A Cholesky LR algorithm for the positive definite symmetric diagonal-plus-semiseparable eigenproblem

Bor Plestenjak
Department of Mathematics
University of Ljubljana
Slovenia

Ellen Van Camp and Marc Van Barel
Department of Computer Science
Katholieke Universiteit Leuven
Belgium

Outline

- Introduction
- Cholesky LR algorithm
- Laguerre's shift
- Implementation
- Numerical examples
- Conclusions

Introduction

- semiseparable: every submatrix from the lower or upper triangular part has rank at most 1.
- diagonal-plus-semiseparable (DPSS): the sum $D + S$ of a diagonal D and a semiseparable S.
- Givens-vector representation of a DPSS matrix is based on a vector $f = [f_1, \ldots, f_n]^T$, Givens rotations $G_i = \begin{bmatrix} c_i & -s_i \\ s_i & c_i \end{bmatrix}$, $i = 1, \ldots, n-1$, and a diagonal $d = [d_1, \ldots, d_n]^T$ as

$$D + S = \begin{bmatrix} c_1 f_1 + d_1 & c_2 s_1 f_1 & \cdots & c_{n-1} s_{n-2:1} f_1 & s_{n-1:1} f_1 \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \cdots & c_{n-1} s_{n-2:1} f_2 & s_{n-1:2} f_2 \\ \vdots & & \ddots & & \vdots \\ c_{n-1} s_{n-2:1} f_1 & c_{n-1} s_{n-2:2} f_2 & \cdots & c_{n-1} f_{n-1} + d_{n-1} & s_{n-1} f_{n-1} \\ s_{n-1:1} f_1 & s_{n-1:2} f_2 & \cdots & s_{n-1} f_{n-1} & f_n + d_n \end{bmatrix},$$

  where $s_{a:b} = s_a s_{a-1} \cdots s_b$. When they appear, we assume that $c_n = 1$ and $s_n = 0$.
- We denote

$$D + S = \mathrm{diag}(d) + \mathrm{Giv}(c, s, f).$$

Our goal is to compute the smallest (or all) eigenvalues of a s.p.d. DPSS matrix.

Motivation

Let A be an $n \times n$ symmetric matrix. One can find an orthogonal matrix Q such that $B = Q^T A Q$ is tridiagonal in $\mathcal{O}(n^3)$ flops. Next, the eigendecomposition of B is computed in $\mathcal{O}(n^2)$ flops.

Instead of the tridiagonal structure one can use DPSS matrices.

Vandebril, Van Camp, Van Barel, Mastronardi (2004):

- For an arbitrary diagonal matrix D there exists an orthogonal matrix Q such that $Q^T A Q = D + S$ is a DPSS matrix. The reduction is $\mathcal{O}(n^2)$ more expensive than in the tridiagonal case.
- The reduction algorithm has a Lanczos-Ritz convergence behaviour and performs a kind of nested subspace iteration at each step. A good choice of the diagonal can compensate slower reduction.

Algorithms for the eigendecomposition of a symmetric DPSS matrix:

- Chandrasekaran, Gu (2004), in Numer. Math.: divide and conquer,
- Mastronardi, Van Camp, Van Barel (2003), tech. report: divide and conquer,
- Bini, Gemignani, Pan (2003), tech. report: QR algorithm,
- Fasino (2004), tech. report: QR algorithm,
- Van Camp, Delvaux, Van Barel, Vandebril, Mastronardi (2005), tech. report: implicit QR algorithm,
- Mastronardi, Van Camp, Van Barel, Vandebril (2004), tech. report: computation of eigenvectors

Cholesky LR algorithm

Let A be a s.p.d. matrix.

$A_0 = A$
$k = 0, 1, 2, \ldots$
choose shift σ_k
$A_k - \sigma_k I = V_k V_k^T$ (Cholesky decomposition, V_k is lower-triangular)
$A_{k+1} = V_k^T V_k + \sigma_k I$

Two steps of the zero shift Cholesky LR are equivalent to one step of the zero shift QR.

The shift σ_k should be such that $A_k - \sigma_k I$ is positive definite.

When applied to a s.p.d. DPSS matrix $D + S$, the shift can be included into the diagonal part.

Grad, Zakrajšek (1972): Cholesky LR with Laguerre's shifts for the symmetric tridiagonal matrices.

DPSS is invariant to Cholesky LR

Theorem: Let $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ be a s.p.d DPSS matrix.

1. If $A = VV^T$ is the Cholesky decomposition of A, then $V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d})$,

2. If $B = V^T V$, then B is a s.p.d. DPSS matrix $B = \mathrm{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \mathrm{diag}(d)$.

$$A_4 = \begin{bmatrix} c_1 f_1 + d_1 & \times & \times & \times \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \times & \times \\ c_3 s_2 s_1 f_1 & c_3 s_2 f_2 & c_3 f_3 + d_3 & \times \\ s_3 s_2 s_1 f_1 & s_3 s_2 f_2 & s_3 f_3 & f_4 + d_4 \end{bmatrix}.$$

$$V_4 = \begin{bmatrix} c_1 \widetilde{f}_1 + \widetilde{d}_1 & & & \\ c_2 s_1 \widetilde{f}_1 & c_2 \widetilde{f}_2 + \widetilde{d}_2 & & \\ c_3 s_2 s_1 \widetilde{f}_1 & c_3 s_2 \widetilde{f}_2 & c_3 \widetilde{f}_3 + \widetilde{d}_3 & \\ s_3 s_2 s_1 \widetilde{f}_1 & s_3 s_2 \widetilde{f}_2 & s_3 \widetilde{f}_3 & \widetilde{f}_4 + \widetilde{d}_4 \end{bmatrix}.$$

$$B_4 = \begin{bmatrix} \widehat{c}_1 \widehat{f}_1 + d_1 & \times & \times & \times \\ \widehat{c}_2 \widehat{s}_1 \widehat{f}_1 & \widehat{c}_2 \widehat{f}_2 + d_2 & \times & \times \\ \widehat{c}_3 \widehat{s}_2 \widehat{s}_1 \widehat{f}_1 & \widehat{c}_3 \widehat{s}_2 \widehat{f}_2 & \widehat{c}_3 \widehat{f}_3 + d_3 & \times \\ \widehat{s}_3 \widehat{s}_2 \widehat{s}_1 \widehat{f}_1 & \widehat{s}_3 \widehat{s}_2 \widehat{f}_2 & \widehat{s}_3 \widehat{f}_3 & \widehat{f}_4 + d_4 \end{bmatrix}.$$

Cholesky decomposition

$$A_4 = \begin{bmatrix} c_1 f_1 + d_1 & \times & \times & \times \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \times & \times \\ c_3 s_2 s_1 f_1 & c_3 s_2 f_2 & c_3 f_3 + d_3 & \times \\ s_3 s_2 s_1 f_1 & s_3 s_2 f_2 & s_3 f_3 & f_4 + d_4 \end{bmatrix}, \quad V_4 = \begin{bmatrix} c_1 \widetilde{f}_1 + \widetilde{d}_1 & & & \\ c_2 s_1 \widetilde{f}_1 & c_2 \widetilde{f}_2 + \widetilde{d}_2 & & \\ c_3 s_2 s_1 \widetilde{f}_1 & c_3 s_2 \widetilde{f}_2 & c_3 \widetilde{f}_3 + \widetilde{d}_3 & \\ s_3 s_2 s_1 \widetilde{f}_1 & s_3 s_2 \widetilde{f}_2 & s_3 \widetilde{f}_3 & \widetilde{f}_4 + \widetilde{d}_4 \end{bmatrix}.$$

We compare the diagonal and the main subdiagonal of A and VV^T:

$$\begin{aligned} c_k f_k + d_k &= \sum_{j=1}^{k-1} (c_k s_{k-1} \cdots s_j \widetilde{f}_j)^2 + (c_k \widetilde{f}_k + \widetilde{d}_k)^2 &&= c_k^2 q_k + (c_k \widetilde{f}_k + \widetilde{d}_k)^2, \\ c_{k+1} s_k f_k &= \sum_{j=1}^{k-1} c_k c_{k+1} s_k (s_{k-1} \cdots s_j \widetilde{f}_j)^2 c_{k+1} s_k \widetilde{f}_k (c_k \widetilde{f}_k + \widetilde{d}_k) &&= c_k c_{k+1} s_k q_k + c_{k+1} s_k \widetilde{f}_k (c_k \widetilde{f}_k + \widetilde{d}_k), \end{aligned}$$

where

$$q_k := \sum_{j=1}^{k-1} (s_{k-1} s_{k-2} \cdots s_j \widetilde{f}_j)^2.$$

The solution is

$$\widetilde{f}_k = \frac{f_k - c_k q_k}{\sqrt{d_k + c_k (f_k - c_k q_k)}}, \qquad \widetilde{d}_k = \frac{d_k}{\sqrt{d_k + c_k (f_k - c_k q_k)}}, \qquad k = 1, \ldots, n.$$

Algorithm for the Cholesky decomposition

If $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ is s.p.d., then $A = VV^T$ for $V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d})$.

function $[\widetilde{f}, \widetilde{d}] = \mathrm{Cholesky}(c, s, f, d)$

$c_n = 1$

$q_1 = 0$

for $k = 1, \ldots, n$:

$$z_k = f_k - c_k \cdot q_k$$
$$y_k = \sqrt{d_k + c_k \cdot z_k}$$
$$\widetilde{f}_k = z_k / y_k$$
$$\widetilde{d}_k = d_k / y_k$$
$$q_{k+1} = s_k^2 (q_k + \widetilde{f}_k^2)$$

Flops: $11n + \mathcal{O}(1)$.

Algorithm for the Cholesky decomposition

If $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ is s.p.d., then $A = VV^T$ for $V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d})$.

function $[\widetilde{f},\ \widetilde{d}] = \mathrm{Cholesky}(c,\ s,\ f,\ d)$
$c_n = 1$
$q_1 = 0$
for $k = 1, \ldots, n$:
$z_k = f_k - c_k \cdot q_k$
$y_k = \sqrt{d_k + c_k \cdot z_k}$
$\widetilde{f}_k = z_k / y_k$
$\widetilde{d}_k = d_k / y_k$
$q_{k+1} = s_k^2 (q_k + \widetilde{f}_k^2)$

Flops: $11n + \mathcal{O}(1)$.

It follows from $c_k f_k + d_k = c_k^2 q_k + (c_k \widetilde{f}_k + \widetilde{d}_k)^2$ that y_k is in fact the diagonal element of V:

$$c_k \widetilde{f}_k + \widetilde{d}_k = \sqrt{d_k + c_k (f_k - c_k q_k)} = \sqrt{d_k + c_k z_k} = y_k.$$

A negative or zero value under the square root appears if A is not positive definite.

Equations for V^TV

The product $B = V^TV$ is a s.p.d. DPSS matrix. The lower triangular elements of B are

$$\begin{aligned} b_{kk} &= (c_k\widetilde{f}_k + \widetilde{d}_k)^2 + (s_k\widetilde{f}_k)^2, \\ b_{jk} &= s_k s_{k+1}\cdots s_{j-1}\widetilde{f}_k(\widetilde{f}_j + c_j\widetilde{d}_j), \end{aligned}$$

where $k = 1, \ldots, n$ and $j > k$.

$B = \mathrm{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \mathrm{diag}(d)$. For $k = 1, \ldots, n-1$ we have

$$\widehat{s}_k^2\widehat{f}_k^2 = \sum_{j=k+1}^{n} b_{jk}^2 = \widetilde{f}_k^2 p_k,, \qquad \text{where} \qquad p_k = \sum_{j=k+1}^{n} (s_k s_{k+1}\cdots s_{j-1})^2(\widetilde{f}_j + c_j\widetilde{d}_j)^2.$$

For p_k we can apply the recursion $p_n = 0$ and $p_k = s_k^2\left(p_{k+1} + (\widetilde{f}_{k+1} + c_{k+1}\widetilde{d}_{k+1})^2\right)$.

From $\widehat{c}_k\widehat{f}_k + d_k = (c_k\widetilde{f}_k + \widetilde{d}_k)^2 + (s_k\widetilde{f}_k)^2$ and the Cholesky decomposition we get

$$\widehat{c}_k\widehat{f}_k = c_k z_k + (s_k\widetilde{f}_k)^2.$$

Equations for V^TV

The product $B = V^TV$ is a s.p.d. DPSS matrix. The lower triangular elements of B are

$$\begin{aligned} b_{kk} &= (c_k\widetilde{f}_k + \widetilde{d}_k)^2 + (s_k\widetilde{f}_k)^2, \\ b_{jk} &= s_k s_{k+1}\cdots s_{j-1}\widetilde{f}_k(\widetilde{f}_j + c_j\widetilde{d}_j), \end{aligned}$$

where $k = 1, \ldots, n$ and $j > k$.

$B = \mathrm{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \mathrm{diag}(d)$. For $k = 1, \ldots, n-1$ we have

$$\widehat{s}_k^2\widehat{f}_k^2 = \sum_{j=k+1}^{n} b_{jk}^2 = \widetilde{f}_k^2 p_k,, \qquad \text{where} \qquad p_k = \sum_{j=k+1}^{n} (s_k s_{k+1}\cdots s_{j-1})^2(\widetilde{f}_j + c_j\widetilde{d}_j)^2.$$

For p_k we can apply the recursion $p_n = 0$ and $p_k = s_k^2\left(p_{k+1} + (\widetilde{f}_{k+1} + c_{k+1}\widetilde{d}_{k+1})^2\right)$.

From $\widehat{c}_k\widehat{f}_k + d_k = (c_k\widetilde{f}_k + \widetilde{d}_k)^2 + (s_k\widetilde{f}_k)^2$ and the Cholesky decomposition we get

$$\widehat{c}_k\widehat{f}_k = c_k z_k + (s_k\widetilde{f}_k)^2.$$

Now, $\widehat{c}_k$, $\widehat{s}_k$, and $\widehat{f}_k$ can be computed from

$$\begin{aligned} \widehat{c}_k\widehat{f}_k &= c_k z_k + (s_k\widetilde{f}_k)^2, \\ \widehat{s}_k\widehat{f}_k &= \widetilde{f}_k\sqrt{p_k}. \end{aligned}$$

Algorithm for the $V^T V$ product

Let $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ be s.p.d.

$$\Longrightarrow A = VV^T, \text{ where } V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d}),$$
$$\Longrightarrow B = V^T V = \mathrm{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \mathrm{diag}(d).$$

function $[\widehat{c}, \widehat{s}, \widehat{f}] = \mathrm{VTV}(c, s, \widetilde{f}, z)$
$c_n = 1$
$\widehat{f}_n = (\widetilde{f}_n + \widetilde{d}_n)^2 - d_n$
$p_n = 0$
for $k = n-1, \ldots, 2, 1$
$\quad p_k = s_k^2 \left(p_{k+1} + (\widetilde{f}_{k+1} + c_{k+1}\widetilde{d}_{k+1})^2 \right)$
$\quad [\widehat{c}_k, \widehat{s}_k, \widehat{f}_k] = \mathrm{Givens}(c_k z_k + s_k^2 \widetilde{f}_k^2, \widetilde{f}_k \sqrt{p_k})$

$[c, s, f] = \mathrm{Givens}(x, y)$ returns the Givens transformation such that $\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$.

Flops: $16n + \mathcal{O}(1) \Longrightarrow$ one step of the zero shift Cholesky LR algorithm: $27n + \mathcal{O}(1)$ flops.

As the eigenvalues are invariant to the sign of $\widehat{s}_k$, $k = 1, \ldots, n-1$, we do not care about it.

Laguerre's shift

Let A be a s.p.d. $n \times n$ matrix with eigenvalues $0 < \lambda_n \leq \lambda_{n-1} \leq \ldots \leq \lambda_1$. We apply Laguerre's method on the characteristic polynomial $f(\lambda) = \det(A - \lambda I)$. If x is an approximation for an eigenvalue of A and

$$S_1(x) = \sum_{i=1}^{n} \frac{1}{\lambda_i - x} = -\frac{f'(x)}{f(x)}$$

$$S_2(x) = \sum_{i=1}^{n} \frac{1}{(\lambda_i - x)^2} = \frac{f'^2(x) - f(x)f''(x)}{f^2(x)}$$

then the next approximation $\widetilde{x}$ by Laguerre's method is given by the equation

$$\widetilde{x} = x + \frac{n}{S_1(x) + \sqrt{(n-1)(nS_2(x) - S_1^2(x))}}.$$

Two important properties of Laguerre's method: if λ_n is a simple eigenvalue and if $x < \lambda_n$ then

- $x < \widetilde{x} < \lambda_n$,
- the convergence towards λ_n is cubic (linear for a multiple eigenvalue).

Laguerre's shift

Let A be a s.p.d. $n \times n$ matrix with eigenvalues $0 < \lambda_n \leq \lambda_{n-1} \leq \ldots \leq \lambda_1$. We apply Laguerre's method on the characteristic polynomial $f(\lambda) = \det(A - \lambda I)$. If x is an approximation for an eigenvalue of A and

$$S_1(x) = \sum_{i=1}^{n} \frac{1}{\lambda_i - x} = -\frac{f'(x)}{f(x)} = \mathrm{trace}((A - xI)^{-1}),$$

$$S_2(x) = \sum_{i=1}^{n} \frac{1}{(\lambda_i - x)^2} = \frac{f'^2(x) - f(x)f''(x)}{f^2(x)} = \mathrm{trace}((A - xI)^{-2}),$$

then the next approximation $\widetilde{x}$ by Laguerre's method is given by the equation

$$\widetilde{x} = x + \frac{n}{S_1(x) + \sqrt{(n-1)(nS_2(x) - S_1^2(x))}}.$$

Two important properties of Laguerre's method: if λ_n is a simple eigenvalue and if $x < \lambda_n$ then

- $x < \widetilde{x} < \lambda_n$,
- the convergence towards λ_n is cubic (linear for a multiple eigenvalue).

Computation of Laguerre's shift

We need $S_1(\sigma) = \mathrm{trace}((A - \sigma I)^{-1})$ and $S_2(\sigma) = \mathrm{trace}((A - \sigma I)^{-2})$.

If $A - \sigma I = VV^T$ is the Cholesky decomposition and $W = V^{-1}$, then

$$S_1(\sigma) = \mathrm{trace}(W^T W) = \|W\|_F^2,$$

$$S_2(\sigma) = \mathrm{trace}(W^T W W^T W) = \mathrm{trace}(W W^T W W^T) = \|W W^T\|_F^2.$$

Computation of Laguerre's shift

We need $S_1(\sigma) = \mathrm{trace}((A - \sigma I)^{-1})$ and $S_2(\sigma) = \mathrm{trace}((A - \sigma I)^{-2})$.

If $A - \sigma I = VV^T$ is the Cholesky decomposition and $W = V^{-1}$, then

$$S_1(\sigma) = \mathrm{trace}(W^T W) = \|W\|_F^2,$$

$$S_2(\sigma) = \mathrm{trace}(W^T W W^T W) = \mathrm{trace}(W W^T W W^T) = \|W W^T\|_F^2.$$

Delvaux, Van Barel (2004): Let $V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d})$ be nonsingular and $\widetilde{d}_i \neq 0$ for $i = 1, \ldots, n$. Then $W = V^{-1} = \mathrm{tril}(\mathrm{Giv}(\bar{c}, \bar{s}, \bar{f})) + \mathrm{diag}(\bar{d})$, where $\bar{d}_i = \widetilde{d}_i^{-1}$ for $i = 1, \ldots, n$.

Let us assume that $W = \mathrm{tril}(\mathrm{Giv}(\bar{c}, \bar{s}, \bar{f})) + \mathrm{diag}(\bar{d})$. The final algorithm will also be correct when W is not DPSS, which happens when $\widetilde{d}_i = 0$ for some $i = 2, \ldots, n-1$.

Computation of Laguerre's shift

$$A_4 = \begin{bmatrix} c_1 f_1 + d_1 & \times & \times & \times \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \times & \times \\ c_3 s_2 s_1 f_1 & c_3 s_2 f_2 & c_3 f_3 + d_3 & \times \\ s_3 s_2 s_1 f_1 & s_3 s_2 f_2 & s_3 f_3 & f_4 + d_4 \end{bmatrix}.$$

Lemma: If $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ is a symmetric $n \times n$ DPSS matrix then

$$\|A\|_F^2 = \sum_{k=1}^{n} (c_k f_k + d_k)^2 + 2 \sum_{k=1}^{n-1} s_k^2 f_k^2.$$

Lemma: If $W = \mathrm{tril}(\mathrm{Giv}(\bar{c}, \bar{s}, \bar{f})) + \mathrm{diag}(\bar{d})$ and $\bar{c}_k \neq 0$ for $k = 2, \ldots, n-1$, then

$$\begin{aligned} \|WW^T\|_F^2 &= \sum_{k=1}^{n} (WW^T)_{kk}^2 + 2 \sum_{k=1}^{n-1} \left(\frac{(WW^T)_{k+1,k}}{\bar{c}_{k+1}} \right)^2, \\ \|W\|_F^2 &= \sum_{k=1}^{n} (WW^T)_{kk}. \end{aligned}$$

Algorithm for the Laguerre's shift

The following algorithm computes $S_1 = \|W\|_F^2$ and $S_2 = \|WW^T\|_F^2$.

function $[S_1,\, S_2] = \text{invtrace}(c,\, s,\, \widetilde{f},\, \widetilde{d},\, y)$

$c_n = \bar{c}_n = 1$

for $k = n-1, \ldots, 2, 1$:

$\quad [\bar{c}_k,\, \bar{s}_k] = \text{Givens}(c_k \bar{c}_{k+1} y_{k+1},\, c_{k+1} s_k \widetilde{d}_k)$

$r_1 = 0$

for $k = 1, \ldots, n-1$

$$\beta_k = -c_{k+1} s_k \widetilde{f}_k / (\bar{c}_{k+1} y_k y_{k+1}) \qquad = w_{k+1,k}/\bar{c}_{k+1}$$
$$\omega_k = \bar{c}_k^{\,2}\, r_k + y_k^{-2} \qquad = (WW^T)_{kk}$$
$$\xi_k = \bar{c}_k \bar{s}_k r_k + \beta_k / y_k \qquad = (WW^T)_{k+1,k}/\bar{c}_{k+1}$$
$$r_{k+1} = \bar{s}_k^2 r_k + \beta_k^2$$

$\omega_n = r_n + y_n^{-2}$

$S_1 = \sum_{k=1}^{n} \omega_k$

$S_2 = \sum_{k=1}^{n} \omega_k^2 + 2 \sum_{k=1}^{n-1} \xi_k^2$

Flops: $31n + \mathcal{O}(1)$ flops.

The algorithm is also correct if $\widetilde{d}_k = 0$ for some $k = 2, \ldots, n-1$.

Implementation

$$A_4 = \begin{bmatrix} c_1 f_1 + d_1 & \times & \times & \times \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \times & \times \\ c_3 s_2 s_1 f_1 & c_3 s_2 f_2 & c_3 f_3 + d_3 & \times \\ s_3 s_2 s_1 f_1 & s_3 s_2 f_2 & s_3 f_3 & f_4 + d_4 \end{bmatrix}.$$

- If $|s_k|$ is small enough for some $k = 1, \ldots, n-1$, then we decouple the problem into two smaller problems with matrices $A(1:k, 1:k)$ and $A(k+1:n, k+1:n)$.

- If $|s_{n-1}|$ is small enough, we take $f_n + d_n$ as an eigenvalue of A and continue with vectors $c(1:n-2)$, $s(1:n-2)$, $f(1:n-1)$, and $d(1:n-1)$. New initial shift is $f_n + d_n$.

- Even if $\sigma_k < \lambda_n$, the Cholesky factorization can fail if the difference is too small. This is a problem since Laguerre's shifts converge faster to the smallest eigenvalue than $(A_k)_{nn}$. One strategy is to relax the shift with a factor τ close to 1, for instance, $\tau = 1 - 10^{-4}$.

- The computation of Laguerre's shift requires more than half of the operations for one step of the Cholesky LR. We can save work by fixing the shift once the shift improvement is small enough.

- The algorithm for Laguerre's shift fails if $c_k = 0$ for some $k = 2, \ldots, n-1$. A simple solution is to perturb c_k into some small δ (for instance, 10^{-20}) whenever $c_k = 0$.

Numerical examples

Numerical results were obtained with Matlab 7.0 running on a Pentium4 2.6 GHz Windows XP.

We compared

- a Matlab implementation of the Cholesky LR algorithm,
- a Matlab implementation of the implicit QR algorithm for DPSS matrices [Van Camp, Delvaux, Van Barel, Vandebril, Mastronardi (2005)],
- the Matlab function `eig`.

Exact eigenvalues were computed in Mathematica 5 using variable precision.

The cutoff criterion for both Cholesky LR and implicit QR is 10^{-16}.

With the maximum relative error we denote $\max_{1 \leq i \leq n} \frac{|\lambda_i - \widetilde{\lambda}_i|}{|\lambda_i|}$.

When the initial matrix is not DPSS, the reduction into a similar DPSS matrix is done by the algorithm of Vandebril, Van Camp, Van Barel, and Mastronardi (2004).

There is a connection between the Lanczos method and the reduction into a similar DPSS matrix which causes that the largest eigenvalues of A are approximated by the lower right diagonal elements of the DPSS matrix. As this is not good for the Cholesky LR method, we reverse the direction of the columns and rows of the DPSS matrix. This can be done in linear time, see, e.g., Vandebril (2004).

Numerical example 1

We take random s.p.d. DPSS matrices of the form

$$A = \mathrm{diag}(1, \ldots, n) + \mathrm{triu}(uv^T, 1) + \mathrm{triu}(uv^T, 1)^T + \alpha I,$$

where u and v are vectors of uniformly distributed random entries on $[0, 1]$, obtained by the Matlab function `rand`, and the shift α is such that the smallest eigenvalue of A is 1. The condition numbers of these matrices are approximately n.

	Cholesky LR			Implicit QR			`eig`	
n	t	steps	error	t	steps	error	t	error
50	0.06	274	$9.2 \cdot 10^{-15}$	0.19	83	$4.3 \cdot 10^{-15}$	0.00	$1.8 \cdot 10^{-14}$
100	0.16	557	$1.0 \cdot 10^{-14}$	0.69	164	$4.8 \cdot 10^{-14}$	0.00	$1.0 \cdot 10^{-13}$
150	0.28	832	$1.8 \cdot 10^{-14}$	1.45	242	$2.9 \cdot 10^{-13}$	0.00	$1.4 \cdot 10^{-13}$
200	0.49	1104	$2.6 \cdot 10^{-14}$	2.59	311	$3.6 \cdot 10^{-13}$	0.02	$1.2 \cdot 10^{-13}$
250	0.72	1390	$6.4 \cdot 10^{-14}$	4.22	414	$6.7 \cdot 10^{-13}$	0.03	$7.6 \cdot 10^{-13}$
300	0.97	1660	$1.3 \cdot 10^{-13}$	6.18	486	$4.7 \cdot 10^{-13}$	0.05	$1.2 \cdot 10^{-13}$
350	1.25	1933	$4.8 \cdot 10^{-14}$	8.86	564	$1.5 \cdot 10^{-12}$	0.09	$5.6 \cdot 10^{-13}$
400	1.59	2194	$1.3 \cdot 10^{-13}$	11.95	684	$4.3 \cdot 10^{-12}$	0.14	$7.8 \cdot 10^{-13}$
450	1.94	2479	$9.8 \cdot 10^{-14}$	15.78	730	$2.2 \cdot 10^{-12}$	0.22	$6.8 \cdot 10^{-13}$
500	2.34	2741	$1.0 \cdot 10^{-13}$	19.72	821	$4.5 \cdot 10^{-12}$	0.28	$3.8 \cdot 10^{-13}$

Numerical example 2

We take symmetric positive definite matrices

$$A = Q\,\mathrm{diag}(1:n)Q^T,$$

where Q is a random orthogonal matrix. Now we have to transform the matrix into a similar DPSS matrix before we can apply Cholesky LR or implicit QR.

	Cholesky LR			Implicit QR			`eig`	
n	t *	steps	error	t *	steps	error	t	error
500	3.1	3455	$2.2 \cdot 10^{-13}$	41.5	942	$2.9 \cdot 10^{-13}$	1.8	$2.4 \cdot 10^{-13}$
1000	12.7	6923	$2.8 \cdot 10^{-13}$	190.9	1804	$6.6 \cdot 10^{-13}$	13.4	$6.2 \cdot 10^{-13}$
1500	26.4	10384	$4.6 \cdot 10^{-13}$	496.1	2641	$4.6 \cdot 10^{-13}$	50.9	$4.5 \cdot 10^{-13}$
2000	46.1	13859	$7.6 \cdot 10^{-13}$	1067.9	3448	$2.8 \cdot 10^{-12}$	123.0	$3.0 \cdot 10^{-12}$
2500	72.8	17326	$9.5 \cdot 10^{-13}$	1962.3	4263	$4.7 \cdot 10^{-12}$	279.6	$4.1 \cdot 10^{-12}$

$*$: times for Cholesky LR and Implicit QR do not include the reduction to a DPSS matrix.

Numerical example 3

We take symmetric positive definite matrices

$$A = Q\,\mathrm{diag}(\lambda_1, \ldots, \lambda_n)Q^T,$$

where Q is a random orthogonal matrix and

$$\lambda_k = 10^{-8+7\frac{k-1}{n-1}}.$$

	Cholesky LR			Implicit QR			`eig`	
n	t *	steps	error	t *	steps	error	t	error
50	0.05	276	$4.3 \cdot 10^{-8}$	0.14	78	$3.1 \cdot 10^{-8}$	0.00	$1.1 \cdot 10^{-8}$
100	0.14	585	$5.4 \cdot 10^{-8}$	0.52	153	$3.8 \cdot 10^{-8}$	0.03	$8.0 \cdot 10^{-9}$
150	0.24	857	$8.3 \cdot 10^{-8}$	1.20	233	$7.3 \cdot 10^{-8}$	0.08	$7.2 \cdot 10^{-9}$
200	0.38	1143	$8.7 \cdot 10^{-8}$	2.14	307	$1.3 \cdot 10^{-7}$	0.17	$6.6 \cdot 10^{-9}$
250	0.52	1438	$7.4 \cdot 10^{-8}$	3.36	377	$8.4 \cdot 10^{-8}$	0.34	$1.1 \cdot 10^{-8}$
300	0.72	1712	$1.6 \cdot 10^{-7}$	5.23	466	$7.0 \cdot 10^{-8}$	0.66	$7.9 \cdot 10^{-9}$

$*$: times for Cholesky LR and Implicit QR do not include the reduction to a DPSS matrix.

Conclusions

- Cholesky LR algorithm exploits the structure of s.p.d. DPSS matrices.
- The method can be combined with Laguerre's shifts.
- It seems natural to compare the method to the implicit QR for DPSS matrices. In Cholesky LR the eigenvalues are computed from the smallest to the largest eigenvalue, therefore the method is very appropriate for applications where one is interested in few of the smallest eigenvalues.
- If the complete spectrum is computed, Cholesky LR is more expensive than implicit QR, but, as it tends to be slightly more accurate, it presents an alternative.
- The proposed method combined with the reduction to DPSS matrices can also be applied to a general s.p.d. matrix.