference $s_j^{(\mu)} \equiv d_j - \bar{d}_j^{(\mu)}$. It yields another version of the `stqds` algorithm called `dstqds` (Algorithm 4) which avoids some subtractions.

3.4. Rank-one modification of T_{k+1} with a prescribed eigenvalue. Given a real number μ different from any eigenvalue of T_k, our aim in this subsection is to modify the $(k+1, k+1)$st entry of T_{k+1} so that the resulting matrix $\tilde{T}_{k+1}^{(\mu)}$ defined in (2.8) has μ as a prescribed eigenvalue. In [10] it has been shown that

$$\tilde{\alpha}_{k+1}^{(\mu)} = \mu + \xi_k^{(\mu)}$$

Algorithm 4 `dstqds`

input μ, $d_1, \ldots, d_k$, $\ell_1, \ldots, \ell_{k-1}$
$s_1^{(\mu)} = \mu$
for $j = 1, \ldots, k-1$ **do**
  $\bar{d}_j^{(\mu)} = d_j - s_j^{(\mu)}$
  $\bar{\ell}_j^{(\mu)} = \frac{d_j \ell_j}{\bar{d}_j^{(\mu)}}$
  $s_{j+1}^{(\mu)} = \mu + \ell_j \bar{\ell}_j^{(\mu)} s_j^{(\mu)}$
end for
$\bar{d}_k^{(\mu)} = d_k - s_k^{(\mu)}$
output $\bar{d}_1^{(\mu)}, \ldots, \bar{d}_k^{(\mu)}, \bar{\ell}_1^{(\mu)}, \ldots, \bar{\ell}_{k-1}^{(\mu)}$

where $\xi_k^{(\mu)}$ is the last component of the solution of the tridiagonal system

$$\begin{bmatrix} \alpha_1 - \mu & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k - \mu \end{bmatrix} \begin{bmatrix} \xi_1^{(\mu)} \\ \vdots \\ \xi_{k-1}^{(\mu)} \\ \xi_k^{(\mu)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_k^2 \end{bmatrix}, \tag{3.7}$$

see also [13, pp. 88-89]. Considering the LDL^T factorization (3.6) of $T_k - \mu I$, it is easy to show that

$$\xi_k^{(\mu)} = \frac{\beta_k^2}{\bar{d}_k^{(\mu)}}, \qquad \text{i.e.} \qquad \tilde{\alpha}_{k+1}^{(\mu)} = \mu + \frac{\beta_k^2}{\bar{d}_k^{(\mu)}}.$$

Suppose now that the matrix T_{k+1} is given in the form of the LDL^T factorization (T_{k+1} is not given explicitly). We would like to modify this factorization in such a way that we obtain the LDL^T factorization of $\tilde{T}_{k+1}^{(\mu)}$. First we observe that if $T_{k+1} = L_{k+1} D_{k+1} L_{k+1}^T$, then the LDL^T factorization of $\tilde{T}_{k+1}^{(\mu)}$ is given by

$$\tilde{T}_{k+1}^{(\mu)} = L_{k+1} \begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & d_k & \\ & & & & \tilde{d}_{k+1}^{(\mu)} \end{bmatrix} L_{k+1}^T, \tag{3.8}$$

(see (3.5)) where

$$\tilde{d}_{k+1}^{(\mu)} = \tilde{\alpha}_{k+1}^{(\mu)} - \beta_k \ell_k = \mu + \xi_k^{(\mu)} - d_k \ell_k^2 = \mu + \frac{\beta_k^2}{\bar{d}_k^{(\mu)}} - d_k \ell_k^2. \tag{3.9}$$

In the following lemma we show that $\tilde{d}_{k+1}^{(\mu)} = d_{k+1} - \bar{d}_{k+1}^{(\mu)}$ where $\bar{d}_{k+1}^{(\mu)}$ is the last diagonal entry of the factorization of $T_{k+1} - \mu I$.

LEMMA 3.1. *Given μ different from any eigenvalue of T_k, consider the LDL^T factorizations of T_{k+1} and $T_{k+1} - \mu I$,*

$$T_{k+1} = L_{k+1} D_{k+1} L_{k+1}^T, \qquad T_{k+1} - \mu I = \bar{L}_{k+1}^{(\mu)} \bar{D}_{k+1}^{(\mu)} \left(\bar{L}_{k+1}^{(\mu)} \right)^T.$$

Let $\tilde{T}_{k+1}^{(\mu)}$ be the rank-one modification (2.8) of T_{k+1} such that μ is an eigenvalue of $\tilde{T}_{k+1}^{(\mu)}$ and consider its LDL^T factorization (3.8). Then it holds that

$$\tilde{d}_{k+1}^{(\mu)} = d_{k+1} - \bar{d}_{k+1}^{(\mu)}.$$

Proof. By a simple algebraic manipulation we obtain

$$d_{k+1} - \bar{d}_{k+1}^{(\mu)} = d_{k+1} - ((d_{k+1} - \mu) + d_k\ell_k^2 - d_k\ell_k\bar{\ell}_k^{(\mu)}) = (\mu - d_k\ell_k^2) + d_k\ell_k\bar{\ell}_k^{(\mu)}.$$

From (3.9) it follows that $\mu - d_k\ell_k^2 = \tilde{d}_{k+1}^{(\mu)} - \beta_k\frac{\beta_k}{\bar{d}_k^{(\mu)}}$, therefore

$$d_{k+1} - \bar{d}_{k+1}^{(\mu)} \;=\; \tilde{d}_{k+1}^{(\mu)} - \beta_k\frac{\beta_k}{\bar{d}_k^{(\mu)}} + d_k\ell_k\bar{\ell}_k^{(\mu)} \;=\; \tilde{d}_{k+1}^{(\mu)} - \ell_k d_k\bar{\ell}_k^{(\mu)} + d_k\ell_k\bar{\ell}_k^{(\mu)} \;=\; \tilde{d}_{k+1}^{(\mu)}$$

which proves the result. ∎

The formula for $\tilde{d}_{k+1}^{(\mu)}$ requires not only the (known) entry d_{k+1}, but also the (so far unknown) entry $\bar{d}_{k+1}^{(\mu)}$ from the LDL^T factorization of $T_{k+1} - \mu I$. In the following we show how to recursively compute the entry $\tilde{d}_{k+1}^{(\mu)}$ so that the LDL^T factorization of $T_{k+1} - \mu I$ need not to be computed.

Lemma 3.2. *With the notation above,*

$$\tilde{d}_1^{(\mu)} \equiv \mu, \qquad \tilde{d}_{k+1}^{(\mu)} = \mu + \ell_k^2\frac{d_k\tilde{d}_k^{(\mu)}}{d_k - \tilde{d}_k^{(\mu)}} \quad \textit{for} \quad k \geq 1. \tag{3.10}$$

Proof. From Lemma 3.1 it follows that $\tilde{d}_{k+1}^{(\mu)}$ is nothing but the difference $s_{k+1}^{(\mu)} = d_{k+1} - \bar{d}_{k+1}^{(\mu)}$ introduced in Algorithm 4. Using

$$s_{k+1}^{(\mu)} - \mu = \ell_k\bar{\ell}_k^{(\mu)}s_k^{(\mu)} = \ell_k\frac{d_k\ell_k}{\bar{d}_k^{(\mu)}}s_k^{(\mu)} = \ell_k\frac{d_k\ell_k}{d_k - s_k^{(\mu)}}s_k^{(\mu)}$$

and $\tilde{d}_k^{(\mu)} = s_k^{(\mu)}$ we obtain the formula (3.10). ∎

Formula (3.10) will be used to compute the inverse of $\tilde{d}_{k+1}^{(\mu)}$.

3.5. Rank-two modification of T_{k+1} with two prescribed eigenvalues. Given two numbers μ and η different from any eigenvalue of T_k, we would like to find a rank-two modification of the matrix T_{k+1} so that the matrix $\tilde{T}_{k+1}^{(\mu,\eta)}$ defined in (2.9) has μ and η as prescribed eigenvalues. In [10] it has been shown that $\tilde{\alpha}_{k+1}^{(\mu,\eta)}$ and $\tilde{\beta}_k^{(\mu,\eta)}$ satisfy

$$\tilde{\alpha}_{k+1}^{(\mu,\eta)} \;=\; \mu + \left(\tilde{\beta}_k^{(\mu,\eta)}\right)^2\zeta_k^{(\mu)}, \qquad \tilde{\alpha}_{k+1}^{(\mu,\eta)} \;=\; \eta + \left(\tilde{\beta}_k^{(\mu,\eta)}\right)^2\zeta_k^{(\eta)},$$

where $\zeta_k^{(\mu)}$, respectively $\zeta_k^{(\eta)}$, is the last component of the tridiagonal system

$$(T_k - \mu I)\zeta^{(\mu)} = e_k, \qquad \zeta^{(\mu)} \equiv \left[\zeta_1^{(\mu)}, \ldots, \zeta_k^{(\mu)}\right]^T,$$

respectively,

$$(T_k - \eta I)\zeta^{(\eta)} = e_k, \qquad \zeta^{(\eta)} \equiv \left[\zeta_1^{(\eta)}, \ldots, \zeta_k^{(\eta)}\right]^T.$$

Summarizing, we first solve systems $(T_k - \mu I)\zeta^{(\mu)} = e_k$, $(T_k - \eta I)\zeta^{(\eta)} = e_k$ and then we obtain $\tilde{\alpha}_{k+1}^{(\mu,\eta)}$ and $\left(\tilde{\beta}_k^{(\mu,\eta)}\right)^2$ as the solution of the 2×2 linear system

$$\begin{bmatrix} 1 & -\xi_k^{(\mu)} \\ 1 & -\xi_k^{(\eta)} \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_{k+1} \\ \left(\tilde{\beta}_k^{(\mu,\eta)}\right)^2 \end{bmatrix} = \begin{bmatrix} \mu \\ \eta \end{bmatrix}.$$

We are now interested in the LDL^T factorization of the matrix $\tilde{T}_{k+1}^{(\mu,\eta)}$, see (2.9),

$$\begin{bmatrix} 1 & & & & \\ \ell_1 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ell_{k-1} & 1 & \\ & & & \tilde{\ell}_k^{(\mu,\eta)} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & & d_k & \\ & & & & \tilde{d}_{k+1}^{(\mu,\eta)} \end{bmatrix} \begin{bmatrix} 1 & \ell_1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ell_{k-1} & \\ & & & 1 & \tilde{\ell}_k^{(\mu,\eta)} \\ & & & & 1 \end{bmatrix}. \tag{3.11}$$

In the following lemma we express $\tilde{\ell}_k^{(\mu,\eta)}$ and $\tilde{d}_{k+1}^{(\mu,\eta)}$ using the entries of the LDL^T factorizations of $T_k - \mu I$ and $T_k - \eta I$.

LEMMA 3.3. *The entries $\tilde{\ell}_k^{(\mu,\eta)}$ and $\tilde{d}_{k+1}^{(\mu,\eta)}$ from (3.11) can be computed using the following formulas,*

$$\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2 = \frac{\bar{d}_k^{(\mu)}\bar{d}_k^{(\eta)}}{d_k^2}\frac{\eta - \mu}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}}, \quad \tilde{d}_{k+1}^{(\mu,\eta)} = \frac{\eta\bar{d}_k^{(\eta)} - \mu\bar{d}_k^{(\mu)}}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}} - d_k\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2. \tag{3.12}$$

Proof. For simplicity of notation in this proof, we will omit the (μ, η) upper indices. From (3.11), it follows that $\tilde{\ell}_k$ and $\tilde{d}_{k+1}$ satisfy

$$\tilde{\ell}_k^2 = \frac{\tilde{\beta}_k^2}{d_k^2}, \qquad \tilde{d}_{k+1} = \tilde{\alpha}_{k+1} - \tilde{\beta}_k\tilde{\ell}_k = \tilde{\alpha}_{k+1} - \frac{\tilde{\beta}_k^2}{d_k}.$$

Therefore,

$$\begin{bmatrix} \tilde{d}_{k+1} \\ \tilde{\ell}_k^2 \end{bmatrix} = \begin{bmatrix} 1 & -d_k^{-1} \\ 0 & d_k^{-2} \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_{k+1} \\ \tilde{\beta}_k^2 \end{bmatrix}, \qquad \begin{bmatrix} \tilde{\alpha}_{k+1} \\ \tilde{\beta}_k^2 \end{bmatrix} = \begin{bmatrix} 1 & d_k \\ 0 & d_k^2 \end{bmatrix} \begin{bmatrix} \tilde{d}_{k+1} \\ \tilde{\ell}_k^2 \end{bmatrix}$$

and $\tilde{\ell}_k^2$ and $\tilde{d}_{k+1}$ solve the system

$$\begin{bmatrix} \mu \\ \eta \end{bmatrix} = \begin{bmatrix} 1 & -\zeta_k^{(\mu)} \\ 1 & -\zeta_k^{(\eta)} \end{bmatrix} \begin{bmatrix} 1 & d_k \\ 0 & d_k^2 \end{bmatrix} \begin{bmatrix} \tilde{d}_{k+1} \\ \tilde{\ell}_k^2 \end{bmatrix} = \begin{bmatrix} 1 & d_k - d_k^2\zeta_k^{(\mu)} \\ 1 & d_k - d_k^2\zeta_k^{(\eta)} \end{bmatrix} \begin{bmatrix} \tilde{d}_{k+1} \\ \tilde{\ell}_k^2 \end{bmatrix}.$$

Using Cramer's rule we obtain

$$\det \begin{bmatrix} 1 & d_k - d_k^2\zeta_k^{(\mu)} \\ 1 & d_k - d_k^2\zeta_k^{(\eta)} \end{bmatrix} = d_k^2\left(\zeta_k^{(\mu)} - \zeta_k^{(\eta)}\right),$$

$$\det\begin{bmatrix} \mu & d_k - d_k^2\zeta_k^{(\mu)} \\ \eta & d_k - d_k^2\zeta_k^{(\eta)} \end{bmatrix} = d_k(\mu-\eta) - d_k^2\left(\mu\zeta_k^{(\eta)} - \eta\zeta_k^{(\mu)}\right), \qquad \det\begin{bmatrix} 1 & \mu \\ 1 & \eta \end{bmatrix} = \eta - \mu,$$

and, therefore

$$\tilde{\ell}_k^2 = \frac{\eta-\mu}{d_k^2\left(\zeta_k^{(\mu)} - \zeta_k^{(\eta)}\right)},$$
$$\tilde{d}_{k+1} = \frac{\mu - \eta - d_k\left(\mu\zeta_k^{(\eta)} - \eta\zeta_k^{(\mu)}\right)}{d_k\left(\zeta_k^{(\mu)} - \zeta_k^{(\eta)}\right)} = \frac{\eta\zeta_k^{(\mu)} - \mu\zeta_k^{(\eta)}}{\zeta_k^{(\mu)} - \zeta_k^{(\eta)}} - d_k\tilde{\ell}_k^2.$$

Using

$$\zeta_k^{(\eta)} = \frac{1}{\bar{d}_k^{(\eta)}}, \qquad \zeta_k^{(\mu)} = \frac{1}{\bar{d}_k^{(\mu)}}$$

we obtain

$$\tilde{\ell}_k^2 = \frac{\eta-\mu}{d_k^2\left(\frac{1}{\bar{d}_k^{(\mu)}} - \frac{1}{\bar{d}_k^{(\eta)}}\right)} = \frac{\bar{d}_k^{(\mu)}\bar{d}_k^{(\eta)}}{d_k^2}\frac{\eta-\mu}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}},$$
$$\tilde{d}_{k+1} = \frac{\frac{\eta}{\bar{d}_k^{(\mu)}} - \frac{\mu}{\bar{d}_k^{(\eta)}}}{\frac{1}{\bar{d}_k^{(\mu)}} - \frac{1}{\bar{d}_k^{(\eta)}}} - d_k\tilde{\ell}_k^2 = \frac{\eta\bar{d}_k^{(\eta)} - \mu\bar{d}_k^{(\mu)}}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}} - d_k\tilde{\ell}_k^2,$$

which completes the proof. □

Using the formula (3.10) we can update the entries $\tilde{d}_k^{(\mu)}$ and $\tilde{d}_k^{(\eta)}$. Then, we can use the relations

$$\bar{d}_k^{(\mu)} = d_k - \tilde{d}_k^{(\mu)}, \qquad \bar{d}_k^{(\eta)} = d_k - \tilde{d}_k^{(\eta)}$$

and compute $\tilde{\ell}_k^{(\mu,\eta)}$ and $\tilde{d}_{k+1}^{(\mu,\eta)}$ using $\tilde{d}_k^{(\mu)}$ and $\tilde{d}_k^{(\eta)}$, as it is shown the following lemma.

LEMMA 3.4. *The entries $\tilde{\ell}_k^{(\mu,\eta)}$ and $\tilde{d}_{k+1}^{(\mu,\eta)}$ from (3.11) can be computed using the following formulas,*

$$\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2 = \frac{\left(d_k - \tilde{d}_k^{(\mu)}\right)\left(d_k - \tilde{d}_k^{(\eta)}\right)(\eta-\mu)}{\left(\tilde{d}_k^{(\mu)} - \tilde{d}_k^{(\eta)}\right)d_k^2}, \tag{3.13}$$

$$\tilde{d}_{k+1}^{(\mu,\eta)} = \frac{d_k(\eta-\mu) + \mu\tilde{d}_k^{(\mu)} - \eta\tilde{d}_k^{(\eta)}}{\tilde{d}_k^{(\mu)} - \tilde{d}_k^{(\eta)}} - d_k\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2. \tag{3.14}$$

Proof. Using $\bar{d}_k^{(\mu)} = d_k - \tilde{d}_k^{(\mu)}$, $\bar{d}_k^{(\eta)} = d_k - \tilde{d}_k^{(\eta)}$ we get

$$\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)} = d_k - \tilde{d}_k^{(\eta)} - (d_k - \tilde{d}_k^{(\mu)}) = \tilde{d}_k^{(\mu)} - \tilde{d}_k^{(\eta)},$$
$$\eta\bar{d}_k^{(\eta)} - \mu\bar{d}_k^{(\mu)} = \eta(d_k - \tilde{d}_k^{(\eta)}) - \mu(d_k - \tilde{d}_k^{(\mu)}) = d_k(\eta-\mu) + \mu\tilde{d}_k^{(\mu)} - \eta\tilde{d}_k^{(\eta)}$$

and by substituting into the formulas (3.12) we obtain

$$\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2 = \frac{\bar{d}_k^{(\mu)}\bar{d}_k^{(\eta)}}{d_k^2}\frac{\eta-\mu}{\bar{d}_k^{(\eta)}-\bar{d}_k^{(\mu)}} = \frac{\left(d_k-\tilde{d}_k^{(\mu)}\right)\left(d_k-\tilde{d}_k^{(\eta)}\right)(\eta-\mu)}{\left(\tilde{d}_k^{(\mu)}-\tilde{d}_k^{(\eta)}\right)d_k^2},$$

$$\tilde{d}_{k+1}^{(\mu,\eta)} = \frac{\eta\bar{d}_k^{(\eta)}-\mu\bar{d}_k^{(\mu)}}{\bar{d}_k^{(\eta)}-\bar{d}_k^{(\mu)}} - d_k\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2 = \frac{d_k(\eta-\mu)+\mu\tilde{d}_k^{(\mu)}-\eta\tilde{d}_k^{(\eta)}}{\tilde{d}_k^{(\mu)}-\tilde{d}_k^{(\eta)}} - d_k\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2,$$

which completes the proof. □

In the formulas that we will use later, we need the ratio

$$\frac{\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2}{\tilde{d}_{k+1}^{(\mu,\eta)}}$$

rather than the values $\tilde{\ell}_k^{(\mu,\eta)}$ and $\tilde{d}_{k+1}^{(\mu,\eta)}$. The following lemma shows the formula for computing this ratio.

LEMMA 3.5. *It holds that*

$$\frac{\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2}{\tilde{d}_{k+1}^{(\mu,\eta)}} = \frac{\left(\left(\tilde{d}_k^{(\eta)}\right)^{-1}-d_k^{-1}\right)\left(\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}\right)(\eta-\mu)}{\eta\left(\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}\right)-\mu\left(\left(\tilde{d}_k^{(\eta)}\right)^{-1}-d_k^{-1}\right)}. \tag{3.15}$$

Proof. Using formulas (3.13) and (3.14) and simple algebraic manipulations we get

$$\begin{aligned}
\left(\frac{\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2}{\tilde{d}_{k+1}^{(\mu,\eta)}}\right)^{-1} &= \frac{d_k(\eta-\mu)+\mu\tilde{d}_k^{(\mu)}-\eta\tilde{d}_k^{(\eta)}}{\tilde{d}_k^{(\mu)}-\tilde{d}_k^{(\eta)}}\left(\tilde{\ell}_k^{(\mu,\eta)}\right)^{-2} - d_k \\
&= \frac{d_k(\eta-\mu)+\mu\tilde{d}_k^{(\mu)}-\eta\tilde{d}_k^{(\eta)}}{\tilde{d}_k^{(\mu)}-\tilde{d}_k^{(\eta)}}\frac{\left(\tilde{d}_k^{(\mu)}-\tilde{d}_k^{(\eta)}\right)d_k^2}{\left(d_k-\tilde{d}_k^{(\mu)}\right)\left(d_k-\tilde{d}_k^{(\eta)}\right)(\eta-\mu)} - d_k \\
&= d_k\left[d_k\frac{d_k(\eta-\mu)+\mu\tilde{d}_k^{(\mu)}-\eta\tilde{d}_k^{(\eta)}}{\left(d_k-\tilde{d}_k^{(\mu)}\right)\left(d_k-\tilde{d}_k^{(\eta)}\right)(\eta-\mu)} - 1\right] \\
&= \frac{d_k^2\left(\eta\tilde{d}_k^{(\mu)}-\mu\tilde{d}_k^{(\eta)}\right)-d_k\left(\tilde{d}_k^{(\mu)}\tilde{d}_k^{(\eta)}\right)(\eta-\mu)}{\left(d_k-\tilde{d}_k^{(\mu)}\right)\left(d_k-\tilde{d}_k^{(\eta)}\right)(\eta-\mu)} \\
&= \frac{\eta\left(\tilde{d}_k^{(\eta)}\right)^{-1}-\mu\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}(\eta-\mu)}{\left(\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}\right)\left(\left(\tilde{d}_k^{(\eta)}\right)^{-1}-d_k^{-1}\right)(\eta-\mu)} \\
&= \frac{\eta\left(\left(\tilde{d}_k^{(\eta)}\right)^{-1}-d_k^{-1}\right)-\mu\left(\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}\right)}{\left(\left(\tilde{d}_k^{(\mu)}\right)^{-1}-d_k^{-1}\right)\left(\left(\tilde{d}_k^{(\eta)}\right)^{-1}-d_k^{-1}\right)(\eta-\mu)}
\end{aligned}$$

which completes the proof. □

3.6. Another rank-two modification of T_{k+1}. The last modification of T_{k+1} that we consider, is to replace β_k in T_{k+1} by $c\,\beta_k$ where c is a given constant. The corresponding Jacobi matrix $\hat{T}_{k+1}^{(c)}$ has the same LDL^T factorization as T_{k+1} up to

$$\left(\hat{\ell}_k^{(c)}\right)^2 = \frac{c^2\beta_k^2}{d_k^2} = c^2\,\ell_k^2, \qquad \hat{d}_{k+1}^{(c)} = \alpha_{k+1} - c^2\,d_k\ell_k^2 = d_{k+1} + (1-c^2)\,d_k\ell_k^2\,,$$

and, therefore,

$$\frac{\left(\hat{\ell}_k^{(c)}\right)^2}{\hat{d}_{k+1}^{(c)}} = \frac{c^2}{d_{k+1} + (1-c^2)\,d_k\ell_k^2}\,\ell_k^2\,. \tag{3.16}$$

4. Algorithms. In this section we use the results from the previous sections for the tridiagonal matrices resulting from the Lanczos and CG algorithms. This will allow us to obtain simple formulas for computing lower and upper bounds for the A-norm of the error. Matching the LDL^T of T_k in (3.2) and (2.1), we obtain

$$\ell_j^2 = \delta_j, \qquad d_j = \gamma_{j-1}^{-1}.$$

Given two prescribed nodes μ and η, let us now consider various modifications of the matrix T_{k+1} and define

$$\tilde{\gamma}_k^{(\mu)} \equiv \left(\tilde{d}_{k+1}^{(\mu)}\right)^{-1}, \qquad \tilde{\gamma}_k^{(\eta)} \equiv \left(\tilde{d}_{k+1}^{(\eta)}\right)^{-1}, \qquad \tilde{\gamma}_k^{(\mu,\eta)} \equiv \left(\tilde{\ell}_k^{(\mu,\eta)}\right)^2 \left(\tilde{d}_{k+1}^{(\mu,\eta)}\right)^{-1}.$$

Using (3.10) and (3.15) we get the updating formulas

$$\begin{aligned}
\tilde{\gamma}_0^{(\mu)} &= \frac{1}{\mu}, \qquad \tilde{\gamma}_k^{(\mu)} = \frac{\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}}{\mu\left(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}\right) + \delta_k}\,,\\
\tilde{\gamma}_0^{(\eta)} &= \frac{1}{\eta}, \qquad \tilde{\gamma}_k^{(\eta)} = \frac{\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1}}{\eta\left(\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1}\right) + \delta_k}\,,\\
&\qquad\qquad \tilde{\gamma}_k^{(\mu,\eta)} = \frac{\left(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}\right)\left(\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1}\right)(\eta-\mu)}{\eta\left(\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1}\right) - \mu\left(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}\right)}.
\end{aligned}$$

Using (3.5) and

$$\|r_0\|^2\,(\ell_1\ldots\ell_{k-1})^2 = \|r_0\|^2\delta_1\ldots\delta_{k-1} = \|r_0\|^2\frac{\|r_1\|^2}{\|r_0\|^2}\ldots\frac{\|r_{k-1}\|^2}{\|r_{k-1}\|^2} = \|r_{k-1}\|^2,$$

one obtains

$$\|r_0\|^2\left(\left[\left(\tilde{T}_{k+1}^{(\mu)}\right)^{-1}\right]_{1,1} - \left(T_k^{-1}\right)_{1,1}\right) = \frac{\ell_k^2}{\tilde{d}_{k+1}^{(\mu)}}\|r_{k-1}\|^2 = \frac{\|r_k\|^2}{\tilde{d}_{k+1}^{(\mu)}}.$$

We can now compute the quantities of the form (2.18),

$$g_k \equiv \|r_0\|^2\left(\left(T_{k+1}^{-1}\right)_{1,1} - \left(T_k^{-1}\right)_{1,1}\right) = \gamma_k\|r_k\|^2,$$

$$g_k^{(\mu)} \equiv \|r_0\|^2 \left(\left[\left(\tilde{T}_{k+1}^{(\mu)} \right)^{-1} \right]_{1,1} - \left(T_k^{-1} \right)_{1,1} \right) = \tilde{\gamma}_k^{(\mu)} \|r_k\|^2,$$
$$g_k^{(\eta)} \equiv \|r_0\|^2 \left(\left[\left(\tilde{T}_{k+1}^{(\eta)} \right)^{-1} \right]_{1,1} - \left(T_k^{-1} \right)_{1,1} \right) = \tilde{\gamma}_k^{(\eta)} \|r_k\|^2,$$
$$g_k^{(\mu,\eta)} \equiv \|r_0\|^2 \left(\left[\left(\tilde{T}_{k+1}^{(\mu,\eta)} \right)^{-1} \right]_{1,1} - \left(T_k^{-1} \right)_{1,1} \right) = \tilde{\gamma}_k^{(\mu,\eta)} \|r_{k-1}\|^2.$$

Hence, the quantities for the three different rules are given almost in the same form. The index of the residual differs for the Gauss-Lobatto rule because the term $(\tilde{\ell}_k^{(\mu,\eta)})^2$ is incorporated in $\tilde{\gamma}_k^{(\mu,\eta)}$. Using the updating formulas for the coefficients $\tilde{\gamma}_k^{(\mu)}$, $\tilde{\gamma}_k^{(\eta)}$ and $\tilde{\gamma}_k^{(\mu,\eta)}$ we can derive updating formulas for $g_k^{(\mu)}$, $g_k^{(\eta)}$ and $g_k^{(\mu,\eta)}$ such that the coefficients $\tilde{\gamma}_k^{(\mu)}$, $\tilde{\gamma}_k^{(\eta)}$ and $\tilde{\gamma}_k^{(\mu,\eta)}$ need not to be computed. Since

$$\begin{aligned}\tilde{\gamma}_k^{(\mu)} &= \frac{\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}}{\mu(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1}) + \delta_k} = \frac{\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\mu)} - \|r_{k-1}\|^2 \gamma_{k-1}}{\mu(\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\mu)} - \|r_{k-1}\|^2 \gamma_{k-1}) + \|r_k\|^2} \\ &= \frac{g_{k-1}^{(\mu)} - g_{k-1}}{\mu(g_{k-1}^{(\mu)} - g_{k-1}) + \|r_k\|^2},\end{aligned}$$

the formulas for $g_k^{(\mu)}$ and $g_k^{(\eta)}$ can be written as

$$g_0^{(\mu)} = \frac{\|r_0\|^2}{\mu}, \qquad g_0^{(\eta)} = \frac{\|r_0\|^2}{\eta}, \tag{4.1}$$

$$g_k^{(\mu)} = \|r_k\|^2 \frac{g_{k-1}^{(\mu)} - g_{k-1}}{\mu(g_{k-1}^{(\mu)} - g_{k-1}) + \|r_k\|^2}, \qquad g_k^{(\eta)} = \|r_k\|^2 \frac{g_{k-1}^{(\eta)} - g_{k-1}}{\mu(g_{k-1}^{(\eta)} - g_{k-1}) + \|r_k\|^2}.$$

The formula for $g_k^{(\mu,\eta)} = \tilde{\gamma}_k^{(\mu,\eta)} \|r_{k-1}\|^2$ takes the form

$$\begin{aligned}g_k^{(\mu,\eta)} = \tilde{\gamma}_k^{(\mu,\eta)} \|r_{k-1}\|^2 &= \|r_{k-1}\|^2 \frac{\left(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1} \right) \left(\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1} \right) (\eta - \mu)}{\eta \left(\tilde{\gamma}_{k-1}^{(\eta)} - \gamma_{k-1} \right) - \mu \left(\tilde{\gamma}_{k-1}^{(\mu)} - \gamma_{k-1} \right)} \\ &= \frac{\left(\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\mu)} - \|r_{k-1}\|^2 \gamma_{k-1} \right) \left(\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\eta)} - \|r_{k-1}\|^2 \gamma_{k-1} \right) (\eta - \mu)}{\eta \left(\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\eta)} - \|r_{k-1}\|^2 \gamma_{k-1} \right) - \mu \left(\|r_{k-1}\|^2 \tilde{\gamma}_{k-1}^{(\mu)} - \|r_{k-1}\|^2 \gamma_{k-1} \right)} \\ &= \frac{\left(g_{k-1}^{(\mu)} - g_{k-1} \right) \left(g_{k-1}^{(\eta)} - g_{k-1} \right) (\eta - \mu)}{\eta \left(g_{k-1}^{(\eta)} - g_{k-1} \right) - \mu \left(g_{k-1}^{(\mu)} - g_{k-1} \right)}.\end{aligned}$$

Summarizing, starting with the formulas (4.1) we obtain for $k = 1, \ldots$ updating formulas taking the simple following form,

$$g_{k-1} = \gamma_{k-1} \|r_{k-1}\|^2, \qquad \Delta_{k-1}^{(\mu)} \equiv g_{k-1}^{(\mu)} - g_{k-1}, \qquad \Delta_{k-1}^{(\eta)} \equiv g_{k-1}^{(\eta)} - g_{k-1}, \tag{4.2}$$

and

$$g_k^{(\mu)} = \|r_k\|^2 \frac{\Delta_{k-1}^{(\mu)}}{\mu\Delta_{k-1}^{(\mu)} + \|r_k\|^2}, \quad g_k^{(\eta)} = \|r_k\|^2 \frac{\Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} + \|r_k\|^2}, \tag{4.3}$$

$$g_k^{(\mu,\eta)} = (\eta - \mu) \frac{\Delta_{k-1}^{(\mu)} \Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} - \mu\Delta_{k-1}^{(\mu)}}. \tag{4.4}$$

Now we have all the needed material to compute bounds. We distinguish three parts in the algorithm for running CG and obtaining bounds for the A-norm of the error.

1. The first part is simply the **CG-iteration** which computes two scalars and updates the vectors,

$$\begin{aligned} \gamma_{k-1} &= \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}, \\ x_k &= x_{k-1} + \gamma_{k-1} p_{k-1}, \\ r_k &= r_{k-1} - \gamma_{k-1} A p_{k-1}, \\ \delta_k &= \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}, \\ p_k &= r_k + \delta_k p_{k-1}. \end{aligned}$$

2. The second part is called the **Quadrature part**. It computes the quantities g_{k-1}, $g_k^{(\mu)}$, $g_k^{(\eta)}$ and $g_k^{(\mu,\eta)}$ using the formulas (4.2), (4.3), and (4.4) if we are interested in computing the Gauss, Gauss-Radau and Gauss-Lobatto bounds.

3. The third part is called the **Estimates part** for iteration $k-d$. In this paper, we use the following way of constructing the estimates from g_{k-1}, $g_k^{(\mu)}$, $g_k^{(\eta)}$ and $g_k^{(\mu,\eta)}$. If $k > d$, then compute

$$\begin{aligned} Q_{k-d,d} &= \sum_{j=k-d+1}^{k} g_j, \\ E_{k-d} &= \sqrt{Q_{k-d,d}}, \qquad E_{k-d}^{(\mu)} = \sqrt{Q_{k-d,d} + g_k^{(\mu)}}, \\ E_{k-d}^{(\eta)} &= \sqrt{Q_{k-d,d} + g_k^{(\eta)}}, \qquad E_{k-d}^{(\mu,\eta)} = \sqrt{Q_{k-d,d} + g_k^{(\mu,\eta)}} \end{aligned}$$

E_{k-d} is the Gauss lower bound. If $\mu < \lambda_1$ (resp. $\eta > \lambda_N$) $E_{k-d}^{(\mu)}$ (resp. $E_{k-d}^{(\eta)}$) is the Gauss-Radau upper (resp. lower) bound and $E_{k-d}^{(\mu,\eta)}$ is the Gauss-Lobatto upper bound. Note that d is a given positive integer indicating how many steps of CG should be precomputed to have the estimate at iteration $k - d$.

Algorithm 5 recalls the CGQL algorithm described in [21] and [13]. Algorithm 6 is the new version using the simpler formulas derived in this paper. We denote it as CGQ (in reference to CGQL, dropping the L because we do not use the Lanczos coefficients any longer). We see that computing the Gauss-Radau upper bound is almost as simple as computing the Gauss lower bound provided we have a $\mu < \lambda_1$.

In practical computations we usually only use the lower bound based on Gauss quadrature and the upper bound based on Gauss-Radau when a lower bound of the

Algorithm 5 *CGQL* (Conjugate Gradients and Quadrature via Lanczos coefficients)

input A, b, x_0, μ
$r_0 = b - Ax_0$, $p_0 = r_0$
$\delta_0 = 0$, $\gamma_{-1} = 1$, $c_1 = 1$, $\beta_0 = 0$, $d_0 = 1$, $\tilde{\alpha}_1^{(\mu)} = \mu$, $\tilde{\alpha}_1^{(\eta)} = \eta$
for $k = 1, \ldots,$ until convergence **do**
CG-iteration (k)

$$
\begin{aligned}
\alpha_k &= \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}},\ \beta_k^2 = \frac{\delta_k}{\gamma_{k-1}^2} \\
d_k &= \alpha_k - \frac{\beta_{k-1}^2}{d_{k-1}},\ g_k = \|r_0\|^2 \frac{c_k^2}{d_k}, \\
\bar{d}_k^{(\mu)} &= \alpha_k - \tilde{\alpha}_k^{(\mu)},\ \tilde{\alpha}_{k+1}^{(\mu)} = \mu + \frac{\beta_k^2}{\bar{d}_k^{(\mu)}},\ g_k^{(\mu)} = \|r_0\|^2 \frac{\beta_k^2 c_k^2}{d_k \left(\tilde{\alpha}_{k+1}^{(\mu)} d_k - \beta_k^2\right)} \\
\bar{d}_k^{(\eta)} &= \alpha_k - \tilde{\alpha}_k^{(\eta)},\ \tilde{\alpha}_{k+1}^{(\eta)} = \eta + \frac{\beta_k^2}{\bar{d}_k^{(\eta)}},\ g_k^{(\eta)} = \|r_0\|^2 \frac{\beta_k^2 c_k^2}{d_k \left(\tilde{\alpha}_{k+1}^{(\eta)} d_k - \beta_k^2\right)} \\
\tilde{\alpha}_{k+1}^{(\mu,\eta)} &= \frac{\bar{d}_k^{(\mu)} \bar{d}_k^{(\eta)}}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}} \left(\frac{\eta}{\bar{d}_k^{(\mu)}} - \frac{\mu}{\bar{d}_k^{(\eta)}}\right),\ \left[\tilde{\beta}_k^{(\mu,\eta)}\right]^2 = \frac{\bar{d}_k^{(\mu)} \bar{d}_k^{(\eta)}}{\bar{d}_k^{(\eta)} - \bar{d}_k^{(\mu)}} (\eta - \mu) \\
g_k^{(\mu,\eta)} &= \|r_0\|^2 \frac{\left[\tilde{\beta}_k^{(\mu,\eta)}\right]^2 c_k^2}{d_k \left(\tilde{\alpha}_{k+1}^{(\mu,\eta)} d_k - \left[\tilde{\beta}_k^{(\mu,\eta)}\right]^2\right)} \\
c_{k+1}^2 &= \frac{\beta_k^2 c_k^2}{d_k^2}
\end{aligned}
$$

Estimates(k,d)
end for

smallest eigenvalue of A is available. In that case we only need to compute g_k and $g_k^{(\mu)}$ using the formulas from Algorithm 6.

Note that in the same way we can also obtain formulas for the anti-Gauss quadrature rule [20, 4]. Defining

$$\hat{g}_k^{(c)} \equiv \|r_0\|^2 \left(\left[\left(\hat{T}_{k+1}^{(c)} \right)^{-1} \right]_{1,1} - \left(T_k^{-1} \right)_{1,1} \right)$$

and using the formula (3.16) we obtain

$$\hat{g}_k^{(c)} = c^2 \frac{g_k g_{k-1}}{g_{k-1} + (1 - c^2) g_k}\,. \tag{4.5}$$

The anti-Gauss quadrature estimate can be then computed analogously,

$$\hat{E}_{k-d}^{(c)} = \sqrt{Q_{k-d,d} + \hat{g}_k^{(c)}}\,.$$

Algorithm 6 *CGQ* (Conjugate Gradients and Quadrature)

input A, b, x_0, μ, η
$r_0 = b - Ax_0$, $p_0 = r_0$
$g_0^{(\mu)} = \frac{\|r_0\|^2}{\mu}$, $g_0^{(\eta)} = \frac{\|r_0\|^2}{\eta}$
for $k = 1, \ldots,$ until convergence **do**
CG-iteration(k)

$$
\begin{aligned}
g_{k-1} &= \gamma_{k-1}\|r_{k-1}\|^2, \\
\Delta_{k-1}^{(\mu)} &= g_{k-1}^{(\mu)} - g_{k-1}\,, \qquad g_k^{(\mu)} = \frac{\|r_k\|^2\Delta_{k-1}^{(\mu)}}{\mu\Delta_{k-1}^{(\mu)} + \|r_k\|^2} \\
\Delta_{k-1}^{(\eta)} &= g_{k-1}^{(\eta)} - g_{k-1}\,, \qquad g_k^{(\eta)} = \frac{\|r_k\|^2\Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} + \|r_k\|^2} \\
g_k^{(\mu,\eta)} &= (\eta-\mu)\frac{\Delta_{k-1}^{(\mu)}\Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} - \mu\Delta_{k-1}^{(\mu)}}\,.
\end{aligned}
$$

Estimates(k,d)
end for

5. Error estimation in preconditioned CG. In the standard view of preconditioning, the CG method is thought of as being applied to a "preconditioned" system

$$\hat{A}\hat{x} = \hat{b}, \qquad \hat{A} = L^{-1}AL^{-T}, \quad \hat{b} = L^{-1}b, \tag{5.1}$$

where L represents a nonsingular (eventually lower triangular) matrix. Denoting the corresponding CG coefficients and vectors with hat and defining

$$x_k \equiv L^{-T}\hat{x}_k, \quad r_k \equiv L\,\hat{r}_k, \quad p_k \equiv L^{-T}\hat{p}_k, \quad z_k \equiv L^{-T}L^{-1}r_k \equiv P^{-1}r_k,$$

(here x_k and r_k represent the approximate solution and residual for the original problem $Ax = b$), we obtain the standard version of the preconditioned CG (PCG) method which involves only $P = LL^T$; for more details see, e.g. [22] or [29].

Since

$$
\begin{aligned}
\|\hat{r}_k\|^2 &= r_k^T L^{-T}L^{-1}r_k = r_k^T P^{-1}r_k = (r_k, z_k)\,, \\
\|\hat{x} - \hat{x}_k\|_{\hat{A}} &= (L^Tx - L^Tx_k)^T L^{-1}AL^{-T}(L^Tx - L^Tx_k) = \|x - x_k\|_A^2,
\end{aligned}
$$

the A-norm of the error in PCG can be estimated similarly as in ordinary CG. One can compute the quadratures-based estimates of the A-norm of the error using the PCG coefficients $\hat{\gamma}_{k-1}$ and inner products (r_k, z_k) (instead of using $\|\hat{r}_k\|^2$). The resulting Algorithm 7 is called PCGQ.

6. Numerical experiments. We present two examples where we demonstrate the effectivity of the new formula

$$g_k^{(\mu)} = \frac{\|r_k\|^2\left(g_{k-1}^{(\mu)} - g_{k-1}\right)}{\mu\left(g_{k-1}^{(\mu)} - g_{k-1}\right) + \|r_k\|^2}$$

Algorithm 7 Preconditioned CGQ (PCGQ) algorithm

input A, b, x_0, P, μ, η
$r_0 = b - Ax_0$, $z_0 = P^{-1}r_0$, $p_0 = z_0$
$g_0^{(\mu)} = \frac{(r_0,z_0)}{\mu}$, $g_0^{(\eta)} = \frac{(r_0,z_0)}{\eta}$
for $k = 1, \ldots, n$ until convergence **do**
$\hat{\gamma}_{k-1} = \frac{z_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$
$x_k = x_{k-1} + \hat{\gamma}_{k-1} p_{k-1}$
$r_k = r_{k-1} - \hat{\gamma}_{k-1} A p_{k-1}$
$z_k = P^{-1} r_k$
$\hat{\delta}_k = \frac{z_k^T r_k}{z_{k-1}^T r_{k-1}}$
$p_k = z_k + \hat{\delta}_k p_{k-1}$

$$
\begin{aligned}
g_{k-1} &= \hat{\gamma}_{k-1}(r_{k-1}, z_{k-1}), \\
\Delta_{k-1}^{(\mu)} &= g_{k-1}^{(\mu)} - g_{k-1}\,, \qquad g_k^{(\mu)} = \frac{(r_k, z_k)\Delta_{k-1}^{(\mu)}}{\mu\Delta_{k-1}^{(\mu)} + (r_k, z_k)} \\
\Delta_{k-1}^{(\eta)} &= g_{k-1}^{(\eta)} - g_{k-1}\,, \qquad g_k^{(\eta)} = \frac{(r_k, z_k)\Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} + (r_k, z_k)} \\
g_k^{(\mu,\eta)} &= (\eta - \mu)\frac{\Delta_{k-1}^{(\mu)}\Delta_{k-1}^{(\eta)}}{\eta\Delta_{k-1}^{(\eta)} - \mu\Delta_{k-1}^{(\mu)}}\,.
\end{aligned}
$$

Estimates(k,d)
end for

that is key for computing the Gauss-Radau quadrature estimate (GR-estimate). Both examples are chosen such that many iterations are necessary for CG (or PCG) to converge, and such that the influence of rounding errors is substantial (in both examples we observe a significant delay of convergence). The aim of numerical experiments is not to focus on the question of how to estimate the A-norm of the error and when to stop the algorithm, but to compare the new and the old formula for computing the existing estimate. For further discussion on estimating the A-norm of the error, stopping criteria, and numerical experiments we refer to [22, 4, 5, 28, 1, 29, 23]. The following experiments are performed in Matlab 7.13 (R2011b).

In the first numerical experiment we solve the system $Ax = b$ with the matrix `bcsstk01` (Harwell-Boeing collection) of order $n = 48$; see also numerical experiments in [23, Chapter 7]. The right-hand side b has been chosen such that b has equal components in the eigenvector basis, and such that $\|b\| = 1$. We choose $x_0 = 0$, $d = 1$, and $\mu = 3.417267e + 3$ (the smallest eigenvalue of the matrix is 3.417267562666500e+3).

In the upper part of Figure 6.1 we plot the A-norm of the error (bold solid line), the Gauss quadrature estimate E_{k-1} (solid line, GQ-estimate), and the GR-estimate $E_{k-1}^{(\mu)}$, where $g_k^{(\mu)}$ is computed using CGQ (dashed line) and using CGQL (dotted line). Visually, it is not possible to distinguish between the estimate $E_{k-1}^{(\mu)}$ computed using CGQ (the new formula) and CGQL (the old formula). However, the lower part

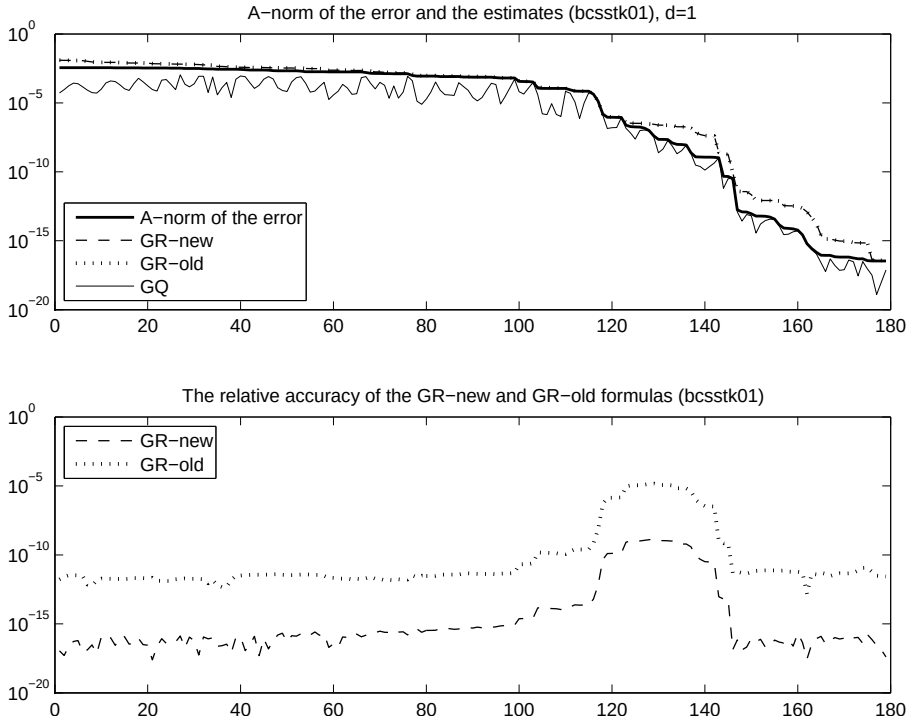

FIG. 6.1. *The upper part: The A-norm of the error (bold solid), the GQ-estimate (solid) and the GR-estimate computed using CGQ (dashed) and CGQL (dotted). The lower part: Relative accuracy of the GR-estimate computed using CGQ (dashed) and CGQL (dotted).*

of Figure 6.1 indicates that the new formula is not only simpler but also more accurate than the old one.

In the lower part of Figure 6.1 we compare the relative accuracy of results computed using the two formulas. First, we compute the quantities $\|r_k\|^2$ and γ_k using the standard double precision CG. Second, we use variable-precision arithmetic in Matlab (Symbolic Toolbox, 64 digits) to get a reasonably accurate value of $E_{k-1}^{(\mu)}$ (computed from the double precision quantities $\|r_k\|^2$ and γ_k). Third, we compute $E_{k-1}^{(\mu)}$ using the standard double precision arithmetic and the two formulas; the corresponding computed values are denoted by $\hat{E}_{k-1}^{(\mu,new)}$ and $\hat{E}_{k-1}^{(\mu,old)}$. Finally, in the lower part of Figure 6.1 we plot the quantities

$$\left| \frac{\hat{E}_{k-1}^{(\mu,new)} - E_{k-1}^{(\mu)}}{E_{k-1}^{(\mu)}} \right| \text{ (dashed)} \qquad \text{and} \qquad \left| \frac{\hat{E}_{k-1}^{(\mu,old)} - E_{k-1}^{(\mu)}}{E_{k-1}^{(\mu)}} \right| \text{ (dotted)}$$

that characterize the relative accuracy of the computed value of $E_{k-1}^{(\mu)}$. We can observe that the new formula is less sensitive to rounding error.

In the second numerical experiment we solve the system $Ax = b$ with the matrix `msc04515` (The University of Florida sparse matrix collection) of order $n = 4515$; see also numerical experiments in [23, Chapter 7]. The right-hand side b has again been chosen such that b has equal components in the eigenvector basis, and $\|b\| = 1$. We choose $x_0 = 0$, $d = 1$, and use the diagonal preconditioner (the preconditioned

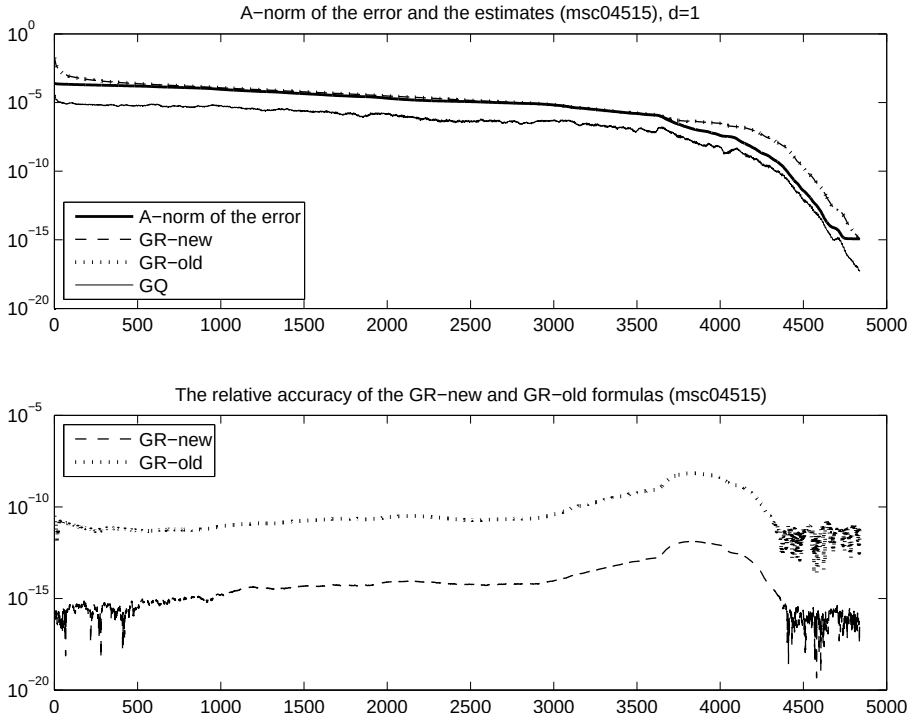

FIG. 6.2. *The upper part: The A-norm of the error (bold solid), the GQ-estimate (solid) and the GR-estimate computed using CGQ (dashed) and CGQL (dotted). The lower part: Relative accuracy of the GR-estimate computed using CGQ (dashed) and CGQL (dotted).*

matrix is diagonally normalized with 1's on the diagonal). The value of μ is given by $\mu = 1.75e-6$ (the smallest eigenvalue of the preconditioned matrix is equal to 1.751795139099631e-6).

In Figure 6.2 we observe more or less the same behaviour as for the first example; the GR-estimate computed using the new formula gives visually the same results, however, the new formula is simpler and more accurate that the old one.

7. Conclusion. In this paper we have described how the bounds based on Gauss quadrature rules for the CG A-norm of the error can be computed in a simple way. In particular, for the Gauss-Radau and Gauss-Lobatto bounds, the preceding implementations computed explicitly the entries of the (modified) Jacobi matrices and used them to compute the bounds. Here we exploited the fact that the LDL^T factorization of the corresponding Jacobi matrix is available in CG and showed how to update LDL^T factorizations of modified Jacobi matrices. The bounds are then computed directly from the known entries of LDL^T factorizations. The algebraic derivation of the new formulas is more difficult than it was when using Jacobi matrices but, in the end, the formulas are simpler. Obtaining simple formulas is a prerequisite for analyzing the behaviour of the bounds in finite precision arithmetic and also for a better understanding of their dependence on the auxiliary parameters μ and η which are lower and upper bounds (or estimates) of the smallest and largest eigenvalues. Numerical experiments predict that the new formulas are less prone to the growth of rounding errors. Therefore, we hope that these improvements will help the implementation of

quadrature-based error bounds into existing and forthcoming CG codes.

REFERENCES

[1] M. Arioli, *A stopping criterion for the conjugate gradient algorithms in a finite element method framework*, Numer. Math., 97 (2004), pp. 1–24.

[2] G. Auchmuty, *A posteriori error estimates for linear equations*, Numer. Math., 61 (1992), pp. 1–6.

[3] C. Brezinski, *Error estimates for the solution of linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 764–781.

[4] D. Calvetti, S. Morigi, L. Reichel, and F. Sgallari, *Computable error bounds and estimates for the conjugate gradient method*, Numer. Algorithms, 25 (2000), pp. 75–88.

[5] ———, *An iterative method with error estimators*, J. Comput. Appl. Math., 127 (2001), pp. 93–119.

[6] 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.

[7] G. Dahlquist, G. H. Golub, and S. G. Nash, *Bounds for the error in linear systems*, in Semi-infinite programming (Proc. Workshop, Bad Honnef, 1978), vol. 15 of Lecture Notes in Control and Information Sci., Springer, Berlin, 1979, pp. 154–172.

[8] B. Fischer and G. H. Golub, *On the error computation for polynomial based iteration methods*, in Recent advances in iterative methods, vol. 60 of IMA Vol. Math. Appl., Springer, New York, 1994, pp. 59–67.

[9] W. Gautschi, *Orthogonal polynomials: computation and approximation*, Oxford University Press, UK, 2004.

[10] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[11] G. H. Golub and G. Meurant, *Matrices, moments and quadrature*, in Numerical analysis 1993 (Dundee, 1993), vol. 303 of Pitman Res. Notes Math. Ser., Longman Sci. Tech., Harlow, 1994, pp. 105–156.

[12] ———, *Matrices, moments and quadrature. II. How to compute the norm of the error in iterative methods*, BIT, 37 (1997), pp. 687–705.

[13] ———, *Matrices, moments and quadrature with applications*, Princeton University Press, USA, 2010.

[14] G. H. Golub and Z. Strakoš, *Estimates in quadratic formulas*, Numer. Algorithms, 8 (1994), pp. 241–268.

[15] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[16] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp. 23 (1969), 221-230; addendum, ibid., 23 (1969), pp. A1–A10.

[17] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[18] P. Jiránek, Z. Strakoš, and M. Vohralík, *A posteriori error estimates including algebraic error and stopping criteria for iterative solvers*, SIAM J. Sci. Comput., 32 (2010), pp. 1567–1590.

[19] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Research Nat. Bur. Standards, 49 (1952), pp. 33–53.

[20] D. P. Laurie, *Anti-Gaussian quadrature formulas*, Math. Comp., 65 (1996), pp. 739–747.

[21] G. Meurant, *The computation of bounds for the norm of the error in the conjugate gradient algorithm*, Numer. Algorithms, 16 (1998), pp. 77–87.

[22] ———, *Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm*, Numer. Algorithms, 22 (1999), pp. 353–365.

[23] ———, *The Lanczos and conjugate gradient algorithms, from theory to finite precision computations*, vol. 19 of Software, Environments, and Tools, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[24] G. Meurant and Z. Strakoš, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numer., 15 (2006), pp. 471–542.

[25] B. N. Parlett, *The new qd algorithms*, Acta Numerica, 15 (1995), pp. 459–491.

[26] B. N. Parlett and I. S. Dhillon, *Relatively robust representations of symmetric tridiagonals*, in Proceedings of the International Workshop on Accurate Solution of Eigenvalue Problems (University Park, PA, 1998), vol. 309, 2000, pp. 121–151.

[27] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, Springer-Verlag, Berlin, Germany, second ed., 1983.

[28] Z. Strakoš and P. Tichý, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80.
[29] ———, *Error estimation in preconditioned conjugate gradients*, BIT, 45 (2005), pp. 789–817.
[30] H. S. Wilf, *Mathematics for the physical sciences*, Wiley, New York, USA, 1962.