Matrices, moments and quadrature *

Gene H. Golub†
Computer Science Department, Stanford University
Stanford CA 94305, USA
golub@sccm.stanford.edu

Gérard Meurant
CEA, Centre d'Etudes de Limeil-Valenton,
94195 Villeneuve St Georges cedex, France
meurant@etca.fr

September 29, 1994

*Presented as the first A.R. Mitchell lecture in Dundee, Scotland, July 1993. We dedicate this paper to Ron Mitchell who has given intellectual leadership and generous support to all.

†The work of this author was supported in part by the National Science Foundation under Grant NSF CCR-8821078

Abstract In this paper we study methods to obtain bounds or approximations of elements of a matrix $f(A)$ where A is a symmetric positive definite matrix and f is a smooth function. These methods are based on the use of quadrature rules and the Lanczos algorithm for diagonal elements and the block Lanczos or the non–symmetric Lanczos algorithms for the non diagonal elements. We give some theoretical results on the behavior of these methods based on results for orthogonal polynomials as well as analytical bounds and numerical experiments on a set of matrices for several functions f.

1 Definition of the problem

Let A be a real symmetric positive definite matrix of order n. We want to find upper and lower bounds (or approximations, if bounds are not available) for the entries of a function of a matrix. We shall examine analytical expressions as well as numerical iterative methods which produce good approximations in a few steps. This problem leads us to consider

$$u^T f(A)v, \tag{1.1}$$

where u and v are given vectors and f is some smooth (possibly C^∞) function on a given interval of the real line.

As an example, if $f(x) = \frac{1}{x}$ and $u^T = e_i^T = (0, \ldots, 0, 1, 0, \ldots, 0)$, the non zero element being in the i–th position and $v = e_j$, we will obtain bounds on the elements of the inverse A^{-1}.

We shall also consider

$$W^T f(A)W,$$

where W is an $n \times m$ matrix. For specificity, we shall most often consider $m = 2$.

Some of the techniques presented in this paper have been used (without any mathematical justification) to solve problems in solid state physics, particularly to compute elements of the resolvant of a Hamiltonian modeling the interaction of atoms in a solid, see [12], [14], [15]. In these studies the function f is the inverse of its argument.

Analytic bounds for elements of inverses of matrices using different techniques have been recently obtained in [17].

The outline of the paper is as follows. Section 2 considers the problem of characterizing the elements of a function of a matrix. The theory is developed in Section 3 and Section 4 deals with the construction of the orthogonal polynomials that are needed to obtain a numerical method for computing bounds. The Lanczos, non–symmetric Lanczos and block Lanczos methods used for the computation of the polynomials are presented there. Applications to the computation of elements of the inverse of a matrix are described in Section 5 where very simple iterative algorithms are given to compute bounds. Some numerical examples are given in Section 6, for different matrices and functions f.

2 Elements of a function of a matrix

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 which we order as

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

By definition, we have

$$f(A) = Qf(\Lambda)Q^T.$$

Therefore,

$$\begin{aligned} u^T f(A)v &= u^T Q f(\Lambda) Q^T v \\ &= \alpha^T f(\Lambda)\beta, \\ &= \sum_{i=1}^{n} f(\lambda_i)\alpha_i\beta_i. \end{aligned}$$

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

$$I[f] = u^T f(A)v = \int_a^b f(\lambda)\ d\alpha(\lambda), \tag{2.1}$$

where the measure α is piecewise constant and defined by

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

When $u = v$, we note that α is an increasing positive function.

The block generalization is obtained in the following way. Let W be an $n \times 2$ matrix, $W = (w_1\ w_2)$, then

$$W^T f(A)W = W^T Q f(\Lambda) Q^T W = \alpha f(\Lambda)\alpha^T,$$

where, of course, α is a $2 \times n$ matrix such that

$$\alpha = (\alpha_1 \ldots \alpha_n),$$

and α_i is a vector with two components. With these notations, we have

$$W^T f(A)W = \sum_{i=1}^{n} f(\lambda_i)\alpha_i\alpha_i^T.$$

This can be written as a matrix Riemann–Stieltjes integral

$$I_B[f] = W^T f(A)W = \int_a^b f(\lambda)\ d\alpha(\lambda).$$

$I_B[f]$ is a 2×2 matrix where the entries of the (matrix) measure α are piecewise constant and defined by

$$\alpha(\lambda) = \sum_{k=1}^{l} \alpha_k\alpha_k^T, \quad \lambda_l \leq \lambda < \lambda_{l+1}.$$

In this paper, we are looking for methods to obtain upper and lower bounds L and U for $I[f]$ and $I_B[f]$,

$$\begin{aligned} L &\leq I[f] \leq U \\ L &\leq I_B[f] \leq U. \end{aligned}$$

In the next section, we review and describe some basic results from Gauss quadrature theory as this plays a fundamental role in estimating the integrals and computing bounds.

3 Bounds on matrix functions as integrals

A way to obtain bounds for the Stieltjes integrals is to use Gauss, Gauss–Radau and Gauss–Lobatto quadrature formulas, see [3],[8],[9]. For 1.1, the general formula we will use is

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f], \tag{3.1}$$

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, see [4],[5],[6],[7].

3.1 The case $u = v$

When $u = v$, the measure is a positive increasing function and it is known (see for instance [18]) that

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \prod_{k=1}^{M} (\lambda - z_k) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda), \quad a < \eta < b. \tag{3.2}$$

If $M = 0$, this leads to the Gauss rule with no prescribed nodes. 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.

Let us recall briefly how the nodes and weights are obtained in the Gauss, Gauss–Radau and Gauss–Lobatto rules. For the measure α, it is possible to define a sequence of polynomials $p_0(\lambda), p_1(\lambda), \ldots$ that are orthonormal with respect to α:

$$\int_a^b p_i(\lambda) p_j(\lambda)\ d\alpha(\lambda) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

and p_k is of exact degree k. Moreover, the roots of p_k are distinct, real and lie in the interval $[a, b]$. We will see how to compute these polynomials in the next Section.

This set of orthonormal polynomials satisfies a three term recurrence relationship (see [20]):

$$\begin{aligned} \gamma_j p_j(\lambda) = (\lambda - \omega_j) p_{j-1}(\lambda) - \gamma_{j-1} p_{j-2}(\lambda), \quad j = 1, 2, \ldots, N \\ p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1, \end{aligned} \tag{3.3}$$

if $\int d\alpha = 1$.

In matrix form, this can be written as

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

where

$$p(\lambda)^T = [p_0(\lambda)\ p_1(\lambda) \cdots\ p_{N-1}(\lambda)],$$
$$e_N^T = (0\ 0\ \cdots\ 0\ 1),$$
$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \gamma_{N-1} & \omega_N \end{pmatrix}. \tag{3.4}$$

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, cf. [7]. We note that all the eigenvalues of J_N are real and simple.

For the Gauss quadrature rule (renaming the weights and nodes w_j^G and t_j^G) we have

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} w_j^G f(t_j^G) + R_G[f],$$

with

$$R_G[f] = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b \left[\prod_{j=1}^{N} (\lambda - t_j^G) \right]^2 d\alpha(\lambda),$$

and the next theorem follows.

Theorem 3.1 *Suppose $u = v$ in 2.1 and f is such that $f^{(2n)}(\xi) > 0,\ \forall n,\ \forall \xi,\ a < \xi < b$, and let*

$$L_G[f] = \sum_{j=1}^{N} w_j^G f(t_j^G).$$

Then, $\forall N,\ \exists \eta \in [a, b]$ such that

$$L_G[f] \leq I[f],$$
$$I[f] - L_G[f] = \frac{f^{(2N)}(\eta)}{(2N)!}.$$

Proof: See [18]. The main idea of the proof is to use a Hermite interpolatory polynomial of degree $2N - 1$ on the N nodes which allows us to express the remainder as an integral of the difference between the function and its interpolatory polynomial and to apply the mean value theorem (as the measure is positive and increasing). As we know the sign of the remainder, we easily obtain bounds.

To obtain the Gauss–Radau rule ($M = 1$ in 3.1–3.2), we should extend the matrix J_N in 3.4 in such a way that it has one prescribed eigenvalue, see [8].

Assume $z_1 = a$, we wish to construct p_{N+1} such that $p_{N+1}(a) = 0$. From the recurrence relation 3.3, we have

$$0 = \gamma_{N+1} p_{N+1}(a) = (a - \omega_{N+1}) p_N(a) - \gamma_N p_{N-1}(a).$$

This gives

$$\omega_{N+1} = a - \gamma_N \frac{p_{N-1}(a)}{p_N(a)}.$$

We have also

$$(J_N - aI)p(a) = -\gamma_N p_N(a) e_N.$$

Let us denote $\delta(a) = [\delta_1(a), \cdots, \delta_N(a)]^T$ with

$$\delta_l(a) = -\gamma_N \frac{p_{l-1}(a)}{p_N(a)} \quad l = 1, \ldots, N.$$

This gives $\omega_{N+1} = a + \delta_N(a)$ and

$$(J_N - aI)\delta(a) = \gamma_N^2 e_N. \tag{3.5}$$

From these relations we have the solution of the problem as: 1) we generate γ_N by the Lanczos process (see Section 4 for the definition), 2) we solve the tridiagonal system 3.5 for $\delta(a)$ and 3) we compute ω_{N+1}. Then the tridiagonal matrix $\hat{J}_{N+1}$ defined as

$$\hat{J}_{N+1} = \begin{pmatrix} J_N & \gamma_N e_N \\ \gamma_N e_N^T & \omega_{N+1} \end{pmatrix},$$

will have a as an eigenvalue and gives the weights and the nodes of the corresponding quadrature rule. Therefore, the recipe is to compute as for the Gauss quadrature rule and then to modify the last step to obtain the prescribed node.

For Gauss–Radau the remainder R_{GR} is

$$R_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - z_1) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda).$$

Again, this is proved by constructing an interpolatory polynomial for the function and its derivative on the t_js and for the function on z_1.

Therefore, if we know the sign of the derivatives of f, we can bound the remainder. This is stated in the following theorem.

Theorem 3.2 *Suppose $u = v$ and f is such that $f^{(2n+1)}(\xi) < 0$, $\forall n, \forall \xi, a < \xi < b$. Let U_{GR} be defined as*

$$U_{GR}[f] = \sum_{j=1}^{N} w_j^a f(t_j^a) + v_1^a f(a),$$

w_j^a, v_1^a, t_j^a being the weights and nodes computed with $z_1 = a$ and let L_{GR} be defined as

$$L_{GR}[f] = \sum_{j=1}^{N} w_j^b f(t_j^b) + v_1^b f(b),$$

w_j^b, v_1^b, t_j^b being the weights and nodes computed with $z_1 = b$. Then, $\forall N$ we have

$$L_{GR}[f] \leq I[f] \leq U_{GR}[f],$$

and

$$I[f] - U_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - a) \left[\prod_{j=1}^{N} (\lambda - t_j^a) \right]^2 d\alpha(\lambda),$$

$$I[f] - L_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j^b) \right]^2 d\alpha(\lambda).$$

Proof : With our hypothesis the sign of the remainder is easily obtained. It is negative if we choose $z_1 = a$, positive if we choose $z_1 = b$.

Remarks :

i) if the sign of the f derivatives is positive, the bounds are reversed.

ii) it is enough to suppose that there exists an n_0 such that $f^{(2n_0+1)}(\eta) < 0$ but, then $N = n_0$ is fixed.

Now, consider the Gauss–Lobatto rule ($M = 2$ in 3.1–3.2), with $z_1 = a$ and $z_2 = b$ as prescribed nodes. Again, we should modify the matrix of the Gauss quadrature rule, see [8]. Here, we would like to have

$$p_{N+1}(a) = p_{N+1}(b) = 0.$$

Using the recurrence relation 3.3 for the polynomials, this leads to a linear system of order 2 for the unknowns ω_{N+1} and γ_N:

$$\begin{pmatrix} p_N(a) & p_{N-1}(a) \\ p_N(b) & p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N \end{pmatrix} = \begin{pmatrix} a\ p_N(a) \\ b\ p_N(b) \end{pmatrix}. \tag{3.6}$$

Let δ and μ be defined as vectors with components

$$\delta_l = -\frac{p_{l-1}(a)}{\gamma_N p_N(a)}, \quad \mu_l = -\frac{p_{l-1}(b)}{\gamma_N p_N(b)},$$

then

$$(J_N - aI)\delta = e_N, \quad (J_N - bI)\mu = e_N,$$

and the linear system 3.6 can be written

$$\begin{pmatrix} 1 & -\delta_N \\ 1 & -\mu_N \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N^2 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix},$$

giving the unknowns that we need. The tridiagonal matrix $\hat{J}_{N+1}$ is then defined as in the Gauss–Radau rule.

Having computed the nodes and weights, we have

$$\int_a^b f(\lambda) d\alpha(\lambda) = \sum_{j=1}^{N} w_j^{GL} f(t_j^{GL}) + v_1 f(a) + v_2 f(b) + R_{GL}[f],$$

where

$$R_{GL}[f] = \frac{f^{(2N+2)}(\eta)}{(2N+2)!} \int_a^b (\lambda - a)(\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda).$$

Then, we have the following obvious result.

Theorem 3.3 *Suppose $u = v$ and f is such that $f^{(2n)}(\eta) > 0$, $\forall n$, $\forall \eta, a < \eta < b$ and let*

$$U_{GL}[f] = \sum_{j=1}^{N} w_j^{GL} f(t_j^{GL}) + v_1 f(a) + v_2 f(b).$$

Then, $\forall N$

$$I[f] \leq U_{GL}[f],$$

$$I[f] - U_{GL}[f] = \frac{f^{(2N+2)}(\eta)}{(2N+2)!} \int_a^b (\lambda - a)(\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j^{GL}) \right]^2 d\alpha(\lambda).$$

We remark that we need not always compute the eigenvalues and eigenvectors of the tridiagonal matrix. Let Y_N be the matrix of the eigenvectors of J_N (or $\hat{J}_N$) whose columns we denote by y_i and T_N be the diagonal matrix of the eigenvalues t_i which give the nodes of the Gauss quadrature rule. It is well known that the weights w_i are given by (cf. [21])

$$\frac{1}{w_i} = \sum_{l=0}^{N-1} p_l^2(t_i).$$

It can be easily shown that

$$w_i = \left(\frac{y_i^1}{p_0(t_i)} \right)^2,$$

where y_i^1 is the first component of y_i.

But, since $p_0(\lambda) \equiv 1$, we have,

$$w_i = (y_i^1)^2 = (e_1^T y_i)^2.$$

Theorem 3.4

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

Proof:

$$\begin{aligned} \sum_{l=1}^{N} w_l f(t_l) &= \sum_{l=1}^{N} e_1^T y_l f(t_l) y_l^T e_1 \\ &= e_1^T \left(\sum_{l=1}^{N} y_l f(t_l) y_l^T \right) e_1 \\ &= e_1^T Y_N f(T_N) Y_N^T e_1 \\ &= e_1^T f(J_N) e_1. \end{aligned}$$

The same statement is true for the Gauss–Radau and Gauss–Lobatto rules. Therefore, in some cases where $f(J_N)$ (or the equivalent) is easily computable (for instance, if $f(\lambda) = 1/\lambda$, see Section 5), we do not need to compute the eigenvalues and eigenvectors of J_N.

3.2 The case $u \neq v$

We have seen that the measure in 2.1 is piecewise constant and defined by

$$\alpha(\lambda) = \sum_{k=1}^{l} \alpha_k \delta_k, \quad \lambda_l \leq \lambda < \lambda_{l+1}.$$

For variable signed weight functions, see [19]. We will see later that for our application, u and v can always be chosen such that $\alpha_k \delta_k \geq 0$. Therefore, in this case α will be a positive increasing function.

In the next Section, we will show that there exists two sequences of polynomials p and q such that

$$\begin{aligned} \gamma_j p_j(\lambda) &= (\lambda - \omega_j) p_{j-1}(\lambda) - \beta_{j-1} p_{j-2}(\lambda), \quad p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1, \\ \beta_j q_j(\lambda) &= (\lambda - \omega_j) q_{j-1}(\lambda) - \gamma_{j-1} q_{j-2}(\lambda), \quad q_{-1}(\lambda) \equiv 0, \quad q_0(\lambda) \equiv 1. \end{aligned}$$

Let

$$\begin{aligned} p(\lambda)^T &= [p_0(\lambda)\ p_1(\lambda) \cdots\ p_{N-1}(\lambda)], \\ q(\lambda)^T &= [q_0(\lambda)\ q_1(\lambda) \cdots\ q_{N-1}(\lambda)], \end{aligned}$$

and

$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \beta_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \beta_{N-1} & \omega_N \end{pmatrix}.$$

Then, we can write

$$\begin{aligned} \lambda p(\lambda) &= J_N p(\lambda) + \gamma_N p_N(\lambda) e_N, \\ \lambda q(\lambda) &= J_N^T q(\lambda) + \beta_N q_N(\lambda) e_N. \end{aligned}$$

Theorem 3.5

$$p_j(\lambda) = \frac{\beta_j \cdots \beta_1}{\gamma_j \cdots \gamma_1} q_j(\lambda).$$

Proof: The theorem is proved by induction. We have

$$\gamma_1 p_1(\lambda) = \lambda - \omega_1,$$

$$\beta_1 q_1(\lambda) = \lambda - \omega_1,$$

therefore

$$p_1(\lambda) = \frac{\beta_1}{\gamma_1} q_1(\lambda).$$

Now, suppose that

$$p_{j-1}(\lambda) = \frac{\beta_{j-1} \cdots \beta_1}{\gamma_{j-1} \cdots \gamma_1} q_{j-1}(\lambda).$$

We have

$$\begin{aligned}\gamma_j p_j(\lambda) &= (\lambda-\omega_j)p_{j-1}(\lambda) - \beta_{j-1}p_{j-2}(\lambda) \\ &= (\lambda-\omega_j)\frac{\beta_{j-1}\cdots\beta_1}{\gamma_{j-1}\cdots\gamma_1}q_{j-1}(\lambda) - \beta_{j-1}\frac{\beta_{j-2}\cdots\beta_1}{\gamma_{j-2}\cdots\gamma_1}q_{j-2}(\lambda).\end{aligned}$$

Multiplying by $\frac{\gamma_{j-1}\cdots\gamma_1}{\beta_{j-1}\cdots\beta_1}$ we obtain the result.

Hence q_N is a multiple of p_N and the polynomials have the same roots which are also the common eigenvalues of J_N and J_N^T.

We will see that it is possible to choose γ_j and β_j such that

$$\gamma_j = \pm\beta_j,$$

with, for instance, $\gamma_j \geq 0$. Then, we have

$$p_j(\lambda) = \pm q_j(\lambda).$$

We define the quadrature rule as

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} f(\lambda_j)s_j t_j + error, \tag{3.7}$$

where λ_j is an eigenvalue of J_N, s_j is the first component of the eigenvector u_j of J_N corresponding to λ_j and t_j is the first component of the eigenvector v_j of J_N^T corresponding to the same eigenvalue, normalized such that $v_j^T u_j = 1$.

We have the following results:

Proposition 3.1 *Suppose that $\gamma_j\beta_j \neq 0$, then the (non–symmetric) Gauss quadrature rule 3.7 is exact for polynomials of degree less than or equal to $N-1$.*

Proof:

The function f can be written as

$$f(\lambda) = \sum_{k=0}^{N-1} c_k p_k(\lambda),$$

and because of the orthonormality properties

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = c_0.$$

For the quadrature rule, we have

$$\begin{aligned}\sum_{j=1}^{N} f(\lambda_j)s_j t_j q_l(\lambda_j) &= \sum_{j=1}^{N}\sum_{k=0}^{N-1} c_k p_k(\lambda_j)s_j t_j q_l(\lambda_j) \\ &= \sum_{k=0}^{N-1} c_k \sum_{j=1}^{N} p_k(\lambda_j)s_j t_j q_l(\lambda_j).\end{aligned}$$

But $p_k(\lambda_j)s_j$ and $q_l(\lambda_j)t_j$ are respectively the components of the eigenvectors of J_N and J_N^T corresponding to λ_j. Therefore they are orthonormal with the normalization that we chose. Hence,

$$\sum_{j=1}^{N} f(\lambda_j)s_j t_j q_l(\lambda_j) = c_l,$$

and consequently

$$\sum_{j=1}^{N} f(\lambda_j)s_j t_j = c_0,$$

which proves the result.

Now, as in [14], we extend the result to polynomials of higher degree.

Theorem 3.6 *Suppose that $\gamma_j\beta_j \neq 0$, then the (non–symmetric) Gauss quadrature rule 3.7 is exact for polynomials of degree less than or equal to $2N-1$.*

Proof:

Suppose f is a polynomial of degree $2N-1$. Then, f can be written as

$$f(\lambda) = p_N(\lambda)s(\lambda) + r(\lambda),$$

where s and r are polynomials of degree less or equal to $N-1$. Then,

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \int_a^b p_N(\lambda)s(\lambda)\ d\alpha(\lambda) + \int_a^b r(\lambda)\ d\alpha(\lambda) = \int_a^b r(\lambda)\ d\alpha(\lambda),$$

since p_N is orthogonal to any polynomial of degree less or equal to $N-1$ because of the orthogonality property of the p and q's.

For the quadrature rule, we have

$$\sum_{j=1}^{N} p_N(\lambda_j)s(\lambda_j)s_j t_j + \sum_{j=1}^{N} r(\lambda_j)s_j t_j.$$

But, as λ_j is an eigenvalue of J_N, it is a root of p_N and

$$\sum_{j=1}^{N} p_N(\lambda_j)s(\lambda_j)s_j t_j = 0.$$

As the quadrature rule has been proven to be exact for polynomials of degree less than $N-1$,

$$\int_a^b r(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} r(\lambda_j)s_j t_j,$$

which proves the Theorem.

We will see in the next Section how to obtain bounds on the integral 2.1.

Now, we extend the Gauss–Radau and Gauss–Lobatto rules to the non–symmetric case. This is almost identical (up to technical details) to the symmetric case.

For Gauss–Radau, assume that the prescribed node is a, then, we would like to have $p_{N+1}(a) = q_{N+1}(a) = 0$. This gives

$$(a - \omega_{N+1})p_N(a) - \beta_N p_{N-1}(a) = 0.$$

If we denote $\delta(a) = [\delta_1(a), \ldots, \delta_N(a)]^T$, with

$$\delta_l(a) = -\beta_N \frac{p_{l-1}(a)}{p_N(a)},$$

we have

$$\omega_{N+1} = a + \delta_N(a),$$

where

$$(J_N - aI)\delta(a) = \gamma_N \beta_N e_N.$$

Therefore, the algorithm is essentially the same as previously discussed.

For Gauss–Lobatto, the algorithm is also almost the same as for the symmetric case. We would like to compute p_{N+1} and q_{N+1} such that

$$p_{N+1}(a) = p_{N+1}(b) = 0, \quad q_{N+1}(a) = q_{N+1}(b) = 0.$$

This leads to solving the linear system

$$\begin{pmatrix} p_N(a) & p_{N-1}(a) \\ p_N(b) & p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \beta_N \end{pmatrix} = \begin{pmatrix} ap_N(a) \\ bp_N(b) \end{pmatrix}.$$

The linear system for the q's whose solution is $(\omega_{N+1}, \gamma_N)^T$ can be shown to have the same solution for ω_{N+1} and $\gamma_N = \pm\beta_N$ depending on the signs relations between the p's and the q's.

Let $\delta(a)$ and $\mu(b)$ be the solutions of

$$(J_N - aI)\delta(a) = e_N, \quad (J_N - bI)\mu(b) = e_N.$$

Then, we have

$$\begin{pmatrix} 1 & -\delta(a)_N \\ 1 & -\mu(b)_N \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N \beta_N \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

When we have the solution of this system, we choose $\gamma_N = \pm\beta_N$ and $\gamma_N \geq 0$.

The question of establishing bounds on the integral will be studied in the next Section.

As for the case $u = v$, we do not always need compute the eigenvalues and eigenvectors of J_N but only the $(1,1)$ element of $f(J_N)$.

3.3 The block case

Now, we consider the block case. The problem is to find a quadrature rule. The integral $\int_a^b f(\lambda)d\alpha(\lambda)$ is a 2×2 symmetric matrix. The most general quadrature formula is of the form

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{j=1}^{N} W_j f(T_j) W_j + error,$$

where W_j and T_j are symmetric 2×2 matrices. In this sum, we have $6N$ unknowns. This quadrature rule can be simplified, since

$$T_j = Q_j \Lambda_j Q_j^T,$$

where Q_j is the orthonormal matrix of the eigenvectors, and Λ_j, the diagonal matrix of the eigenvalues of T_j. This gives

$$\sum_{j=1}^{N} W_j Q_j f(\Lambda_j) Q_j^T W_j.$$

But $W_j Q_j f(\Lambda_j) Q_j^T W_j$ can be written as

$$f(\lambda_1) z_1 z_1^T + f(\lambda_2) z_2 z_2^T,$$

where the vector z_i is 2×1. Therefore, the quadrature rule can be written as

$$\sum_{j=1}^{2N} f(t_j) w_j w_j^T,$$

where t_j is a scalar and w_j is a vector with 2 components. In this quadrature rule, there are also $6N$ unknowns.

In the next Section, we will show that there exists orthogonal matrix polynomials such that

$$\lambda p_{j-1}(\lambda) = p_j(\lambda)\Gamma_j + p_{j-1}(\lambda)\Omega_j + p_{j-2}(\lambda)\Gamma_{j-1}^T,$$
$$p_0(\lambda) \equiv I_2, \quad p_{-1}(\lambda) \equiv 0.$$

This can be written as

$$\lambda[p_0(\lambda), \ldots, p_{N-1}(\lambda)] = [p_0(\lambda), \ldots, p_{N-1}(\lambda)] J_N + [0, \ldots, 0, p_N(\lambda)\Gamma_N],$$

where

$$J_N = \begin{pmatrix} \Omega_1 & \Gamma_1^T & & & \\ \Gamma_1 & \Omega_2 & \Gamma_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-2} & \Omega_{N-1} & \Gamma_{N-1}^T \\ & & & \Gamma_{N-1} & \Omega_N \end{pmatrix}, \tag{3.8}$$

is a block tridiagonal matrix of order $2N$ and a banded matrix whose half bandwidth is 2 (we have at most 5 non zero elements in a row).

If we denote $P(\lambda) = [p_0(\lambda), \ldots, p_{N-1}(\lambda)]^T$, we have as J_N is symmetric

$$J_N P(\lambda) = \lambda P(\lambda) - [0, \ldots, 0, p_N(\lambda)\Gamma_N]^T.$$

We note that if λ is an eigenvalue, say λ_r, of J_N and if we choose $u = u_r$ to be a two element vector whose components are the first two components of an eigenvector corresponding to λ_r, then $P(\lambda_r)u$ is this eigenvector (because of the relations that are satisfied) and if

Γ_N is non singular, $p_N^T(\lambda_r)u = 0$. The difference with the scalar case is that although the eigenvalues are real, it might be that they are of multiplicity greater than 1 (although this is unlikely except in the case of the Gauss-Radau and Gauss-Lobatto rule where this condition is enforced).

We define the quadrature rule as:

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{i=1}^{2N} f(\lambda_i) u_i u_i^T + error, \tag{3.9}$$

where $2N$ is the order of J_N, the eigenvalues λ_i are those of J_N and u_i is the vector consisting of the two first components of the corresponding eigenvector, normalized as before. In fact, if there are multiple eigenvalues, the quadrature rule should be written as follows. Let $\mu_i, i = 1, \ldots, l$ be the set of distinct eigenvalues and q_i their multiplicities. The quadrature rule is then

$$\sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \right) f(\mu_i). \tag{3.10}$$

We will show in the next Section that the Gauss quadrature rule is exact for polynomials of degree $2N-1$ and how to obtain estimates of the error.

We extend the process described for scalar polynomials to the matrix analog of the Gauss–Radau quadrature rule. Let a be an extreme eigenvalue of A. We would like a to be a double eigenvalue of J_{N+1}. We have

$$J_{N+1}P(a) = aP(a) - [0, \ldots, 0, p_{N+1}(a)\Gamma_{N+1}]^T.$$

Then, we need to require $p_{N+1}(a) \equiv 0$. From the recurrence relation this translates into

$$ap_N(a) - p_N(a)\Omega_{N+1} - p_{N-1}(a)\Gamma_N^T = 0.$$

Therefore, if $p_N(a)$ is non singular, we have

$$\Omega_{N+1} = aI_2 - p_N(a)^{-1}p_{N-1}(a)\Gamma_N^T.$$

We must compute the right hand side. This can be done by noting that

$$J_N \begin{pmatrix} p_0(a)^T \\ \vdots \\ p_{N-1}(a)^T \end{pmatrix} = a \begin{pmatrix} p_0(a)^T \\ \vdots \\ p_{N-1}(a)^T \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T p_N(a)^T \end{pmatrix}.$$

Multiplying on the right by $p_N(a)^{-T}$, we get the matrix equation

$$(J_N - aI) \begin{pmatrix} -p_0(a)^T p_N(a)^{-T} \\ \vdots \\ -p_{N-1}(a)^T p_N(a)^{-T} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T \end{pmatrix}.$$

Thus, we solve

$$(J_N - aI) \begin{pmatrix} \delta_0(a) \\ \vdots \\ \delta_{N-1}(a) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T \end{pmatrix},$$

and hence

$$\Omega_{N+1} = aI_2 + \delta_{N-1}(a)^T \Gamma_N^T.$$

The generalization of Gauss–Lobatto to the block case is a little more tricky. We would like to have a and b as double eigenvalues of the matrix J_{N+1}. This leads to satisfying the following two matrix equations

$$ap_N(a) - p_N(a)\Omega_{N+1} - p_{N-1}(a)\Gamma_N^T = 0$$

$$bp_N(b) - p_N(b)\Omega_{N+1} - p_{N-1}(b)\Gamma_N^T = 0$$

This can be written as

$$\begin{pmatrix} I_2 & p_N^{-1}(a)p_{N-1}(a) \\ I_2 & p_N^{-1}(b)p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \Omega_{N+1} \\ \Gamma_N^T \end{pmatrix} = \begin{pmatrix} aI_2 \\ bI_2 \end{pmatrix}.$$

We now consider the problem of computing (or avoid computing) $p_N^{-1}(\lambda)p_{N-1}(\lambda)$. Let $\delta(\lambda)$ be the solution of

$$(J_N - \lambda I)\delta(\lambda) = (0 \ldots 0 \ I_2)^T.$$

Then, as before

$$\delta_{N-1}(\lambda) = -p_{N-1}(\lambda)^T p_N(\lambda)^{-T} \Gamma_N^{-T}.$$

We can easily show that $\delta_{N-1}(\lambda)$ is symmetric. We consider solving a 2×2 block linear system

$$\begin{pmatrix} I & X \\ I & Y \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} aI \\ bI \end{pmatrix}.$$

Consider the block factorization

$$\begin{pmatrix} I & X \\ I & Y \end{pmatrix} = \begin{pmatrix} I & 0 \\ I & W \end{pmatrix} \begin{pmatrix} I & X \\ 0 & Z \end{pmatrix},$$

thus $WZ = Y - X$.

The solution of the system

$$\begin{pmatrix} I & 0 \\ I & W \end{pmatrix} \begin{pmatrix} U_1 \\ V_1 \end{pmatrix} = \begin{pmatrix} aI \\ bI \end{pmatrix},$$

gives

$$WV_1 = (b-a)I.$$

The next step is

$$\begin{pmatrix} I & X \\ 0 & Z \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} U_1 \\ V_1 \end{pmatrix},$$

and we get

$$ZV = V_1 = W^{-1}(b-a)I$$

or

$$(WZ)V = (b-a)I.$$

Therefore

$$V = (b-a)(Y-X)^{-1}.$$

Hence, we have

$$Y - X = p_N^{-1}(b)p_{N-1}(b) - p_N^{-1}(a)p_{N-1}(a) = \Gamma_N(\delta_{N-1}(a) - \delta_{N-1}(b)).$$

This means that

$$\Gamma_N^T = (b-a)(\delta_{N-1}(a) - \delta_{N-1}(b))^{-1}\Gamma_N^{-1},$$

or

$$\Gamma_N^T\Gamma_N = (b-a)(\delta_{N-1}(a) - \delta_{N-1}(b))^{-1}.$$

Then, Γ_N is given as a Cholesky decomposition of the right hand side matrix. The right hand side is positive definite because $\delta_{N-1}(a)$ is a diagonal block of the inverse of $(J_N - aI)^{-1}$ which is positive definite because the eigenvalues of J_N are larger that a and $-\delta_{N-1}(b)$ is the negative of a diagonal block of $(J_N - bI)^{-1}$ which is positive definite because the eigenvalues of J_N are smaller that b.

From Γ_N, we can compute Ω_{N+1}:

$$\Omega_{N+1} = aI_2 + \Gamma_N\delta_{N-1}(a)\Gamma_N^T.$$

As for the scalar case, it is not always needed to compute the weights and the nodes for the quadrature rules.

Theorem 3.7 *We have*

$$\sum_{i=1}^{2N} f(\lambda_i)u_iu_i^T = e^T f(J_N)e,$$

where $e^T = (I_2\ 0 \ldots 0)$.

Proof:

The quadrature rule is

$$\sum_{i=1}^{2N} u_i f(\lambda_i)u_i^T.$$

If y_i are the eigenvectors of J_N then $u_i = e^T y_i$ and

$$\begin{aligned}
\sum_{i=1}^{2N} u_i f(\lambda_i)u_i^T &= \sum_{i=1}^{2N} e^T y_i f(\lambda_i) y_i^T e \\
&= e^T \left(\sum_{i=1}^{2N} y_i f(\lambda_i) y_i^T \right) e \\
&= e^T Y_N f(T_N) Y_N^T e \\
&= e^T f(J_N) e
\end{aligned}$$

where Y_N is the matrix of the eigenvectors and T_N the diagonal matrix of the eigenvalues of J_N.

Note that bounds for non diagonal elements can also be obtained by considering $e_i^T f(A)e_i$, $e_j^T f(A)e_j$ and $\frac{1}{2}(e_i + e_j)^T f(A)(e_i + e_j)$.

4 Construction of the orthogonal polynomials

In this section we consider the problem of computing the orthonormal polynomials or equivalently the tridiagonal matrices that we need. A very natural and elegant way to do this is to use Lanczos algorithms.

4.1 The case $u = v$

When $u = v$, we use the classical Lanczos algorithm.

Let $x_{-1} = 0$ and x_0 be given such that $\|x_0\| = 1$. The Lanczos algorithm is defined by the following relations,

$$\gamma_j x_j = r_j = (A - \omega_j I)x_{j-1} - \gamma_{j-1}x_{j-2}, \quad j = 1, \ldots$$
$$\omega_j = x_{j-1}^T A x_{j-1},$$
$$\gamma_j = \|r_j\|.$$

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

$$\text{span}\{x_0, Ax_0, \ldots, A^l x_0\}.$$

Proposition 4.1 *The vector x_j is given by*

$$x_j = p_j(A)x_0,$$

where p_j is a polynomial of degree j defined by the three term recurrence (identical to 3.3)

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

Proof:

$$\gamma_1 x_1 = (A - \omega_1 I)x_0,$$

is a first order polynomial in A. Therefore, the Proposition is easily obtained by induction.

Theorem 4.1 *If $x_0 = u$, we have*

$$x_k^T x_l = \int_a^b p_k(\lambda)p_l(\lambda)d\alpha(\lambda).$$

Proof: As the x_j's are orthonormal, we have

$$\begin{aligned} x_k^T x_l &= x_0^T P_k(A)^T P_l(A)x_0 \\ &= x_0^T Q P_k(\Lambda)Q^T Q P_l(\Lambda)Q^T x_0 \\ &= x_0^T Q P_k(\Lambda)P_l(\Lambda)Q^T x_0 \\ &= \sum_{j=1}^n p_k(\lambda_j)p_l(\lambda_j)\hat{x}_j^2, \end{aligned}$$

where $\hat{x} = Q^T x_0$.

Therefore, the p_j's are the orthonormal polynomials related to α that we were referring to in 3.3.

4.2 The case $u \neq v$

We apply the non–symmetric Lanczos algorithm to a symmetric matrix A.

Let $x_{-1} = \hat{x}_{-1} = 0$, and $x_0, \hat{x}_0$ be given with $x_0 \neq \hat{x}_0$ and $x_0^T \hat{x}_0 = 1$. Then we define the iterates for $j = 1, \ldots$ by

$$\gamma_j x_j = r_j = (A - \omega_j I) x_{j-1} - \beta_{j-1} x_{j-2}, \tag{4.1}$$

$$\beta_j \hat{x}_j = \hat{r}_j = (A - \omega_j I) \hat{x}_{j-1} - \gamma_{j-1} \hat{x}_{j-2}, \tag{4.2}$$

$$\omega_j = \hat{x}_{j-1}^T A x_{j-1},$$

$$\gamma_j \beta_j = \hat{r}_j^T r_j.$$

This algorithm generates two sequences of mutually orthogonal vectors as we have

$$x_l^T \hat{x}_k = \delta_{kl}.$$

We have basically the same properties as for the Lanczos algorithm.

Proposition 4.2

$$x_j = p_j(A) x_0, \quad \hat{x}_j = q_j(A) \hat{x}_0,$$

where p_j and q_j are polynomials of degree j defined by the three term recurrences

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

$$\beta_j q_j(\lambda) = (\lambda - \omega_j) q_{j-1}(\lambda) - \gamma_{j-1} q_{j-2}(\lambda), \quad q_{-1}(\lambda) \equiv 0, \quad q_0(\lambda) \equiv 1.$$

Proof: The Proposition is easily obtained by induction.

Theorem 4.2 *If $x_0 = u$ and $\hat{x}_0 = v$, then*

$$x_k^T \hat{x}_l = \int_a^b p_k(\lambda) q_l(\lambda) d\alpha(\lambda) = \delta_{kl}.$$

Proof: As the x_j's and $\hat{x}_j$'s are orthonormal the proof is identical to the proof of Theorem 4.1.

We have seen in the previous Section the relationship between the p and q's. The polynomials q are multiples of the polynomials p.

In this particular application of the non–symmetric Lanczos algorithm it is possible to choose γ_j and β_j such that

$$\gamma_j = \pm \beta_j = \pm \sqrt{|\hat{r}_j^T r_j|},$$

with, for instance, $\gamma_j \geq 0$ and $\beta_j = \mathrm{sgn}(\hat{r}_j^T r_j) \gamma_j$. Then, we have

$$p_j(\lambda) = \pm q_j(\lambda).$$

The main difference with the case of symmetric Lanczos is that this algorithm may break down, e.g. we can have $\gamma_j \beta_j = 0$ at some step.

We would like to use the non–symmetric Lanczos algorithm with $x_0 = e_i$ and $\hat{x}_0 = e_j$ to get estimates of $f(A)_{i,j}$. Unfortunately, this is not possible as this implies $x_0^T \hat{x}_0 = 0$. A way to get around this problem is to set $x_0 = e_i/\delta$ and $\hat{x}_0 = \delta e_i + e_j$. This will give an estimate of $f(A)_{i,j}/\delta + f(A)_{i,i}$ and we can use the bounds we get for the diagonal elements (using for instance symmetric Lanczos) to obtain bounds for the non diagonal entry. An added adantage is that we are able to choose δ so that $\gamma_j \beta_j > 0$ and therefore $p_j(\lambda) = q_j(\lambda)$. This can be done by starting with $\delta = 1$ and restarting the algorithm with a larger value of δ as soon as we find a value of j for which $\gamma_j \beta_j \leq 0$.

Regarding expressions for the remainder, we can do exactly the same as for symmetric Lanczos. We can write

$$R(f) = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b p_N(\lambda)^2 \, d\alpha(\lambda).$$

However, we know that $p_N(\lambda) = \pm q_N(\lambda)$ and

$$\int_a^b p_N(\lambda) q_N(\lambda) \, d\alpha(\lambda) = 1.$$

This shows that the sign of the integral in the remainder can be computed using the algorithm and we have the following result.

Theorem 4.3 *Suppose f is such that $f^{(2n)}(\eta) > 0$, $\forall n$, $\forall \eta$, $a < \eta < b$. Then, the quadrature rule 3.7 gives a lower bound if*

$$\prod_{j=1}^{N} sgn(\hat{r}_j^T r_j) = 1,$$

and an upper bound otherwise. In both cases, we have

$$|R(f)| = \frac{f^{(2N)}(\eta)}{(2N)!}.$$

For Gauss–Radau and Gauss–Lobatto we cannot do the same thing. Bounds can be obtained if we choose the initial vectors (e.g. δ) such that the measure is positive and increasing. In this case we are in exactly the same framework as for the symmetric case and the same results are obtained. Note however that it is not easy to make this choice a priori. Some examples are given in Section 6. A way to proceed is to start with $\delta = 1$ and to restart the algorithm 4.2 with a larger value of δ whenever we have $\gamma_j \beta_j \leq 0$.

4.3 The block case

Now, we consider the block Lanczos algorithm, see [10],[16]. Let X_0 be an $n \times 2$ given matrix, such that $X_0^T X_0 = I_2$ (chosen as U defined before). Let $X_{-1} = 0$ be an $n \times 2$ matrix. Then

$$\Omega_j = X_{j-1}^T A X_{j-1},$$

$$R_j = AX_{j-1} - X_{j-1}\Omega_j - X_{j-2}\Gamma_{j-1}^T,$$
$$X_j\Gamma_j = R_j$$

The last step is the QR decomposition of R_j such that X_j is $n \times 2$ with $X_j^T X_j = I_2$ and Γ_j is 2×2. The matrix Ω_j is 2×2 and Γ_j is upper triangular.

It may happen that R_j is rank deficient and in that case Γ_j is singular. The solution of this problem is given in [10]. One of the columns of X_j can be chosen arbitrarily. To complete the algorithm, we choose this column to be orthogonal with the previous block vectors X_k. We can for instance choose another vector (randomly) and orthogonalize it against the previous ones.

This algorithm generates a sequence such that

$$X_j^T X_i = \delta_{ij} I_2,$$

where I_2 is the 2×2 identity matrix.

Proposition 4.3

$$X_i = \sum_{k=0}^{i} A^k X_0 C_k^{(i)},$$

where $C_k^{(i)}$ *are* 2×2 *matrices.*

Proof: The proof is given by induction.

We define a matrix polynomial $p_i(\lambda)$, a 2×2 matrix, as

$$p_i(\lambda) = \sum_{k=0}^{i} \lambda^k C_k^{(i)}.$$

Thus, we have the following result.

Theorem 4.4

$$X_i^T X_j = \int_a^b p_i(\lambda)^T d\alpha(\lambda) p_j(\lambda) = \delta_{ij} I_2.$$

Proof:

Using the orthogonality of the X_is, we can write

$$\begin{aligned}
\delta_{ij} I_2 = X_i^T X_j &= \left(\sum_{k=0}^{i} (C_k^{(i)})^T X_0^T A^k \right) \left(\sum_{l=0}^{j} A^l X_0 C_l^{(j)} \right) \\
&= \sum_{k,l} (C_k^{(i)})^T X_0^T Q \Lambda^{k+l} Q^T X_0 C_l^{(j)} \\
&= \sum_{k,l} (C_k^{(i)})^T \alpha \Lambda^{k+l} \alpha^T C_l^{(j)} \\
&= \sum_{k,l} (C_k^{(i)})^T \left(\sum_{m=1}^{n} \lambda_m^{k+l} \alpha_m \alpha_m^T \right) C_l^{(j)} \\
&= \sum_{m=1}^{n} \left(\sum_k \lambda_m^k (C_k^{(i)})^T \right) \alpha_m \alpha_m^T \left(\sum_l \lambda_m^l C_l^{(j)} \right).
\end{aligned}$$

The p_js can be considered as matrix orthogonal polynomials for the (matrix) measure α. To compute the polynomials, we need to show that the following recurrence relation holds.

Theorem 4.5 *The matrix valued polynomials p_j satisfy*

$$p_j(\lambda)\Gamma_j = \lambda p_{j-1}(\lambda) - p_{j-1}(\lambda)\Omega_j - p_{j-2}(\lambda)\Gamma_{j-1}^T,$$

$$p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv I_2,$$

where λ is a scalar.

Proof: From the previous definition, it is easily shown by induction that p_j can be generated by the given (matrix) recursion.

As we have seen before this can be written as

$$\lambda[p_0(\lambda), \ldots, p_{N-1}(\lambda)] = [p_0(\lambda), \ldots, p_{N-1}(\lambda)]J_N + [0, \ldots, 0, p_N(\lambda)\Gamma_N],$$

and as $P(\lambda) = [p_0(\lambda), \ldots, p_{N-1}(\lambda)]^T$,

$$J_N P(\lambda) = \lambda P(\lambda) - [0, \ldots, 0, p_N(\lambda)\Gamma_N]^T,$$

with J_N defined by 3.8.

Most of the following results on the properties of the matrix polynomials are derived from [1].

Proposition 4.4 *The eigenvalues of J_N are the zeroes of $det[p_N(\lambda)]$.*

Proof: Let μ be a zero of $det[p_N(\lambda)]$. As the rows of $p_N(\mu)$ are linearly dependent, there exists a vector v with two components such that

$$v^T p_N(\mu) = 0.$$

This implies that

$$\mu[v^T p_0(\mu), \ldots, v^T p_{N-1}(\mu)] = [v^T p_0(\mu), \ldots, v^T p_{N-1}(\mu)]J_N.$$

Therefore μ is an eigenvalue of J_N. $det[p_N(\lambda)]$ is a polynomial of degree $2N$ in λ. Hence, there exists $2N$ zeroes of the determinant and therefore all eigenvalues are zeroes of $det[p_N(\lambda)]$.

Proposition 4.5 *For λ and μ real, we have the analog of the Christoffel–Darboux identity (see [21]) :*

$$p_{N-1}(\mu)\Gamma_N^T p_N^T(\lambda) - p_N(\mu)\Gamma_N p_{N-1}^T(\lambda) = (\lambda - \mu)\sum_{m=0}^{N-1} p_m(\mu)p_m^T(\lambda). \tag{4.3}$$

Proof: From the previous results, we have

$$\Gamma_{j+1}^T p_{j+1}^T(\lambda) = \lambda p_j^T(\lambda) - \Omega_{j+1} p_j^T(\lambda) - \Gamma_j p_{j-1}^T(\lambda),$$

$$p_{j+1}(\mu)\Gamma_{j+1} = \mu p_j(\mu) - p_j(\mu)\Omega_{j+1} - p_{j-1}(\mu)\Gamma_j^T.$$

Multiplying the first relation by $p_j(\mu)$ on the left and the second one by $p_j^T(\lambda)$ on the right gives

$$p_j(\mu)\Gamma_{j+1}^T p_{j+1}^T(\lambda) - p_{j+1}(\mu)\Gamma_{j+1} p_j^T(\lambda) =$$
$$= (\lambda - \mu) p_j(\mu) p_j^T(\lambda) - p_j(\mu)\Gamma_j p_{j-1}^T(\lambda) + p_{j-1}(\mu)\Gamma_j^T p_j^T(\lambda).$$

Summing these equalities, some terms cancel and we get the desired result.

In particular, if we choose $\lambda = \mu$ in 4.3, we have that $p_N(\lambda)\Gamma_N p_{N-1}^T(\lambda)$ is symmetric.

Proposition 4.6

$$\sum_{m=0}^{N-1} u_s^T p_m(\lambda_s) p_m^T(\lambda_r) u_r = \delta_{rs}.$$

Proof: If we set $\lambda = \lambda_s$ and $\mu = \lambda_r$ and multiply the Christoffel–Darboux identity 4.3 on the left by u_s^T and on the right by u_r, we have $p_N^T(\lambda_r)u_r = 0$ and $p_N^T(\lambda_s)u_s = 0$, and we get the result if $\lambda_s \neq \lambda_r$. Let

$$K_{N-1}(\mu, \lambda) = \sum_{m=0}^{N-1} p_m(\mu) p_m^T(\lambda).$$

As $p_0(\lambda) = I_2$, $K_{N-1}(\lambda, \lambda)$ is a symmetric positive definite matrix and therefore defines a scalar product. If λ_r is a multiple eigenvalue, there exist linearly independent eigenvectors that we could orthonormalize. If λ_r is an eigenvalue of multiplicity q_r, there exist q_r linearly independent vectors $v_r^1, \ldots, v_r^{q_r}$ with two components such that the vectors $P(\lambda_r)v_r^j, j = 1, \ldots, q_r$ are the eigenvectors associated with λ_r. We can certainly find a set of vectors $(w_r^1, \ldots, w_r^{q_r})$ spanning the same subspace as $(v_r^1, \ldots, v_r^{q_r})$ and such that

$$(w_r^k)^T K_{N-1}(\lambda_r, \lambda_r) w_r^l = \delta_{kl}.$$

This property is nothing else than the orthogonality relation of the eigenvectors.

Proposition 4.7

$$\sum_{r=1}^{2N} p_i^T(\lambda_r) u_r u_r^T p_l(\lambda_r) = \delta_{il} I_2, \quad i = 0, \ldots, N-1, \quad l = 0, \ldots, N-1.$$

Proof: Note that the eigenvectors of J_N are linearly independent. We take a set of N vectors with two elements : $\{y_0, \ldots, y_{N-1}\}$, and write

$$[y_0^T, \ldots, y_{N-1}^T]^T = \sum_{r=1}^{2N} \alpha_r [u_r^T p_0(\lambda_r), \ldots, u_r^T p_{N-1}(\lambda_r)]^T,$$

or

$$y_l = \sum_{r=1}^{2N} p_l^T(\lambda_r) u_r \alpha_r, \quad l = 0, \ldots, N-1.$$

Multiplying on the left by $u_s^T p_l(\lambda_s)$ and summing :

$$\sum_{l=0}^{N-1} u_s^T p_l(\lambda_s) y_l = \alpha_s, \quad s = 1, \ldots, 2N.$$

Therefore,

$$y_i = \sum_{r=1}^{2N} \sum_{l=0}^{N-1} p_i^T(\lambda_r) u_r u_r^T p_l(\lambda_r) y_l.$$

This gives the desired result.

To prove that the block quadrature rule is exact for polynomials of degree up to $2N-1$, we cannot use the same method as for the scalar case where the given polynomial is factored because of commutativity problems. Therefore, we take another approach that has been used in a different setting in [2]. The following results are taken from [2].

We will consider all the monomials $\lambda^k, k = 1, \ldots, 2N-1$. Let M_k the moment matrix, be defined as

$$M_k = \int_a^b \lambda^k \, d\alpha(\lambda).$$

We write the (matrix) orthonormal polynomials p_j as

$$p_j(\lambda) = \sum_{k=0}^{j} p_k^{(j)} \, \lambda^k,$$

$p_k^{(j)}$ being a matrix of order 2. Then, we have

$$\int_a^b p_j^T(\lambda) \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T \int_a^b \lambda^k \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T M_k,$$

and more generally

$$\int_a^b p_j^T(\lambda) \lambda^q \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T M_{k+q}.$$

We write these equations for $j = N-1$. Note that because of the orthogonality of the polynomials, we have

$$\int_a^b p_{N-1}^T(\lambda) \lambda^q \, d\alpha(\lambda) = 0, \quad q = 0, \ldots, N-2.$$

Let H_N be the block Hankel matrix of order $2N$, defined as

$$H_N = \begin{pmatrix} M_0 & \cdots & M_{N-1} \\ \vdots & & \vdots \\ M_{N-1} & \cdots & M_{2N-2} \end{pmatrix}.$$

Then

$$H_N \begin{pmatrix} p_0^{(N-1)} \\ \vdots \\ p_{N-2}^{(N-1)} \\ p_{N-1}^{(N-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \int_a^b p_{N-1}^T(\lambda)\lambda^{N-1}\, d\alpha(\lambda) \end{pmatrix}.$$

We introduce some additional notation. Let L_N be a block upper triangular matrix of order $2N$,

$$L_N = \begin{pmatrix} p_0^{(0)} & p_0^{(1)} & \cdots & p_0^{(N-1)} \\ 0 & p_1^{(1)} & \cdots & p_1^{(N-1)} \\ & & \ddots & \vdots \\ & & & p_{N-1}^{(N-1)} \end{pmatrix}.$$

Let V_N be a $4N \times 2N$ matrix defined in block form as

$$V_N = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_l \end{pmatrix},$$

where B_j is a $q_j \times 2N$ matrix,

$$B_j = \begin{pmatrix} I_2 & \mu_j I_2 & \cdots & \mu_j^{N-1} I_2 \\ \vdots & \vdots & \vdots & \vdots \\ I_2 & \mu_j I_2 & \cdots & \mu_j^{N-1} I_2 \end{pmatrix},$$

and the μ_j are the eigenvalues of A.

Let K_i^j be a $2q_i \times 2q_j$ matrix

$$K_i^j = \begin{pmatrix} K_{N-1}(\mu_i, \mu_j) & \cdots & K_{N-1}(\mu_i, \mu_j) \\ \vdots & \vdots & \vdots \\ K_{N-1}(\mu_i, \mu_j) & \cdots & K_{N-1}(\mu_i, \mu_j) \end{pmatrix},$$

and

$$K = \begin{pmatrix} K_1^1 & K_1^2 & \ldots & K_1^l \\ K_2^1 & \ldots & \ldots & K_2^l \\ \vdots & & & \vdots \\ K_l^1 & \ldots & \ldots & K_l^l \end{pmatrix}.$$

Proposition 4.8

$$V_N L_N = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_l \end{pmatrix},$$

where C_j is a $2q_j \times 2N$ matrix,

$$C_j = \begin{pmatrix} p_0(\mu_j) & p_1(\mu_j) & \cdots & p_{N-1}(\mu_j) \\ \vdots & \vdots & \vdots & \vdots \\ p_0(\mu_j) & p_1(\mu_j) & \cdots & p_{N-1}(\mu_j) \end{pmatrix}.$$

Proof: This is straightforward by the definition of the polynomials $p_j(\lambda)$.

Proposition 4.9

$$L_N^T H_N L_N = I.$$

Proof: the generic term of $H_N L_N$ is

$$(H_N L_N)_{ij} = \sum_{s=1}^{j} M_{s+i-2}\ p_{s-1}^{(j-1)}.$$

Therefore the generic term of $L_N^T H_N L_N$ is

$$(L_N^T H_N L_N)_{ij} = \sum_{r=1}^{i}\sum_{s=1}^{j} \int_a^b (p_{r-1}^{(i-1)})^T \lambda^{s+r-2}\ d\alpha(\lambda) p_{s-1}^{(j-1)}.$$

Splitting the power of λ we can easily see that this is

$$(L_N^T H_N L_N)_{ij} = \int_a^b p_{i-1}^T(\lambda)\ d\alpha(\lambda)\ p_{j-1}(\lambda).$$

Therefore because of the orthonormality properties, we have

$$(L_N^T H_N L_N)_{ij} = \begin{cases} I_2 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

This result implies that

$$H_N^{-1} = L_N L_N^T.$$

Proposition 4.10

$$V_N L_N (V_N L_N)^T = K.$$

Proof: this is just using the definition of K_i^j.

Now, we define a $2N \times 4N$ matrix W_N^T whose only non zero components in row i are in position $(i, 2i-1)$ and $(i, 2i)$ and are successively the two components of

$$(w_1^1)^T, \ldots, (w_1^{q_1})^T, (w_2^1)^T, \ldots, (w_l^{q_l})^T$$

. Then because of the way the w_i^j are constructed we have

Proposition 4.11

$$W_N^T K W_N = I.$$

Proposition 4.12 *$W_N^T V_N$ is a non singular $2N \times 2N$ matrix.*

Proof:

$$W_N^T V_N H_N^{-1} V_N^T W_N = W_N^T V_N L_N L_N^T V_N^T W_N = W_N^T K W_N = I.$$

This shows that $W_N^T V_N$ is non singular.

Then, we have the main result

Theorem 4.6 *The quadrature rule 3.9 or 3.10 is exact for polynomials of order less than or equal to* $2N-1$.

Proof: From Proposition 4.12, we have

$$H_N^{-1} = (W_N^T V_N)^{-1}(V_N^T W_N)^{-1}.$$

Therefore,

$$H_N = (V_N^T W_N)(W_N^T V_N).$$

By identification of the entries of the two matrices we have,

$$M_k = \sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \right) \mu_i^k, \quad k = 0, \ldots, 2N-2.$$

It remains to prove that the quadrature rule is exact for $k = 2N-1$. As we have,

$$H_{N+1} \begin{pmatrix} p_0^{(N)} \\ \vdots \\ p_{N-1}^{(N)} \\ p_N^{(N)} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \int_a^b p_N^T(\lambda)\lambda^N \ d\alpha(\lambda) \end{pmatrix}.$$

Writing the $(N-1)$th block row of this equality, we get

$$M_{2N-1}\ p_N^{(N)} = -\sum_{r=0}^{N-1} M_{N+r-1}\ p_r^{(N)}.$$

We have proved before that

$$M_{N+r-1} = \sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} w_i^j (w_i^j)^T \right) \mu_i^{N+r-1}.$$

By substitution, we get

$$M_{2N-1}\ p_N^{(N)} = -\sum_{r=0}^{N-1} \sum_{i=1}^{l} \sum_{j=1}^{q_i} w_i^j (w_i^j)^T \mu_i^{N+r-1} p_r^{(N)}.$$

We use the fact that

$$(w_i^j)^T \sum_{r=0}^{N-1} \mu_i^r p_r^{(N)} = (w_i^j)^T p_N(\mu_i) - (w_i^j)^T \mu_i^N p_N^{(N)},$$

and

$$(w_i^j)^T p_N(\mu_i) = 0.$$

This shows that

$$M_{2N-1} p_N^{(N)} = \sum_{i=1}^{l} \sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \mu_i^{2N-1} p_N^{(N)}.$$

As $p_N^{(N)}$ is non singular, we get the result.

To obtain expressions for the remainder, we would like to use a similar approach as for the scalar case. However there are some differences, as the quadrature rule is exact for polynomials of order $2N-1$ and we have $2N$ nodes, we cannot interpolate with an Hermite polynomial and we have to use a Lagrange polynomial. By Theorems 2.1.1.1 and 2.1.4.1 of [18], there exists a polynomial q of degree $2N-1$ such that

$$q(\lambda_j) = f(\lambda_j), \quad j = 1, \ldots, 2N$$

and

$$f(x) - q(x) = \frac{s(x) f^{(2N)}(\xi(x))}{(2N)!},$$

where

$$s(x) = (x - \lambda_1) \cdots (x - \lambda_{2N}).$$

If we can apply the mean value theorem, the remainder $R(f)$ which is a 2×2 matrix can be written as

$$R(f) = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b s(\lambda)\ d\alpha(\lambda).$$

Unfortunately s does not have a constant sign over the interval $[a, b]$. Therefore this representation formula for the remainder is of little practical use for obtaining bounds with the knowledge of the sign of the entries of the remainder.

It is easy to understand why we cannot directly obtain bounds with this block approach. We must use $W = (e_i\ e_j)$ and the block Lanczos algorithm with $X_0 = W$. For the block Lanczos algorithm we multiply successively A with the Lanczos vectors. If A is sparse, most of the components of these products are 0 for the first few iterates of the algorithm. Therefore, it is likely that at the beginning, the estimates that we will get for the non diagonal entries will be 0. This explains why we cannot directly obtain upper or lower bounds with Gauss, Gauss–Radau or Gauss–Lobatto. A way to avoiding this difficulty is to use $W = (e_i + e_j\ e_j)$ but this cannot be done since $X_0^T X_0 \neq I_2$.

However, we will see in the numerical experiments that the estimates we get are often quite good.

5 Application to the inverse of a matrix

In this Section we consider obtaining analytical bounds for the entries of the inverse of a given matrix and simplifying the algorithms to compute numerical bounds and approximations.

We consider

$$f(\lambda) = \frac{1}{\lambda}, \quad 0 < a < b,$$

and hence

$$f^{(2n+1)}(\lambda) = -(2n+1)!\ \lambda^{-(2n+2)},$$

and

$$f^{(2n)}(\lambda) = (2n)!\ \lambda^{-(2n+1)}.$$

Therefore, the even derivatives are positive on $[a, b]$ and the odd derivatives are negative which implies that we can apply Theorems 3.1, 3.2 and 3.3.

Consider a dense non singular matrix $A = (a_{ij})_{i,j=1,\ldots,m}$. We choose $u = x_0 = e_i$ and we apply the Lanczos algorithm. From results of the first iteration we can obtain analytical results. The first step of the Lanczos algorithm gives us

$$\omega_1 = e_i^T A e_i = a_{ii},$$

$$\gamma_1 x_1 = r_1 = (A - \omega_1 I) e_i.$$

Let s_i be defined by

$$s_i^2 = \sum_{j \neq i} a_{ji}^2,$$

and let

$$d_i^T = (a_{1,i}, \ldots, a_{i-1,i}, 0, a_{i+1,i}, \ldots, a_{m,i}).$$

Then

$$\gamma_1 = s_i, \quad x_1 = \frac{1}{s_i} d_i.$$

From this, we have

$$\omega_2 = \frac{1}{s_i^2} \sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i}.$$

From this data, we compute the Gauss rule and get a lower bound on the diagonal element:

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \omega_2 \end{pmatrix},$$

$$J_2^{-1} = \frac{1}{\omega_1 \omega_2 - \gamma_1^2} \begin{pmatrix} \omega_2 & -\gamma_1 \\ -\gamma_1 & \omega_1 \end{pmatrix}.$$

The lower bound is given by

$$\frac{\omega_2}{\omega_1 \omega_2 - \gamma_1^2} = \frac{\sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i}}{a_{i,i} \sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i} - \left(\sum_{k \neq i} a_{k,i}^2 \right)^2}.$$

Note that this bound does not depend on the eigenvalues a and b.

Now, we consider the Gauss–Radau rule. Then,

$$\tilde{J}_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & x \end{pmatrix},$$

the eigenvalues λ are the roots of $(\omega_1 - \lambda)(x - \lambda) - \gamma_1^2 = 0$, which gives the relation

$$x = \lambda + \frac{\gamma_1^2}{\omega_1 - \lambda}.$$

To obtain an upper bound we set $\lambda = a$. The solution is

$$x = x_a = a + \frac{\gamma_1^2}{\omega_1 - a}.$$

For the Gauss–Lobatto rule, we have the same problem except that we want $\tilde{J}_2$ to have a and b as eigenvalues. This leads to solving the following linear system,

$$\begin{pmatrix} \omega_1 - a & -1 \\ \omega_1 - b & -1 \end{pmatrix} \begin{pmatrix} x \\ \gamma_1^2 \end{pmatrix} = \begin{pmatrix} a\omega_1 - a^2 \\ b\omega_1 - b^2 \end{pmatrix}.$$

Solving this system and computing the $(1,1)$ element of the inverse gives

$$\frac{a + b - \omega_1}{ab}.$$

Hence we have the following result.

Theorem 5.1 *We have the following bounds*

$$\frac{\sum_{k\neq i} \sum_{l\neq i} a_{k,i} a_{k,l} a_{l,i}}{a_{i,i} \sum_{k\neq i} \sum_{l\neq i} a_{k,i} a_{k,l} a_{l,i} - \left(\sum_{k\neq i} a_{k,i}^2\right)^2} \leq (A^{-1})_{i,i}$$

$$\frac{a_{i,i} - b + \frac{s_i^2}{b}}{a_{i,i}^2 - a_{i,i} b + s_i^2} \leq (A^{-1})_{i,i} \leq \frac{a_{i,i} - a + \frac{s_i^2}{a}}{a_{i,i}^2 - a_{i,i} a + s_i^2}$$

$$(A^{-1})_{i,i} \leq \frac{a + b - a_{ii}}{ab}.$$

It is not too easy to derive analytical bounds from the block Lanczos algorithm as we have to compute repeated inverses of 2×2 matrices.

It is much easier to use the non–symmetric Lanczos method with the Gauss–Radau rule. We are looking at the sum of the (i,i) and (i,j) elements of the inverse. Let

$$t_i = \gamma_1 \beta_1 = \sum_{k\neq i} a_{k,i}(a_{k,i} + a_{k,j}) - a_{i,j}(a_{i,j} + a_{i,i}).$$

Then, the computations are essentially the same as for the diagonal case.

Theorem 5.2 *For $(A^{-1})_{i,j} + (A^{-1})_{i,i}$ we have the two following estimates*

$$\frac{a_{i,i} + a_{i,j} - a + \frac{t_i}{a}}{(a_{i,i} + a_{i,j})^2 - a(a_{i,i} + a_{i,j}) + t_i}, \quad \frac{a_{i,i} + a_{i,j} - b + \frac{t_i}{b}}{(a_{i,i} + a_{i,j})^2 - b(a_{i,i} + a_{i,j}) + t_i}.$$

If $t_i \geq 0$, the first expression with a gives an upper bound and the second one with b a lower bound. Then, we have to subtract the bounds for the diagonal term to get bounds on $(A^{-1})_{i,j}$.

The previous results can be compared with those obtained by other methods in [17]. Results can also be obtained for sparse matrices taking into account the sparsity structure.

In the computations using the Lanczos algorithm for the Gauss, Gauss–Radau and Gauss–Lobatto rules, we need to compute the $(1,1)$ element of the inverse of a tridiagonal matrix. This may be done in many different ways, see for instance [13]. Here, we will show that we can compute this element of the inverse incrementally as we go through the Lanczos algorithm and we obtain the estimates for very few additional operations. This is stated in the following theorem where b_j stands for the bounds for Lanczos iteration j and the ω_js and the γ_js are generated by Lanczos.

Theorem 5.3 *The following algorithm yields a lower bound b_j of A_{ii}^{-1} by the Gauss quadrature rule, a lower bound $\bar{b}_j$ and an upper bound $\hat{b}_j$ through the Gauss–Radau quadrature rule and an upper bound $\breve{b}_j$ through the Gauss–Lobatto rule.*

Let $x_{-1} = 0$ and $x_0 = e_i$, $\omega_1 = a_{ii}$, $\gamma_1 = \|(A - \omega_1 I)e_i\|$, $b_1 = \omega_1^{-1}$, $d_1 = \omega_1$, $c_1 = 1$, $\hat{d}_1 = \omega_1 - a$, $\bar{d}_1 = \omega_1 - b$, $x_1 = (A - \omega_1 I)e_i/\gamma_1$.

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

$$r_j = (A - \omega_j I)x_{j-1} - \gamma_{j-1}x_{j-2},$$

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

$$\gamma_j = \|r_j\|,$$

$$x_j = \frac{r_j}{\gamma_j},$$

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

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

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

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

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

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

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

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

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

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

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

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

Proof: We have from 3.4

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

Let $x_N^T = (0 \ \ldots \ 0 \ \gamma_N)$, so that

$$J_{N+1} = \begin{pmatrix} J_N & x_N \\ x_N^T & \omega_{N+1} \end{pmatrix}.$$

Letting

$$\tilde{J} = J_N - \frac{x_N x_N^T}{\omega_{N+1}},$$

the upper left block of J_{N+1}^{-1} is $\tilde{J}^{-1}$. This can be obtained through the use of the Sherman–Morrison formula (see [11]),

$$\tilde{J}^{-1} = J_N^{-1} + \frac{(J_N^{-1} x_N)(x_N^T J_N^{-1})}{\omega_{N+1} - x_N^T J_N^{-1} x_N}.$$

Let $j_N = J_N^{-1} e_N$ be the last column of the inverse of J_N. With this notation, we have

$$\tilde{J}^{-1} = J_N^{-1} + \frac{\gamma_N^2 j_N j_N^T}{\omega_{N+1} - \gamma_N^2 (j_N)_N}.$$

Therefore, it is clear that we only need the first and last elements of the last column of the inverse of J_N. This can be obtained using the Cholesky decomposition of J_N. It is easy to check that if we define

$$d_1 = \omega_1, \quad d_i = \omega_i - \frac{\gamma_{i-1}^2}{d_{i-1}}, \quad i = 2, \ldots, N$$

then

$$(j_N)_1 = (-1)^{N-1} \frac{\gamma_1 \cdots \gamma_{N-1}}{d_1 \cdots d_N}, \quad (j_N)_N = \frac{1}{d_N}.$$

When we put all these results together we get the proof of the Theorem.

The algorithm is essentially the same for the non–symmetric case. The modified algorithm is the following, using $f_1 = 1$:

$$\begin{aligned} r_j &= (A - \omega_j I) x_{j-1} - \beta_{j-1} x_{j-2}, \\ \hat{r}_j &= (A - \omega_j I) \hat{x}_{j-1} - \gamma_{j-1} \hat{x}_{j-2}, \\ \omega_j &= \hat{x}_{j-1}^T A x_{j-1}, \\ \gamma_j \beta_j &= \hat{r}_j^T r_j., \end{aligned}$$

$$x_j = \frac{r_j}{\gamma_j},$$

$$\hat{x}_j = \frac{\hat{r}_j}{\beta_j},$$

$$b_j = b_{j-1} + \frac{\gamma_{j-1}\beta_{j-1}c_{j-1}f_{j-1}}{d_{j-1}(\omega_j d_{j-1} - \gamma_{j-1}\beta_{j-1})},$$

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

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

$$f_j = f_{j-1}\frac{\beta_{j-1}}{d_{j-1}},$$

$$\hat{d}_j = \omega_j - a - \frac{\gamma_{j-1}\beta_{j-1}}{\hat{d}_{j-1}},$$

$$\bar{d}_j = \omega_j - b - \frac{\gamma_{j-1}\beta_{j-1}}{\bar{d}_{j-1}},$$

$$\hat{\omega}_j = a + \frac{\gamma_j\beta_j}{\hat{d}_j},$$

$$\bar{\omega}_j = b + \frac{\gamma_j\beta_j}{\bar{d}_j},$$

$$\hat{b}_j = b_j + \frac{\gamma_j\beta_j c_j f_j}{d_j(\hat{\omega}_j d_j - \gamma_j\beta_j)},$$

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

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

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

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

We have the analog for the block case. For simplicity we only consider the Gauss rule. Then, in 3.8

$$J_N = \begin{pmatrix} \Omega_0 & \Gamma_0^T & & & \\ \Gamma_0 & \Omega_1 & \Gamma_1^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-3} & \Omega_{N-2} & \Gamma_{N-2}^T \\ & & & \Gamma_{N-2} & \Omega_{N-1} \end{pmatrix},$$

and

$$J_{N+1} = \begin{pmatrix} J_N & x_N \\ x_N^T & \omega_{N+1} \end{pmatrix},$$

with

$$x_N^T = (0 \ \ldots \ 0 \ \Gamma_{N-1}),$$

and

$$\tilde{J} = J_n - x_N \Omega_N^{-1} x_N^T.$$

Here, we use the Sherman–Morrison–Woodbury formula (see [11]) which is a generalization of the formula we used before. Then,

$$\tilde{J}^{-1} = J_N^{-1} + J_N^{-1} x_N (\Omega_N - x_N^T J_N^{-1} x_N)^{-1} x_N^T J_N^{-1}.$$

In order to compute all the elements, we need a block Cholesky decomposition of J_N. We obtain the following algorithm which gives a 2×2 matrix B_i, the block element of the inverse that we need.

Theorem 5.4 *Let* $B_0 = \Omega_0^{-1}, \quad D_0 = \Omega_0, \quad C_0 = I$, *for* $i = 1, \ldots$ *we compute*

$$B_i = B_{i-1} + C_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T (\Omega_i - \Gamma_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T)^{-1} \Gamma_{i-1} D_{i-1}^{-1} C_{i-1}^T,$$

$$D_i = \Omega_i - \Gamma_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T,$$

$$C_i = C_{i-1} \Gamma_{i-1}^T D_{i-1}^{-1}.$$

These recurrences for 2×2 matrices can be easily computed. Hence, the approximations can be computed as we apply the block Lanczos algorithm.

Given these algorithms to compute the estimates, we see that almost all of the operations are a result of the Lanczos algorithm. Computing the estimate has a complexity independent of the problem size.

To compute a diagonal entry, the Lanczos algorithm needs per iteration the following operations: 1 matrix–vector product, $4N$ multiplies and $4N$ adds. To compute two diagonal entries and a non diagonal one, the block Lanczos algorithm needs per iteration: 2 matrix–vector products, $9N$ multiplies, $7N$ adds plus the QR decomposition which is $8N$ flops (see [11]). The non–symmetric Lanczos algorithm requires per iteration: 1 matrix–vector product, $6N$ multiplies and $6N$ adds.

Therefore, if we only want to estimate diagonal elements it is best to use the Lanczos algorithm. If we want to estimate a non diagonal element, it is best to use the block Lanczos algorithm since we get three estimates in one run while for the non–symmetric Lanczos method we need also to have an estimate of a diagonal element. The number of flops is the same but for the block Lanczos we have three estimates instead of two with the non–symmetric Lanczos. On the other hand, the non–symmetric Lanczos gives bounds but the block Lanczos yields only estimates.

As we notice before, we can compute bounds for the non diagonal elements by considering $\frac{1}{2}(e_i + e_j)^T A^{-1}(e_i + e_j)$. For this, we need to run three Lanczos algorithms that is per iteration: 3 matrix–vector products, $12N$ multiplies and $12N$ adds to get 3 estimates. This no more operations than in the block Lanczos case but here, we can get bounds.

With the non–symmetric Lanczos, we have 2 bounds with 2 matrix–vector products, $10N$ multiplies and $10N$ adds.

One can ask why in the case of the inverse are we not solving the linear system

$$Au = e_i$$

to obtain the i^{th} column of the inverse at once. To our knowledge, it is not possible then to tell if the estimates are upper or lower bounds. Moreover this can be easily added to the algorithm of Theorem 5.3.

Let $Q_N = [x_0, \ldots, x_{N-1}]$ be the matrix of the Lanczos vectors. Then we have the approximate solution

$$u_N = Q_N y_N,$$

where y_N is the solution of

$$J_N y_N = Q_N^T e_i = e_1.$$

This tridiagonal linear system can be easily solved incrementally from the LDL^T decomposition, see [11]. This yields a variant of the conjugate gradient algorithm. We give numerical examples in Section 6 and show that our methods give better bounds.

6 Numerical examples

In this Section, we first describe the examples we use and then we give numerical results for some specific functions f.

6.1 Description of the examples

First we look at examples of small dimension for which the inverses are known. Then, we will turn to larger examples arising from the discretization of partial differential equations. Most of the numerical computations have been done with Matlab 3.5 on an Apple Macintosh Powerbook 170 and a few ones on a Sun workstation.

Example 1.

First, we consider

$$A = I + uu^T, \quad u^T = (1, 1, \ldots, 1)$$

This matrix has two distinct eigenvalues 1 and $n+1$. Therefore, the minimal polynomial is of degree 2 and the inverse can be written as

$$A^{-1} = \frac{2+n}{1+n}\, I - \frac{1}{1+n} A.$$

Example 2.

The entries of the Hilbert matrix are given by $\frac{1}{i+j-1}$. We consider a matrix of dimension 5 which is

$$A(\alpha) = \alpha I_5 + \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix}.$$

The inverse of $A(0)$ is

$$A^{-1} = \begin{pmatrix} 25 & -300 & 1050 & -1400 & 630 \\ -300 & 4800 & -18900 & 26880 & -1260 \\ 1050 & -18900 & 79380 & -117600 & 56700 \\ -1400 & 26880 & -117600 & 179200 & -88200 \\ 630 & -1260 & 56700 & -88200 & 44100 \end{pmatrix},$$

and the eigenvalues of $A(0)$ are

$$(3.288\ 10^{-6},\ 3.059\ 10^{-4},\ 1.141\ 10^{-2},\ 0.209,\ 1.567)$$

Example 3.

We take an example of dimension 10,

$$A = \frac{1}{11} \begin{pmatrix} 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 9 & 18 & 16 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\ 8 & 16 & 24 & 21 & 18 & 15 & 12 & 9 & 6 & 3 \\ 7 & 14 & 21 & 28 & 24 & 20 & 16 & 12 & 8 & 4 \\ 6 & 12 & 18 & 24 & 30 & 25 & 20 & 15 & 10 & 5 \\ 5 & 10 & 15 & 20 & 25 & 30 & 24 & 18 & 12 & 6 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 21 & 14 & 7 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 & 16 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}.$$

It is easily seen (cf. [13]) that the inverse is a tridiagonal matrix

$$A^{-1} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

The eigenvalues of A are therefore distinct and given by

$$(0.2552,\ 0.2716,\ 0.3021,\ 0.3533,\ 0.4377,\ 0.5830,\ 0.8553,\ 1.4487,\ 3.1497,\ 12.3435).$$

Example 4.

We have

$$A = \begin{pmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix},$$

whose inverse is

$$A^{-1} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 3 & 3 & 3 \\ 1 & 3 & 5 & 5 & 5 \\ 1 & 3 & 5 & 7 & 7 \\ 1 & 3 & 5 & 7 & 9 \end{pmatrix},$$

whose eigenvalues are

$$(0.0979,\ 0.8244,\ 2.0000,\ 3.1756,\ 3.9021).$$

Example 5.

We use a matrix of dimension 10 constructed with the TOEPLITZ function of Matlab,

$$A = 21I_{10} + \text{TOEPLITZ}(1:10).$$

This matrix has distinct eigenvalues but most of them are very close together:

(0.5683, 14.7435, 18.5741, 19.5048, 20.0000, 20.2292, 20.3702, 20.4462, 20.4875, 65.0763).

Example 6.

This example is the matrix arising from the 5–point finite difference of the Poisson equation in a unit square. This gives a linear system

$$Ax = b$$

of order 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

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

For $m = 6$, the minimum and maximum eigenvalues are 0.3961 and 7.6039.

Example 7.

This example arises from the 5–point finite difference approximation of the following equation in a unit square,

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

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.

For $m = 6$, the minimum and maximum eigenvalues are 0.4354 and 6828.7.

6.2 Results for a polynomial function

To numerically check some of the previous theorems, f was chosen as a polynomial of degree q,

$$f(\lambda) = \prod_{i=1}^{q} (\lambda - i).$$

We chose Example 6 with $m = 6$, that is a matrix of order 36.

1) We compute the $(2,2)$ element of $f(A)$ and we vary the order of the polynomial. In the next table, we give, as a function of the degree q of the polynomial, the value of N to have an "exact" result (4 digits in Matlab) for the Gauss rule.

q	2	3	4	5	6
N	2	2	3	3	4

From these results, we can conclude that the maximum degree for which the results are exact is $q = 2N - 1$, as predicted by the theory.

For the Gauss–Radau rule, we get

q	2	3	4	5	6	7	8	9
N	2	2	2	3	3	4	4	5

From this we deduce $q = 2N$ as predicted.

For the Gauss–Lobatto rule, we have

q	2	3	4	5	6	7	8	9
N	1	1	2	2	3	3	4	4

This shows that $q = 2N + 1$ which is what we expect.

2) If we consider the block case to compute the $(3,1)$ element of the polynomial, we get the same results, therefore the block Gauss rule is exact for the degree $2N - 1$, the block Gauss–Radau rule is exact for degree $2N$ and the block Gauss–Lobatto is exact for degree $2N + 1$.

3) The same is also true for the non–symmetric Lanczos algorithm if we want to compute the sum of the $(3,1)$ and $(3,3)$ elements.

6.3 Bounds for the inverse

6.3.1 Diagonal elements

Now, we turn to some numerical experiments using Matlab on the examples described above. Usually the results will be given using the "short" format of Matlab. In the following results, Nit denotes the number of iterations of the Lanczos algorithm. This corresponds to N for the Gauss and Gauss–Radau rules and $N - 1$ for the Gauss–Lobatto rule.

Example 1.

Because of the properties of the matrix we should get the answer in two steps. We have $\omega_0 = 2$ and $\gamma_0 = n - 1$, therefore, the lower bound from Gauss–Radau is $\frac{n}{n+1}$ and the upper bound is $\frac{n}{n+1}$, the exact result. If we look at the lower bound from the Gauss rule, we find the same value. This is also true for the numerical experiments as well as for Gauss–Lobatto.

Example 2.

Let us consider $(A(0)^{-1})_{33}$ whose exact value is 79380. The Gauss rule, as a function of the degree of the quadrature, gives

Results from Gauss rule

Nit=1	2	3	4	5
5	26.6	1808.3	3666.8	79380

The Gauss–Radau rule gives upper and lower bounds. For a and b, we use the computed values from the EIG function of Matlab.

Results from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	23.74	1801.77	3666.58	3559.92	79380
up bnd	257674	216812	202814	79380	79380

Results from Gauss–Lobatto rule

Nit=1	2	3	4	5
265330	216870	202860	79268	79380

The results are not as good as expected. The exact results should have been obtained for $Nit = 3$. The discrepancy comes from round off errors, particularly for the lower bound, because of the eigenvalue distribution of A and a poor convergence rate of the Lanczos algorithm in this case. To see how this is related to the conditioning of A, let us vary α. For simplicity we consider only the Gauss rule. The following tables give results for different values of α and the $(3,3)$ element of the inverse. The exact values are $(\alpha = 0.01,\ 70.3949)$, $(\alpha = 0.1,\ 7.7793)$, $(\alpha = 1,\ 0.9054)$.

Lower bound from Gauss rule for $\alpha = 0.01$

Nit=2	3	4	5
20.5123	69.7571	70.3914	70.3949

Lower bound from Gauss rule for $\alpha = 0.1$

Nit=2	3	4	5
6.7270	7.7787	7.7793	7.7793

Lower bound from Gauss rule for $\alpha = 1$

Nit=2	3	4	5
0.9040	0.9054	0.9054	0.9054

We see that when A is well conditioned, the numerical results follow the theory. The discrepancies probably arise from the poor convergence of the smallest eigenvalues of J_N towards those of A.

Example 3.

We are looking for bounds for $(A^{-1})_{55}$ whose exact value is, of course, 2.

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5	6	7
0.3667	1.3896	1.7875	1.9404	1.9929	1.9993	2

Results for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5	6	7
b_1	1.3430	1.7627	1.9376	1.9926	1.9993	2.0117	2
b_2	3.0330	2.2931	2.1264	2.0171	2.0020	2.0010	2

Upper bounds for $(A^{-1})_{55}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7
3.1341	2.3211	2.1356	2.0178	2.0021	2.0001	2

In this example 5 or 6 iterations should be sufficient, so we are a little off the theory.

Example 4.

We look at bounds for $(A^{-1})_{55}$ whose exact value is 4.5

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5
1	2	3	4	4.5

Lower and upper bounds for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	1.3910	2.4425	3.4743	4.5	4.5
up bnd	5.8450	4.7936	4.5257	4.5	4.5

Upper bounds for $(A^{-1})_{55}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5
7.8541	5.2361	4.6180	4.5	4.5

Example 5.

We get for $(A^{-1})_{55}$, whose value is 0.0595,

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5
0.0455	0.0511	0.0523	0.0585	0.0595

Lower and upper bounds for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	0.0508	0.0522	0.0582	0.0595	0.0595
up bnd	0.4465	0.0721	0.0595	0.0595	0.0595

Upper bounds for $(A^{-1})_{55}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5
1.1802	0.0762	0.0596	0.0595	0.0595

Because some eigenvalues are very close together, we get the exact answers a little sooner than it is predicted by theory.

Example 6.

Consider $m = 6$. Then we have a system of order 36 and we look for bounds on $(A^{-1})_{18,18}$ whose value is 0.3515. There are 19 distinct eigenvalues, therefore we should get the exact answer in about 10 iterations for Gauss and Gauss–Radau and 9 iterations for Gauss–Lobatto.

Lower bounds for $(A^{-1})_{18,18}$ from the Gauss rule

Nit=1	2	3	4	8	9
0.25	0.3077	0.3304	0.3411	0.3512	0.3515

Lower and upper bounds for $(A^{-1})_{18,18}$ from the Gauss–Radau rule

	Nit=1	2	3	4	8	9
lw bnd	0.2811	0.3203	0.3366	0.3443	0.3514	0.3515
up bnd	0.6418	0.4178	0.3703	0.3572	0.3515	0.3515

Upper bounds for $(A^{-1})_{18,18}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	8
1.3280	0.4990	0.3874	0.3619	0.3515

Now, we consider $m = 16$ which gives a matrix of order 256. We want to compute bounds for the $(125, 125)$ element whose value is 0.5604. In this case there are 129 distinct eigenvalues, so we should find the exact answer in about 65 iterations at worst. These computations for a larger problem have been done on a Sun Sparcstation 1+. We find the following results.

Lower bounds for $(A^{-1})_{125,125}$ from the Gauss rule

Nit=2	3	4	5	6	7	8	9	10	20
0.3333	0.3929	0.4337	0.4675	0.4920	0.5084	0.5201	0.5301	0.5378	0.5600

Lower and upper bounds for $(A^{-1})_{125,125}$ from the Gauss–Radau rule

	Nit=2	3	4	5	6	7	8	10	20
lw bnd	0.3639	0.4140	0.4514	0.4804	0.5006	0.5146	0.5255	0.5414	0.5601
up bnd	1.5208	1.0221	0.8154	0.7130	0.6518	0.6139	0.5925	0.5730	0.5604

Upper bounds for $(A^{-1})_{125,125}$ from the Gauss–Lobatto rule

Nit=2	3	4	5	6	7	8	9	10	18
2.1011	1.2311	0.8983	0.7585	0.6803	0.6310	0.6012	0.5856	0.5760	0.5604

We have very good estimates much sooner than predicted. This is because there are distinct eigenvalues which are very close together.

We also ran two other examples with $m = 10$ and $m = 20$ that show that the number of iterations to reach a "correct" value with four exact digits grows like m. This can be expected from the Lanczos method.

Example 7.

We took $m = 6$ as in the previous example. So we have a matrix of dimension 36. The $(2, 2)$ element of the inverse has an "exact" value of 0.3088 and there are 23 distinct eigenvalues so that the exact answer should be obtained after 12 iterations but the matrix is ill conditioned. We get the following results:

Lower bounds for $(A^{-1})_{2,2}$ from the Gauss rule

Nit=1	2	3	4	5	6	8	10	12	15
0.25	0.2503	0.2510	0.2525	0.2553	0.2609	0.2837	0.2889	0.3036	0.3088

Lower and upper bounds for $(A^{-1})_{2,2}$ from the Gauss–Radau rule

	Nit=2	3	4	5	6	7	8	10	12	15
lw bnd	0.2504	0.2516	0.2538	0.2583	0.2699	0.2821	0.2879	0.2968	0.3044	0.3088
up bnd	0.5375	0.5202	0.5121	0.5080	0.5060	0.5039	0.5013	0.3237	0.3098	0.3088

Upper bounds for $(A^{-1})_{2,2}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5	6	8	10	12	15
2.2955	0.5765	0.5289	0.5156	0.5093	0.5065	0.5020	0.3237	0.3098	0.3088

6.3.2 Non diagonal elements with non–symmetric Lanczos

Here, we use the non–symmetric Lanczos algorithm to get estimates on non diagonal elements.

Example 1.

The matrix is of dimension $n = 5$. All the non diagonal elements are $-1/6 = -0.1667$ and the diagonal elements are equal to 0.8333.

We compute the sum of the $(2, 2)$ and $(2, 1)$ elements (e.g. $\delta = 1$), that is 0.6667. With the Gauss rule, after 1 iteration we get 0.3333 and after 2 iterations 0.6667. With Gauss–Radau, we obtain the result in one iteration as well as with Gauss–Lobatto.

We note that for $\delta = 1$, the measure is not positive but for $\delta = 2$ the measure is positive and increasing.

Example 2.

Let us consider first the Hilbert matrix and $(A(0)^{-1})_{2,1}$ whose exact value is 79380. We compute the sum of the $(2, 2)$ and $(2, 1)$ elements, (i.e. $\delta = 1$) that is 4500. The

Gauss rule, as a function of the number of iterations, gives

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
1.2	-21.9394	73.3549	667.1347	4500

Consider now the non–symmetric Gauss–Radau rule.

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5
b1	-17.5899	73.1917	667.1277	667.0093	4500
b2	144710	155040	51854	4500	4500

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5
142410	155570	51863	3789.2	4500

Note that the measure is positive and increasing therefore, we obtain a lower bound with the Gauss rule, b_1 is a lower bound and b_2 an upper bound with Gauss–Radau and Gauss–Lobatto gives an upper bound.

Again the results are not so good. Consider now the results of the non–symmetric Gauss rule with a better conditioned problem by looking at $A(0.1)$. The sum of the elements we compute is 5.1389. In the last line we indicate if the product of the non diagonal coefficients is positive (p) or negative (n). If it is positive we should have a lower bound, an upper bound otherwise. Note that in this case, the measure is positive but not increasing.

Estimates from the non–symmetric Gauss rule for $\alpha = 0.1$

Nit=1	2	3	4	5
1.0714	6.1735	5.1341	5.1389	5.1389
p	n	p	p	p

We see that the algorithm is able to determine if it is computing a lower or an upper bound.

Estimates from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4
b1	6.5225	5.1338	5.1389	5.1389
b2	5.0679	5.2917	5.1390	5.1389

We remark that b_1 and b_2 are alternatively upper and lower bounds.

Estimates from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4
5.0010	5.3002	5.1390	5.1389

Again, we do not have an upper bound with Gauss–Lobatto but the results oscillate around the exact value. In this case, this can be fixed by using a value $\delta = 3$ that gives a positive increasing measure.

Example 3.

We are looking for estimates for the sum of the $(2,2)$ and $(2,1)$ elements whose exact value is 1. First, we use $\delta = 1$ for which the measure is positive but not increasing.

Estimates from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	7
0.4074	0.6494	0.8341	0.9512	0.9998	1.0004	1
p	p	p	p	p	n	p

Estimates from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5	6	7
b1	0.6181	0.8268	0.9488	0.9998	1.0004	1.0001	1
b2	2.6483	1.4324	1.0488	1.0035	1.0012	0.9994	1

Estimates from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7	8
3.2207	1.4932	1.0529	1.0036	1.0012	0.9993	0.9994	1

Here we have a small problem at the end near convergence, but the estimates are quite good. Note that for $\delta = 4$ the measure is positive and increasing.

Example 4.

This example illustrates some of the problems that can happen with the non–symmetric Lanczos algorithm. We would like to compute the sum of the $(2,2)$ and $(2,1)$ elements that is 2. After 2 iterations we have a breakdown of the Lanczos algorithm as $\gamma\beta = 0$. The same happens at the first iteration for the Gauss–Radau rule and at the second one for the Gauss–Lobatto rule. Choosing a value of δ different from 1 cures the breakdown problem. We can obtain bounds with a value $\delta = 10$ (with a positive and increasing measure). Then the value we are looking for is 1.55 and the results follow.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
0.5263	0.8585	1.0333	1.4533	1.55

Bounds from the non–symmetric Gauss–Radau rule

	Nit=2	3	4
b1	1.0011	1.2771	1.55
b2	1.9949	1.5539	1.55

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=2	3	4
2.2432	1.5696	1.55

Example 5.

The sum of the $(2,2)$ and $(2,1)$ elements is 0.6158.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
0.0417	0.0974	0.4764	0.6155	0.6158
p	p	p	p	p

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4
b1	0.0847	0.4462	0.6154	0.6158
b2	0.9370	0.6230	0.6158	0.6158

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4
1.1261	0.6254	0.6159	0.6158

Example 6.

We consider $m = 6$, then, we have a system of order 36 and we look for estimates of the sum of the $(2,2)$ and $(2,1)$ elements which is 0.4471. Remember there are 19 distinct eigenvalues.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	7	8	9
0.3333	0.4000	0.4262	0.4369	0.4419	0.4446	0.4461	0.4468	0.4471
p	p	p	p	p	p	p	p	p

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5	6	7	8	9
b1	0.3675	0.4156	0.4320	0.4390	0.4436	0.4456	0.4466	0.4470	0.4471
b2	0.7800	0.5319	0.4690	0.4537	0.4490	0.4476	0.4472	0.4472	0.4471

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7	8	9	10
1.6660	0.6238	0.4923	0.4596	0.4505	0.4480	0.4473	0.4472	0.4472	0.4471

Example 7.

We took $m = 6$ as in the previous example. So we have a matrix of dimension 36. The sum of the $(2,2)$ and $(2,1)$ elements of the inverse is 0.3962 and there are 23 distinct eigenvalues. We get the following results:

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	8	10	12	15
0.3333	0.3336	0.3340	0.3348	0.3363	0.3396	0.3607	0.3689	0.3899	0.3962
p	p	p	p	p	p	p	p	p	p

Bounds from the non–symmetric Gauss–Radau rule

	Nit=2	3	4	5	6	8	10	12	15
b1	0.3337	0.3343	0.3355	0.3380	0.3460	0.3672	0.3803	0.3912	0.3962
b2	0.6230	0.5930	0.5793	0.5725	0.5698	0.5660	0.4078	0.3970	0.3962

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	8	10	12	15
2.2959	0.6898	0.6081	0.5850	0.5746	0.5703	0.5664	0.4078	0.3970	0.3962

Finally, one can ask why we do not store the Lanczos vectors x_j and compute an approximation to the solution of $Au = e_i$. This can be done doing the following. Let

$$Q_N = [x_0, \ldots, x_{N-1}].$$

If we solve

$$J_N y_N = e_1,$$

then the approximate solution is given by $Q_N y_N$.

Unfortunately this does not give bounds and even the approximations are not as good as with our algorithms. Consider Example 5 and computing the fifth column of the inverse. For the element (1,5) whose "exact" value is 0.460, we find

Estimates from solving the linear system

Nit= 2	3	4	5	6
-0.0043	-0.0046	0.0382	0.0461	0.0460

By computing bounds for the sum of the (5,5) and (1,5) elements and subtracting the bounds for the (5,5) element, we obtain

Bounds from the Gauss–Radau quadrature rules

	Nit= 2	3	4
lw bnd	0.0048	0.0451	0.0460
up bnd	0.0551	0.0473	0.0460

We see that we get good bounds quite fast.

6.3.3 Non diagonal elements with block Lanczos

Here, we use the block Lanczos algorithm to get estimates on non diagonal elements. Unfortunately, most of the examples are too small to be of interest as for matrices of dimension 5 we cannot go further than 2 block iterations. Nevertheless, let us look at the results.

Example 1.

Consider a matrix of dimension $n = 5$. Then all the non diagonal elements are $-1/6 = -0.1667$.

We compute the $(2,1)$ element. With the block Gauss rule, after 2 iterations we get -0.1667. With block Gauss–Radau, we get the exact answer in 1 iteration as well as with Gauss–Lobatto.

Example 2.

The $(2,1)$ element of the inverse of the Hilbert matrix $A(0)$ of dimension 5 is -300.

With the block Gauss rule, after 2 iterations we find -90.968. Note that this is an upper bound. With block Gauss–Radau, 2 iterations give -300.2 as a lower bound and -300 as an upper bound. Block Gauss–Lobatto gives -5797 as a lower bound.

Now we consider $A(0.1)$ for which the $(2,1)$ element of the inverse is -1.9358. After 2 iterations, block Gauss gives -2.2059 a lower bound and block Gauss–Radau and Gauss–Lobatto give the exact answer.

Example 3.

The $(2,1)$ element is -1, the $(3,1)$ element is 0. After 2 iterations we get the exact answers with Gauss as well as with Gauss–Radau. Gauss–Lobatto gives -0.0609, a lower bound. Three iterations give the exact answer.

Example 4.

The $(2,1)$ element is $1/2$. After 2 iterations, we have 0.3182 which is a lower bound. Gauss–Radau gives 0.2525 and 0.6807. Gauss–Lobatto gives 0.7236 which is an upper bound.

Example 5.

The $(2,1)$ element is 0.2980. Two iterations give 0.2329, a lower bound and 3 iterations give the exact answer. Gauss–Radau and Gauss–Lobatto also give the exact answer in 3 iterations.

Example 6.

This example uses $n = 36$. The $(2,1)$ element is 0.1040. Remember that we should do about 10 iterations. We get the following figures.

Estimates from the block Gauss rule

Nit=2	3	4	5	6	7	8
0.0894	0.0974	0.1008	0.1024	0.1033	0.1037	0.1040

Estimates from the block Gauss–Radau rule

Nit=2	3	4	5	6	7	8
0.0931	0.0931	0.1017	0.1029	0.1035	0.1038	0.1040
0.1257	0.1103	0.1059	0.1046	0.1042	0.1041	0.1040

Estimates from the block Gauss–Lobatto rule

Nit=2	3	4	5	6	7	8
0.1600	0.1180	0.1079	0.1051	0.1041	0.1043	0.1041

Note that here everything works. Gauss gives a lower bound, Gauss–Radau a lower and an upper bound and Gauss–Lobatto an upper bound.

Example 7.

We would like to obtain estimates of the $(2,1)$ whose value is 0.0874. We get the following results.

Estimates from the block Gauss rule

Nit=2	4	6	8	10	12	14	15
0.0715	0.0716	0.0722	0.0761	0.0789	0.0857	0.0873	0.0874

Estimates from the block Gauss–Radau rule

Nit=2	4	6	8	10	12	14	15
0.0715	0.0717	0.0731	0.0782	0.0831	0.0861	0.0873	0.0874
0.1375	0.1216	0.1184	0.1170	0.0894	0.0876	0.0874	0.0874

Estimates from the block Gauss–Lobatto rule

Nit=2	4	6	8	10	12	14
0.1549	0.1237	0.1185	0.1176	0.0894	0.0876	0.0874

Note that in this example we obtain bounds. Now, to illustrate what we said before about the estimates being 0 for some iterations, we would like to estimate the $(36,1)$ element of the inverse which is 0.005.

Estimates from the block Gauss rule

Nit=2	4	6	8	10	11
0.	0.	0.0023	0.0037	0.0049	0.0050

Estimates from the block Gauss–Radau rule

Nit=2	4	6	8	10	11
0.	0.	0.0023	0.0037	0.0049	0.0050
0.	0.	0.0024	0.0050	0.0050	0.0050

Estimates from the block Gauss–Lobatto rule

Nit=2	4	6	8	10
0.	0.	0.0022	0.0043	0.0050

6.3.4 Dependence on the eigenvalue estimates

In this sub–Section, we numerically investigate how the bounds and estimates of the Gauss–Radau rules depend on the accuracy of the estimates of the eigenvalues of A. We take Example 6 with $m = 6$ and look at the results given by the Gauss–Radau rule as a function of a and b. Remember that in the previous experiments we took for a and b the values returned by the EIG function of Matlab.

It turns out that the estimates are only weakly dependent of the values of a and b (for this example). We look at the number of iterations needed to obtain an upper for the element $(18, 18)$ with four exact digits and with an "exact" value of b. The results are given in the following table.

a=10^{-4}	10^{-2}	0.1	0.3	0.4	1	6
15	13	11	11	8	8	9

We have the same properties when b is varied.

Therefore, we see that the estimation of the extreme eigenvalues does not seem to matter very much and can be obtained with a few iterations of Lanczos or with the Gerschgorin circles.

6.4 Bounds for the exponential

In this Section we are looking for bounds of diagonal elements of the exponential of the matrices of some of the examples.

6.4.1 diagonal elements

Example 1

We consider the $(2, 2)$ element whose value is 82.8604. With Gauss, Gauss–Radau and Gauss–Lobatto we obtain the exact value in 2 iterations.

Example 2

We would like to compute the $(3, 3)$ element whose value is 1.4344. Gauss gives the answer in 3 iterations, Gauss–Radau and Gauss–Lobatto in 2 iterations.

Example 3

The $(5, 5)$ entry is $4.0879\ 10^4$. Gauss obtains the exact value in 4 iterations, Gauss–Radau and Gauss–Lobatto in 3 iterations.

Example 6

We consider the $(18, 18)$ element whose value is 197.8311. We obtain the following results.

Lower bounds from the Gauss rule

Nit=2	3	4	5	6	7
159.1305	193.4021	197.5633	197.8208	197.8308	197.8311

Lower and upper bounds from the Gauss–Radau rule

	Nit=2	3	4	5	6
lw bnd	182.2094	196.6343	197.7779	197.8296	197.8311
up bnd	217.4084	199.0836	197.8821	197.8325	197.8311

Upper bounds from the Gauss–Lobatto rule

Nit=2	3	4	5	6	7
273.8301	203.4148	198.0978	197.8392	197.8313	197.8311

We remark that to compute diagonal elements of the exponential the convergence rate is quite fast.

6.4.2 non diagonal elements

Here we consider only Example 6 and we would like to compute the element $(2,1)$ whose value is -119.6646. First, we use the block Lanczos algorithm which give the following results.

Results from the block Gauss rule

Nit=2	3	4	5	6
-111.2179	-119.0085	-119.6333	-119.6336	-119.6646

Results from the block Gauss–Radau rule

	Nit=2	3	4	5	6
b1	-115.9316	-119.4565	-119.6571	-119.6644	-119.6646
b2	-122.2213	-119.7928	-119.6687	-119.6647	-119.6646

Results from the block Gauss–Lobatto rule

Nit=2	3	4	5	6
-137.7050	-120.6801	-119.7008	-119.6655	-119.6646

Now, we use the non–symmetric Lanczos algorithm. The sum of the $(2,2)$ and $(2,1)$ elements of the exponential is 73.9023.

Results from the non–symmetric Gauss rule

Nit=2	3	4	5	6	7
54.3971	71.6576	73.7637	73.8962	73.9021	73.9023

Results from the non–symmetric Gauss–Radau rule

	Nit=2	3	4	5	6
b1	65.1847	73.2896	73.8718	73.9014	73.9023
b2	84.0323	74.6772	73.9323	73.9014	73.9023

Results from the non–symmetric Gauss–Lobatto rule

Nit=2	3	4	5	6	7
113.5085	77.2717	74.0711	73.9070	73.9024	73.9023

6.5 Bounds for other functions

When one uses domain decomposition methods for matrices arising from the finite difference approximation of partial differential equations in a rectangle, it is known that the matrix

$$A = \sqrt{T + \frac{1}{4}T^2},$$

where T is the matrix of the one dimensional Laplacian, is a good preconditioner for the Schur complement matrix. It is interesting to see if we can estimate some elements of the matrix A to generate a Toeplitz tridiagonal approximation to A.

We have

$$T = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

and we choose an example of dimension 100. We estimate the $(50,50)$ element whose exact value is 1.6367 with the Gauss–Radau rule. We obtain the following results.

Estimates from the Gauss–Radau rule

	Nit=2	3	4	5	10	15	20
lw	1.6014	1.6196	1.6269	1.6305	1.6355	1.6363	1.6365
up	1.6569	1.6471	1.6430	1.6409	1.6378	1.6371	1.6369

We estimate the non diagonal elements by using the block Gauss rule. We choose the $(49,50)$ element.

Estimates from the block Gauss rule

Nit=2	3	4	5	10	15	20
-0.6165	-0.6261	-0.6302	-0.6323	-0.6354	-0.6361	-0.6363

Now, we construct a Toeplitz tridiagonal matrix C whose elements are chosen From the estimates given at the fifth iteration. We took the average of the Gauss–Radau values for the diagonal (1.6357) and -0.6323 for the non diagonal elements. We look at the spectrum of $C^{-1}A$. The condition number is 13.35 the minimum eigenvalue being 0.0837 and the maximum one being 1.1174, but there are 86 eigenvalues between 0.9 and the maximum eigenvalue. Therefore, the matrix C (requiring only 5 iterations of some Lanczos algorithms) seems to be a good preconditioner for A which is itself a good preconditioner for the Schur complement.

7 Conclusions

We have shown how to obtain bounds (or in certain cases estimates) of the entries of a function of a symmetric positive definite matrix. The proposed algorithms use the Lanczos algorithm to estimate diagonal entries and either the non–symmetric Lanczos or block Lanczos algorithms for the non diagonal entries.

The algorithms are particularly simple for the inverse of a matrix. Analytical bounds are derived by considering one or two iterations of these algorithms. We have seen in the numerical experiments that very good approximations are obtained in a few iterations.

References

[1] F.V. Atkinson, "Discrete and continuous boundary problems", (1964) Academic Press

[2] S. Basu, N.K. Bose, "Matrix Stieltjes series and network models", SIAM J. Math. Anal. v14 n2 (1983) pp 209–222

[3] 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

[4] P. Davis, P. Rabinowitz, "Methods of numerical integration", Second Edition (1984) Academic Press

[5] W. Gautschi, "Construction of Gauss–Christoffel quadrature formulas", Math. Comp. 22 (1968) pp 251–270

[6] W. Gautschi, "Orthogonal polynomials– constructive theory and applications", J. of Comp. and Appl. Math. 12 & 13 (1985) pp 61–76

[7] G.H. Golub, J.H. Welsch, "Calculation of Gauss quadrature rule" Math. Comp. 23 (1969) pp 221–230

[8] G.H. Golub, "Some modified matrix eigenvalue problems", SIAM Review v15 n2 (1973) pp 318–334

[9] G.H. Golub, "Bounds for matrix moments", Rocky Mnt. J. of Math., v4 n2 (1974) pp 207–211

[10] G.H. Golub, R. Underwood, "The block Lanczos method for computing eigenvalues", in Mathematical Software III, Ed. J. Rice (1977) pp 361–377

[11] G.H. Golub, C. van Loan, "Matrix Computations", Second Edition (1989) Johns Hopkins University Press

[12] R. Haydock, "Accuracy of the recursion method and basis non–orthogonality", Computer Physics Communications 53 (1989) pp 133–139

[13] G. Meurant, "A review of the inverse of tridiagonal and block tridiagonal matrices", SIAM J. Matrix Anal. Appl. v13 n3 (1992) pp 707–728

[14] C.M. Nex, "Estimation of integrals with respect to a density of states", J. Phys. A, v11 n4 (1978) pp 653–663

[15] C.M. Nex, "The block Lanczos algorithm and the calculation of matrix resolvents", Computer Physics Communications 53 (1989) pp 141–146

[16] D.P. O'Leary, "The block conjugate gradient algorithm and related methods", Linear Alg. and its Appl. v29 (1980) pp 293–322

[17] P.D. Robinson, A. Wathen, "Variational bounds on the entries of the inverse of a matrix", IMA J. of Numer. Anal. v12 (1992) pp 463–486

[18] J. Stoer, R. Bulirsch, "Introduction to numerical analysis", Second Edition (1983) Springer Verlag

[19] G.W. Struble, "Orthogonal polynomials: variable–signed weight functions", Numer. Math v5 (1963) pp 88–94

[20] G. Szegö, "Orthogonal polynomials", Third Edition (1974) American Mathematical Society

[21] H.S. Wilf, "Mathematics for the physical sciences", (1962) Wiley