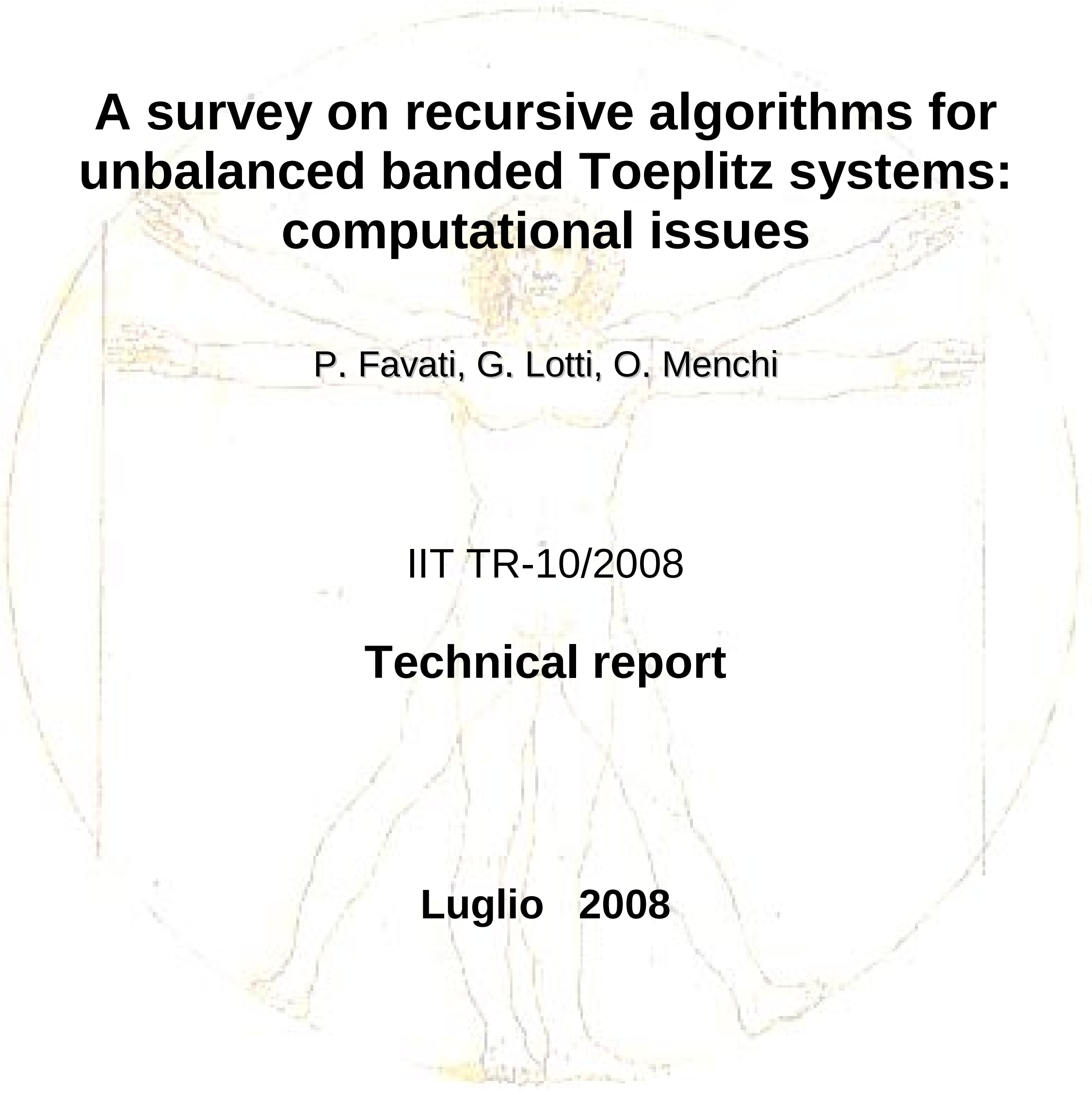


A survey on recursive algorithms for unbalanced banded Toeplitz systems: computational issues

P. Favati, G. Lotti, O. Menchi

IIT TR-10/2008

Technical report

Luglio 2008

A survey on recursive algorithms for unbalanced banded Toeplitz systems: computational issues

P. Favati G. Lotti O. Menchi

Abstract

Several direct recursive algorithms for the solution of band Toeplitz systems are considered. All the methods exploit the displacement rank properties, which allow a large reduction of computational efforts and storage requirements. Some algorithms make use of the Sherman-Morrison-Woodbury formula and result to be particularly suitable for the case of unbalanced bandwidths. The computational costs of the algorithms under consideration are compared both in a theoretical and practical setting. Some stability issues are discussed as well.

1 Introduction

A Toeplitz matrix is a matrix whose elements are constant along each descending diagonal from left to right. Toeplitz matrices appear in many applications where shift invariance properties are present. Classical examples are the systems arising from the finite difference discretization of differential equations. Toeplitz systems arise also in the modeling of queueing problems with Markov chains and in image deconvolution problems, where the blurring operator is described in terms of a point spread function space invariant with respect to translation, see [9] for a survey of some scientific applications.

By exploiting the persymmetry of Toeplitz matrices, these systems can be easily solved with a computational cost equal to the square of the size (see [13]). Faster methods employing recursive techniques have been devised (see for example [5]).

Often the systems with Toeplitz matrices are banded in a natural way or can be considered banded from a numerical point of view because of the decay of the coefficients when going away from the diagonal. Since large scale banded Toeplitz systems are frequently encountered in the applications, efficient algorithms are especially required. Techniques based on the cyclic reduction or on preconditioned conjugate gradient have been proposed [3], [8], [20].

In this note we consider the problem of solving a linear system whose matrix A has a band Toeplitz structure, with m upper diagonals and $n \geq m$ lower diagonals. Such a matrix can be seen as having a block tridiagonal structure, with blocks of size n. We assume that the size of A is multiple of n. Matrix A is said to be *unbalanced* if m is lower than n. The problem is described in Section 2.

A fundamental device when dealing with Toeplitz matrices is the displacement rank theory, which allows a substantial reduction in the computational cost and storage requirements. For this reason a short survey of the basic displacement properties is included in Section 3.

The following four recursive methods are taken into consideration. For all the methods, except the second one, a considerable saving of the cost is achieved by using the Sherman-Morrison-Woodbury (in the following SMW) formula for computing the inverses.

- `ge` a recursive factorization, which in the case of a block tridiagonal matrix like (3) coincides with the block Gaussian elimination,
- `cr` the algorithm of [3], based on the cyclic reduction, which can be seen as the block Gaussian elimination of a suitably permuted matrix,
- `mcr` a modified version of algorithm of `cr`, which implements the SMW formula as suggested in [17],
- `st` a version of the divide-and-conquer algorithm introduced by Stewart in [22] and applied [2] to Toeplitz matrices, modified in [11] in order to exploit the band presence.

The methods are described in Sections 4, 5, 6 and 7. For each method, a brief sketch of its implementation is presented and an approximation of the multiplicative cost of the version exploiting displacement properties is given.

An important aspect which should always be taken into account is that of stability. Citing from [13] "unstable Toeplitz techniques abound and caution must be exercised". Block Gaussian elimination is known to be stable for diagonally dominant matrices. A brief analysis of the stability of each method is given in Section 8. We examine also how diagonal dominance properties influence the stability of the SMW formula used for computing the inverses.

A comparative analysis of the costs is presented in Section 9. Finally in Section 10 numerical experiments are performed to test the effectiveness of the different methods and their performance with respect to costs and stability.

2 The problem

We consider here the linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

where A is a non singular banded Toeplitz matrix, with m upper diagonals, and

$n \geq m$ lower diagonals

$$A = \begin{bmatrix} a_0 & a_{-1} & \dots & a_{-m} & & & \\ a_1 & a_0 & a_{-1} & \dots & a_{-m} & & \\ \vdots & \ddots & \ddots & \ddots & & \ddots & \\ a_n & \dots & a_1 & a_0 & a_{-1} & \dots & a_{-m} \\ & \ddots & & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & & \ddots & \ddots & a_{-1} \\ & & & a_n & \dots & a_1 & a_0 \end{bmatrix}. \tag{2}$$

We consider the case of an unbalanced matrix A, where m is significantly lower than n. Without loss of generality we assume $n\,N$ to be the size of A, with $N = 2^p$, $p \geq 2$. Then matrix A can be partitioned as an $N \times N$ block tridiagonal Toeplitz matrix

$$A = \begin{bmatrix} B_0 & B_{-1} & & & \\ B_1 & B_0 & B_{-1} & & \\ & \ddots & \ddots & \ddots & \\ & & B_1 & B_0 & B_{-1} \\ & & & B_1 & B_0 \end{bmatrix}, \tag{3}$$

where B_0, B_1 and B_{-1} are Toeplitz blocks of order n

$$B_0 = \begin{bmatrix} a_0 & \dots & a_{-m} & & \\ \vdots & \ddots & & \ddots & \\ \vdots & & \ddots & & a_{-m} \\ \vdots & & & \ddots & \vdots \\ a_{n-1} & \dots & \dots & \dots & a_0 \end{bmatrix}, \tag{4}$$

$$B_1 = \begin{bmatrix} a_n & \dots & \dots & a_1 \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ & & & a_n \end{bmatrix}, \qquad B_{-1} = \begin{bmatrix} 0 & & & & \\ \vdots & \ddots & & & \\ a_{-m} & & 0 & & \\ \vdots & \ddots & & \ddots & \\ a_{-1} & \dots & a_{-m} & \dots & 0 \end{bmatrix}.$$

We assume that $a_n \neq 0$ and $\det B_0 \neq 0$. In the following we use the vectors

$\boldsymbol{e}_j$, the jth canonical vector of size n,

$\overline{\boldsymbol{e}}_j$, the jth canonical vector of size m,

$\boldsymbol{f}_0 = B_0\boldsymbol{e}_1$, the first column of B_0,

$\boldsymbol{f}_{-1} = B_{-1}\boldsymbol{e}_1$, the first column of B_{-1},

and the matrices

O_h, the null matrix of order h,

I_h, the identity matrix of order h,

$I_{h,k}$, the rectangular matrix formed by the first h rows of I_k if $h \le k$ or by the first k columns of I_h if $k \le h$,

J_h, the exchange matrix of order h,

$\overline{F}_h$, the $h \times m$ matrix formed by the first m columns of I_h,

$\underline{F}_h$, the $h \times m$ matrix formed by the last m columns of I_h,

$\overline{E}_h$, the $h \times n$ matrix formed by the first n columns of I_h,

$\underline{E}_h$, the $h \times n$ matrix formed by the last n columns of I_h.

For the sake of notation simplicity, these matrices will be used without the index h, i.e. without mentioning the number of rows, when it is possible to infer the right size from the context. So the persymmetry of A and B_0 is expressed by $A = JA^TJ$ and $B_0 = JB_0^TJ$ (in the first case $J = J_{nN}$ and in the second case $J = J_n$).

We consider also the $m \times m$ block R in the bottom left angle of B_{-1}, i.e.

$$R = \begin{bmatrix} a_{-m} & & O \\ \vdots & \ddots & \\ a_{-1} & \dots & a_{-m} \end{bmatrix}, \quad \text{such that} \quad B_{-1} = \underline{F} R \overline{F}^T.$$

The function log, used for expressing the computational cost of the algorithms, is the base 2 logarithm.

3 Exploiting the Toeplitz structure

A significant reduction of the computational cost can be achieved by making use of the concept of displacement rank, introduced in [16], [15], [4]. This concept allows a compact representation of the matrices involved in the algorithms. In this section, after the first definitions and properties, we examine how to perform the basic operations on matrices represented in such a way.

3.1 Basic properties

Let $h \ge 2$ be an integer and consider *down-shift* matrix of order h

$$Z_h = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}.$$

For an $h \times k$ matrix T the following *displacement operator* will be considered here

$$\Delta(T) = TZ_k - Z_h T \tag{5}$$

(the subscripts h and k will be dropped when equal). Matrix T is said to have *d-rank* r if $\Delta(T)$ has rank r. In this case $\Delta(T)$ can be expressed in the form

$$\Delta(T) = \sum_{i=1}^{r} \boldsymbol{x}_i \boldsymbol{y}_i^T, \tag{6}$$

for suitable h-vectors $\boldsymbol{x}_i$ and k-vectors $\boldsymbol{y}_i$, called the *d-vectors* of T. Matrix T is said to be *Toeplitz-like* if its d-rank r is low with respect to its size (more exactly if $r = O(1)$ for $h, k \to \infty$). All the matrices discussed in this paper are Toeplitz-like.

The d-vectors, together with the first column $T\mathbf{e}_1$, allow the complete reconstruction of T. In fact from (6) it follows that

$$T = L(T\mathbf{e}_1)\, I_{h,k} + \sum_{i=1}^{r} L(\boldsymbol{x}_i)\, I_{h,k}\, L^T(Z_k \boldsymbol{y}_i), \tag{7}$$

where $L(\boldsymbol{s})$ denotes the lower triangular Toeplitz matrix whose first column is $\boldsymbol{s}$. The set of the d-vectors plus the first column constitute the *d-description* of T, while formula (7) provides the *d-representation* of T.

The components of the vector $Z_k \boldsymbol{y}_i$ are shifted one place to the right. Since $Z_k \boldsymbol{e}_n = \boldsymbol{0}$, if one of the vector $\boldsymbol{y}_i$ coincides with $\boldsymbol{e}_n$ the corresponding ith term is missing from the sum of (7).

If a Toeplitz-like matrix T is (square and) persymmetric, then

$$\big[\Delta(T)\big]^T = Z^T T^T - T^T Z^T = -J(TZ - ZT)J = -J\Delta(T)J.$$

Hence, for any pair $\boldsymbol{x}_i \boldsymbol{y}_i^T J$ appearing in sum (6), the pair $-\boldsymbol{y}_i \boldsymbol{x}_i^T J$ must appear too. Then r must be even and (6) becomes

$$\Delta(T) = \sum_{i=1}^{r/2} \big(\boldsymbol{x}_i \boldsymbol{y}_i^T J - \boldsymbol{y}_i \boldsymbol{x}_i^T J\big).$$

It follows that under the hypothesis of persymmetry the d-representation (7) of T becomes

$$T = L(T\mathbf{e}_1) + \sum_{i=1}^{r/2} L(\boldsymbol{x}_i)\, L^T(ZJ\boldsymbol{y}_i) - \sum_{i=1}^{r/2} L(\boldsymbol{y}_i)\, L^T(ZJ\boldsymbol{x}_i). \tag{8}$$

From (5) we derive the following rules:

1. displacement of a sum $\quad \Delta(S + T) = \Delta(S) + \Delta(T)$,

2. displacement of a product $\quad \Delta(ST) = \Delta(S)\,T + S\,\Delta(T)$,
3. displacement of the inverse $\quad \Delta(T^{-1}) = -T^{-1}\Delta(T)T^{-1}$.

Hence the inverse matrix has the same d-rank as the direct matrix.

We give now the d-description of the matrices we are considering.
Matrix B_0 size $n \times n$ d-rank=2

$$\Delta(B_0) = \boldsymbol{e}_1\boldsymbol{e}_n^T B_{-1} - B_{-1}\boldsymbol{e}_1\boldsymbol{e}_n^T = \boldsymbol{e}_1\boldsymbol{f}_{-1}^T J - \boldsymbol{f}_{-1}\boldsymbol{e}_1^T J.$$

The first column $\boldsymbol{f}_0$ is not a d-vector.
Matrix B_1 size $n \times n$ d-rank=2

$$\Delta(B_1) = \boldsymbol{e}_1\boldsymbol{e}_n^T B_0 - B_0\boldsymbol{e}_1\boldsymbol{e}_n^T = \boldsymbol{e}_1\boldsymbol{f}_0^T J - \boldsymbol{f}_0\boldsymbol{e}_1^T J.$$

The first column is the d-vector $a_n\boldsymbol{e}_1$.
Matrix B_{-1} size $n \times n$ d-rank=0

$$\Delta(B_{-1}) = O.$$

The first column $\boldsymbol{f}_{-1}$ is not a d-vector.
Matrix B_0^{-1} size $n \times n$ d-rank=2

$$\begin{aligned}\Delta(B_0^{-1}) &= -B_0^{-1}\Delta(B_0)B_0^{-1} = -B_0^{-1}\big(\boldsymbol{e}_1\boldsymbol{f}_{-1}^T J - \boldsymbol{f}_{-1}\boldsymbol{e}_1^T J\big)B_0^{-1} \\ &= \widehat{\boldsymbol{f}}_1\,\widehat{\boldsymbol{f}}_2^T J - \widehat{\boldsymbol{f}}_2\,\widehat{\boldsymbol{f}}_1^T J,\end{aligned}$$

where

$$\widehat{\boldsymbol{f}}_1 = B_0^{-1}B_{-1}\boldsymbol{e}_1 = B_0^{-1}\boldsymbol{f}_{-1}, \qquad \widehat{\boldsymbol{f}}_2 = B_0^{-1}\boldsymbol{e}_1.$$

The first column of B_0^{-1} is the d-vector $\widehat{\boldsymbol{f}}_2$.
Matrices $\overline{E} = \overline{E}_h$ and $\underline{E} = \underline{E}_h$ (with $h > n$)

$$\Delta(\overline{E}) = -\widetilde{\boldsymbol{e}}_{n+1}\boldsymbol{e}_n^T, \qquad \Delta(\overline{E}^{\,T}) = 0, \qquad \Delta(\underline{E}) = 0, \qquad \Delta(\underline{E}^T) = \boldsymbol{e}_1\widetilde{\boldsymbol{e}}_{h-n}^T,$$

where $\widetilde{\boldsymbol{e}}_j$ is the jth canonical vector of length h.
Matrices $\overline{F} = \overline{F}_n$ and $\underline{F} = \underline{F}_n$

$$\Delta(\overline{F}) = -\boldsymbol{e}_{m+1}\boldsymbol{e}_m^T, \qquad \Delta(\overline{F}^T) = O, \qquad \Delta(\underline{F}) = O, \qquad \Delta(\underline{F}^T) = \boldsymbol{e}_1\boldsymbol{e}_{n-m}^T.$$

3.2 Product of a Toeplitz-like matrix by vector

The most important operation that we have to perform is the multiplication of T by a vector $\boldsymbol{v}$. To see how it is made in a fast way and to estimate the cost, we start with the case of the lower triangular Toeplitz matrix. Let $\boldsymbol{s}$ and $\boldsymbol{v}$ be two input vectors of order n. To compute the product $L(\boldsymbol{s})\boldsymbol{v}$ or the product

$L^T(ZJ\boldsymbol{s})\boldsymbol{v}$, the two vectors $\boldsymbol{v}$ and $\boldsymbol{s}$ are embedded into vectors of double size $\boldsymbol{v}'$ and $\boldsymbol{s}'$. Then we consider the circulant matrix $C(\boldsymbol{s}')$ whose first column is $\boldsymbol{s}'$

$$\boldsymbol{v}' = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{s}' = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{0} \end{bmatrix}, \qquad C(\boldsymbol{s}') = \begin{bmatrix} L(\boldsymbol{s}) & L^T(ZJ\boldsymbol{s}) \\ L^T(ZJ\boldsymbol{s}) & L(\boldsymbol{s}) \end{bmatrix}.$$

Computing

$$\boldsymbol{z} = C(\boldsymbol{s}')\boldsymbol{v}' = \begin{bmatrix} L(\boldsymbol{s})\boldsymbol{v} \\ L^T(ZJ\boldsymbol{s})\boldsymbol{v} \end{bmatrix},$$

we find the product $L(\boldsymbol{s})\boldsymbol{v}$ in the first half of $\boldsymbol{z}$ and the product $L^T(ZJ\boldsymbol{s})\boldsymbol{v}$ in the second half of $\boldsymbol{z}$.

Circulant matrices of order $2n$ are diagonalized by the Fourier matrix F_{2n} whose (j,k)th element is $f_{jk} = \exp(\pi \boldsymbol{i} jk/n)$, for $j,k = 0,\dots,2n-1$. Hence $\boldsymbol{z}$ can be obtained by computing successively the vectors

$$\boldsymbol{w} = \mathcal{F}(\boldsymbol{v}'), \qquad \boldsymbol{g} = \mathcal{F}(\boldsymbol{s}'), \qquad \boldsymbol{h} = \boldsymbol{g} \odot \boldsymbol{w}, \qquad \boldsymbol{z} = \mathcal{F}^{-1}(\boldsymbol{h}), \tag{9}$$

where $\odot$ indicates the element-wise product of two vectors of the same size and $\mathcal{F}$ and $\mathcal{F}^{-1}$ indicate the discrete Fourier transform and its inverse, defined as

$$\mathcal{F}(\boldsymbol{x}) = F_{2n}\,\boldsymbol{x}, \qquad \mathcal{F}^{-1}(\boldsymbol{x}) = \frac{1}{2n}\,F_{2n}^*\,\boldsymbol{x}.$$

Since the cost for computing the Fourier transform of a real vector of order n using a specialized FFT routine is $3/4n\log n + O(n)$ real multiplications [21], in what follows we indicate by $\pi_n = 3/2n\log n$ the cost of each Fourier transform (implicitly assuming n large enough to neglect the term $O(n)$). We estimate the cost of computing the product $L(\boldsymbol{s})\boldsymbol{v}$ or $L^T(ZJ\boldsymbol{s})\boldsymbol{v}$ by the number of Fourier transforms required (in some cases one or both input vectors may have already been encountered and transformed).

For a general Toeplitz-like matrix T expressed as in (7), the product is found by combining the results obtained for the different triangular matrices. Due to the linearity of the Fourier transform, the transformation of the initial vector $\boldsymbol{v}$ and of the final output vector is counted only once. To help estimating the cost of the product for the matrices involved in the different methods, we will give for each matrix its d-description and a corresponding sketch of the lower and upper triangles. The sections of identity matrix dealing with the changes of size required for nonsquare matrices are represented by rectangles.

Sometimes it happens that different matrices are multiplied by the same vector or that the same matrix is multiplied by different vectors. To avoid counting double, we count separately the cost of the transformation of the input vector and assume that when a vector involved in the d-description of a matrix is computed, it is immediately transformed and the cost of this transformation is counted separately. Practically, this means that to each pair of a lower triangle of size n times an upper triangle of size m the cost $\pi_n + \pi_m$ is counted, while no cost is counted for a single triangle. One cost is assigned to the final vector. Different tables give the costs of the multiplication for the matrices of each algorithm. If not explicitly indicated, the cost for multiplying the transposed matrices is the same.

3.3 Inverse of a Toeplitz-like matrix

Many algorithms have been devised for computing the inverse of a Toeplitz-like matrix T. They are called "fast" and "superfast", according to the technique employed. Fast algorithms, like the one by Trench [23], exploit the persimmetry of Toeplitz matrices and have a computational cost $O(n^2)$, where n is the order of T. Superfast algorithms, which employ recursive techniques, have a lower computational cost $O(n \log^2 n)$ but are less stable. For this reason we employ here a fast algorithm modified from Levinson's for Toeplitz-like matrices [19]. It solves k systems with the same matrix T having d-rank r with multiplicative cost $\sigma_{n,r'}(k) + O(n)$, where

$$\sigma_{n,r'}(k) = (2k + 5r' + 1)/2\, n^2, \tag{10}$$

with $r' = r$ if the first column of T is one of its d-vectors and $r' = r+1$ otherwise (see [12], Table 1).

As we will see, all the methods require the knowledge of B_0^{-1}. To compute this inverse one can apply also the algorithm described in [2], which exploits the presence of an upper band and has a cost $O(m^2 \log(n/m) + n \log n)$. This computation is made in an initialization phase, preceding a transformation phase and a substitution phase which constitute the core of the algorithm. For this reason the cost of computing B_0^{-1} will be not considered in the theoretical estimates of the costs given in terms of π_n, π_m and $\sigma_{n,r'}(k)$.

4 Algorithm `ge`

It is well-known that when A is a tridiagonal matrix, the recursive method based on the partitioned LU factorization coincides with the Gaussian elimination. Hence the first method we consider for solving (1), which will be called `ge`, results to be just the block Gaussian elimination. It is based on the sequence of the Schur complements

$$S_0 = B_0, \qquad S_i = B_0 - B_1 S_{i-1}^{-1} B_{-1}, \quad \text{for} \quad i = 1, \dots, N-1, \tag{11}$$

and consists of two phases.

1. Transformation phase, where matrices S_i^{-1} are constructed.

2. Substitution phase: partitioning the vectors $\boldsymbol{b}$ and $\boldsymbol{x}$ into subvectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_N$ and $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ of size n, the solution of system (1) is found

by performing a *forward* substitution, where the vectors $\boldsymbol{z}_i$ are computed from the right hand-side

$$\begin{aligned} &\boldsymbol{z}_1 = B_0^{-1}\boldsymbol{b}_1, \\ &\boldsymbol{z}_i = S_{i-1}^{-1}(\boldsymbol{b}_i - B_1\boldsymbol{z}_{i-1}), \quad \text{for} \quad i = 2,\dots,N, \end{aligned} \tag{12}$$

followed by a *backward* substitution

$$\begin{aligned} &\boldsymbol{x}_N = \boldsymbol{z}_N, \\ &\boldsymbol{x}_i = \boldsymbol{z}_i - S_{i-1}^{-1}B_{-1}\boldsymbol{x}_{i+1}, \quad \text{for} \quad i = N-1,\dots,1. \end{aligned} \tag{13}$$

It is evident that the main part of the computational cost of the method lies in the transformation phase. This cost can be reduced if we exploit the fact that according to (11) each matrix S_i differs from B_0 by a correction of low rank (at most m). Since $B_{-1} = \underline{F}R\overline{F}^T$, by SMW formula we have

$$S_{i+1}^{-1} = (B_0 - B_1S_i^{-1}\underline{F}R\overline{F}^T)^{-1} = B_0^{-1} + B_0^{-1}U_iP_i^{-1}V, \tag{14}$$

where

$$V = R\overline{F}^T B_0^{-1}, \qquad U_i = B_1S_i^{-1}\underline{F} \qquad P_i = I - VU_i. \tag{15}$$

Once B_0^{-1} has been computed, the computation of the inverse of the $n \times n$ matrices S_i can be made through the inversion of the smaller $m \times m$ matrices P_i.

4.1 Sketchy implementation

The algorithm can be implemented in the following way. All the computations are carried out by using the d-descriptions of the involved matrices.

Function `ge` $(A, N, \boldsymbol{b})$

```
1. compute $B_0^{-1}$ and $V$,
2. for $i = 0,\dots,N-1$
3.     compute $U_i$ and $P_i$ by applying (15),
4.     compute $P_i^{-1}$,
5.     compute $S_{i+1}^{-1}$ by applying (14),
6. for $i = 1,\dots,N$ compute the vectors $\boldsymbol{z}_i$ by applying (12),
7. for $i = N,\dots,1$ compute the vectors $\boldsymbol{x}_i$ by applying (13).
```

4.2 D-description of the matrices

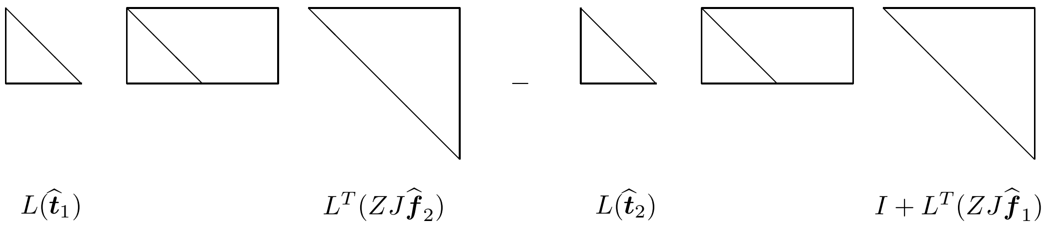
Matrix $V = R\overline{F}^T B_0^{-1}$ size $m \times n$ d-rank=2. Since $\Delta(R\overline{F}^T) = O$,

$$\Delta(V) = R\overline{F}^T\Delta(B_0^{-1}) = V\big(\boldsymbol{f}_{-1}\widehat{\boldsymbol{f}}_2^T J - \boldsymbol{e}_1\widehat{\boldsymbol{f}}_1^T J\big) = \widehat{\boldsymbol{t}}_1\widehat{\boldsymbol{f}}_2^T J - \widehat{\boldsymbol{t}}_2\widehat{\boldsymbol{f}}_1^T J,$$

where

$$\widehat{\boldsymbol{t}}_1 = V\boldsymbol{f}_{-1}, \qquad \widehat{\boldsymbol{t}}_2 = V\boldsymbol{e}_1.$$

The first column of V is the d-vector $\widehat{\boldsymbol{t}}_2$. The d-representation of V has the form



$L(\widehat{\boldsymbol{t}}_1)$
$L^T(ZJ\widehat{\boldsymbol{f}}_2)$
$-$
$L(\widehat{\boldsymbol{t}}_2)$
$I + L^T(ZJ\widehat{\boldsymbol{f}}_1)$

Matrix $S_i = B_0 - B_1 S_{i-1}^{-1} B_{-1}$ size $n \times n$ d-rank=3

$$\Delta(S_i) = \boldsymbol{s}_1^{(i)} \boldsymbol{t}_1^{(i)T} + \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} - \boldsymbol{f}_{-1}\boldsymbol{e}_n^T,$$

where

$$\boldsymbol{s}_1^{(0)} = \boldsymbol{f}_0, \qquad \boldsymbol{t}_1^{(0)T} = \boldsymbol{0}^T, \qquad \boldsymbol{s}_2^{(0)} = \boldsymbol{e}_1, \qquad \boldsymbol{t}_2^{(0)T} = \boldsymbol{f}_{-1}^T J,$$

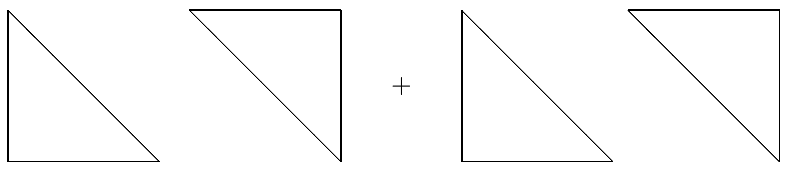
and for $i \geq 1$

$$\begin{aligned} \boldsymbol{s}_1^{(i)} &= S_i \boldsymbol{e}_1, \qquad & \boldsymbol{t}_1^{(i)T} &= \boldsymbol{e}_n^T S_{i-1}^{-1} B_{-1}, \\ \boldsymbol{s}_2^{(i)} &= B_1 S_{i-1}^{-1} \boldsymbol{s}_2^{(i-1)}, \qquad & \boldsymbol{t}_2^{(i)T} &= \boldsymbol{t}_2^{(i-1)T} S_{i-1}^{-1} B_{-1}. \end{aligned}$$

Proof by induction. For $i = 0$ we have $S_0 = B_0$ and the relation holds. For $i \geq 1$, since $B_1 \boldsymbol{e}_1 \boldsymbol{e}_n^T = \boldsymbol{e}_1 \boldsymbol{e}_n^T B_1$, we have

$$\begin{aligned} &\Delta(S_{i+1}) = \Delta(B_0) - \Delta(B_1) S_i^{-1} B_{-1} - B_1 \Delta(S_i^{-1}) B_{-1} \\ &= \boldsymbol{e}_1 \boldsymbol{e}_n^T B_{-1} - B_{-1}\boldsymbol{e}_1\boldsymbol{e}_n^T - \boldsymbol{e}_1 \boldsymbol{e}_n^T B_0 S_i^{-1} B_{-1} + B_0 \boldsymbol{e}_1 \boldsymbol{e}_n^T S_i^{-1} B_{-1} \\ &\quad + B_1 \boldsymbol{e}_1 \boldsymbol{e}_n^T S_{i-1}^{-1} B_{-1} S_i^{-1} B_{-1} + B_1 S_i^{-1} \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} S_i^{-1} B_{-1} - B_1 S_i^{-1} B_{-1} \boldsymbol{e}_1 \boldsymbol{e}_n^T S_i^{-1} B_{-1} \\ &= \boldsymbol{e}_1 \boldsymbol{e}_n^T B_{-1} - \boldsymbol{e}_1 \boldsymbol{e}_n^T \bigl(B_0 - B_1 S_{i-1}^{-1} B_{-1}\bigr) S_i^{-1} B_{-1} \\ &\quad + \bigl(B_0 - B_1 S_i^{-1} B_{-1}\bigr) \boldsymbol{e}_1 \boldsymbol{e}_n^T S_i^{-1} B_{-1} + B_1 S_i^{-1} \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} S_i^{-1} B_{-1} - \boldsymbol{f}_{-1} \boldsymbol{e}_n^T \\ &= S_{i+1} \boldsymbol{e}_1 \boldsymbol{e}_n^T S_i^{-1} B_{-1} + B_1 S_i^{-1} \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} S_i^{-1} B_{-1} - \boldsymbol{f}_{-1} \boldsymbol{e}_n^T. \end{aligned}$$

The first column of S_i is the d-vector $\boldsymbol{s}_i^{(1)}$. The d-representation of S_i has the form


$L(\boldsymbol{s}_1^{(i)})$
$I + L^T(Z\boldsymbol{t}_1^{(i)})$
$+$
$L(\boldsymbol{s}_2^{(i)})$
$L^T(Z\boldsymbol{t}_2^{(i)})$

Matrix $U_i = B_1 S_i^{-1} \underline{F}$ size $n \times m$ d-rank=3

$$\Delta(U_i) = \Delta(B_1 S_i^{-1} \underline{F}) = \Delta(B_1) S_i^{-1} \underline{F} + B_1 \Delta(S_i^{-1}) \underline{F}$$

$$= \big(\boldsymbol{e}_1 \boldsymbol{e}_n^T B_0 - B_0 \boldsymbol{e}_1 \boldsymbol{e}_n^T\big) S_i^{-1} \underline{F} - B_1 S_i^{-1} \big(\boldsymbol{s}_1^{(i)} \boldsymbol{t}_1^{(i)T} + \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} - B_{-1} \boldsymbol{e}_1 \boldsymbol{e}_n^T\big) S_i^{-1} \underline{F}$$

$$= -(B_0 - B_1 S_i^{-1} B_{-1}) \boldsymbol{e}_1 \boldsymbol{e}_n^T S_i^{-1} \underline{F} - B_1 S_i^{-1} \boldsymbol{s}_2^{(i)} \boldsymbol{t}_2^{(i)T} S_i^{-1} \underline{F}$$

$$+ \boldsymbol{e}_1 \boldsymbol{e}_n^T (B_0 - B_1 S_{i-1}^{-1} B_{-1}) S_i^{-1} \underline{F}$$

$$= \boldsymbol{u}_1^{(i)} \boldsymbol{v}_1^{(i)T} + \boldsymbol{u}_2^{(i)} \boldsymbol{v}_2^{(i)T} + \boldsymbol{e}_1 \boldsymbol{e}_n^T \underline{F},$$

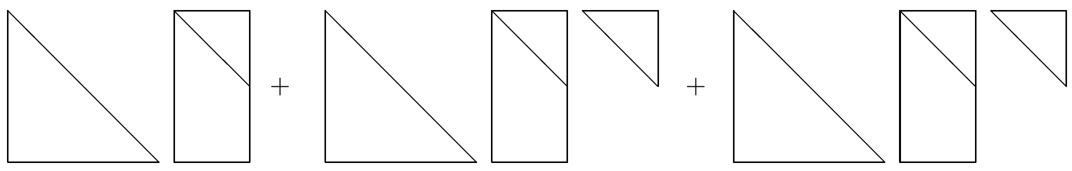
where

$$\boldsymbol{u}_1^{(i)} = -B_0 \boldsymbol{e}_1 + B_1 S_i^{-1} B_{-1} \boldsymbol{e}_1 = -\boldsymbol{s}_1^{(i+1)}, \qquad \boldsymbol{v}_1^{(i)T} = \boldsymbol{e}_n^T S_i^{-1} \underline{F} = \widetilde{\boldsymbol{t}}_3^{(i)T} \underline{F},$$

$$\boldsymbol{u}_2^{(i)} = -B_1 S_i^{-1} \boldsymbol{s}_2^{(i)} = -\boldsymbol{s}_2^{(i+1)}, \qquad \boldsymbol{v}_2^{(i)T} = \boldsymbol{t}_2^{(i)T} S_i^{-1} \underline{F} = \widetilde{\boldsymbol{t}}_2^{(i)T} \underline{F},$$

$\widetilde{\boldsymbol{t}}_2^{(i)}$ and $\widetilde{\boldsymbol{t}}_3^{(i)}$ are d-vectors of S_i^{-1}.

The first column of U_i is $\boldsymbol{u}^{(i)} = U_i \boldsymbol{e}_1 = B_1 S_i^{-1} \underline{F} \boldsymbol{e}_1 = B_1 S_i^{-1} \boldsymbol{e}_{n-m+1}$ and is not a d-vector. The d-representation of U_i has the form


$L(\boldsymbol{u}^{(i)})$
$+$
$L(\boldsymbol{u}_1^{(i)})$
$L^T(Z\boldsymbol{v}_1^{(i)})$
$+$
$L(\boldsymbol{u}_2^{(i)})$
$L^T(Z\boldsymbol{v}_2^{(i)})$

Matrix $P_i = I - VU_i$ size $m \times m$ d-rank=4 (since $\underline{F} - B_{-1} B_0^{-1} U_i = P_i$)

$$\Delta(P_i) = -V \Delta(U_i) - \Delta(V) U_i$$

$$= -\big(V\boldsymbol{u}_1^{(i)}\big) \boldsymbol{v}_1^{(i)T} - \big(V\boldsymbol{u}_2^{(i)}\big) \boldsymbol{v}_2^{(i)T} - \big(V\boldsymbol{e}_1\big) \boldsymbol{e}_n^T \underline{F}$$

$$-\big(VB_{-1}\boldsymbol{e}_1\big)\ \big(\boldsymbol{e}_n^T B_0^{-1} U_i\big) + \big(V\boldsymbol{e}_1\big)\ \big(\boldsymbol{e}_n^T B_{-1} B_0^{-1} U_i\big)$$

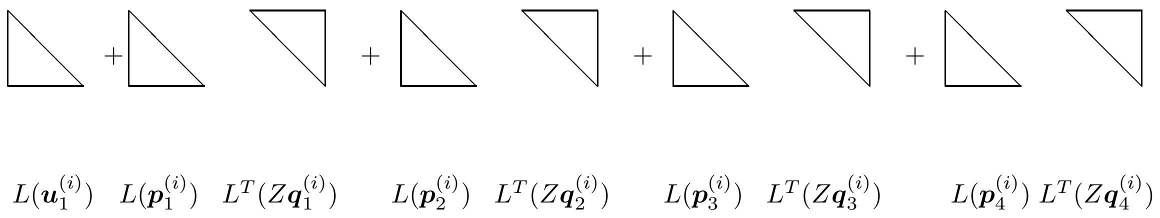
$$= -\big(V\boldsymbol{u}_1^{(i)}\big) \boldsymbol{v}_1^{(i)T} - \big(V\boldsymbol{u}_2^{(i)}\big) \boldsymbol{v}_2^{(i)T} - \big(V\boldsymbol{e}_1\big)\big(\boldsymbol{e}_n^T \underline{F} P_i\big) - \big(VB_{-1}\boldsymbol{e}_1\big)\ \big(\boldsymbol{e}_n^T B_0^{-1} U_i\big)$$

$$= \boldsymbol{p}_1^{(i)} \boldsymbol{q}_1^{(i)T} + \boldsymbol{p}_2^{(i)} \boldsymbol{q}_2^{(i)T} + \boldsymbol{p}_3^{(i)} \boldsymbol{q}_3^{(i)T} + \boldsymbol{p}_4^{(i)} \boldsymbol{q}_4^{(i)T},$$

where

$$\begin{aligned}
\boldsymbol{p}_1^{(i)} &= -V\boldsymbol{u}_1^{(i)}, & \boldsymbol{q}_1^{(i)T} &= \boldsymbol{v}_1^{(i)T},\\
\boldsymbol{p}_2^{(i)} &= -V\boldsymbol{u}_2^{(i)}, & \boldsymbol{q}_2^{(i)T} &= \boldsymbol{v}_2^{(i)T},\\
\boldsymbol{p}_3^{(i)} &= -V\boldsymbol{e}_1 = -\widehat{\boldsymbol{t}}_2, & \boldsymbol{q}_3^{(i)T} &= \boldsymbol{e}_n^T\underline{F}P_i = \boldsymbol{e}_n^T\underline{F} - \widehat{\boldsymbol{f}}_1^T JU_i,\\
\boldsymbol{p}_4^{(i)} &= -VB_{-1}\boldsymbol{e}_1 = -\widehat{\boldsymbol{t}}_1, & \boldsymbol{q}_4^{(i)T} &= \boldsymbol{e}_n^T B_0^{-1}U_i = \widehat{\boldsymbol{f}}_2^T JU_i.
\end{aligned}$$

The first column of P_i is $\boldsymbol{p}^{(i)} = P_i\underline{F}\boldsymbol{e}_1 = \underline{F}\boldsymbol{e}_1 - VU_i\underline{F}\boldsymbol{e}_1 = \underline{F}\boldsymbol{e}_1 - V\boldsymbol{u}^{(i)}$ and is not a d-vector. The d-representation of P_i has the form


+
+
+
+
$L(\boldsymbol{u}_1^{(i)})$
$L(\boldsymbol{p}_1^{(i)})$
$L^T(Z\boldsymbol{q}_1^{(i)})$
$L(\boldsymbol{p}_2^{(i)})$
$L^T(Z\boldsymbol{q}_2^{(i)})$
$L(\boldsymbol{p}_3^{(i)})$
$L^T(Z\boldsymbol{q}_3^{(i)})$
$L(\boldsymbol{p}_4^{(i)})$
$L^T(Z\boldsymbol{q}_4^{(i)})$

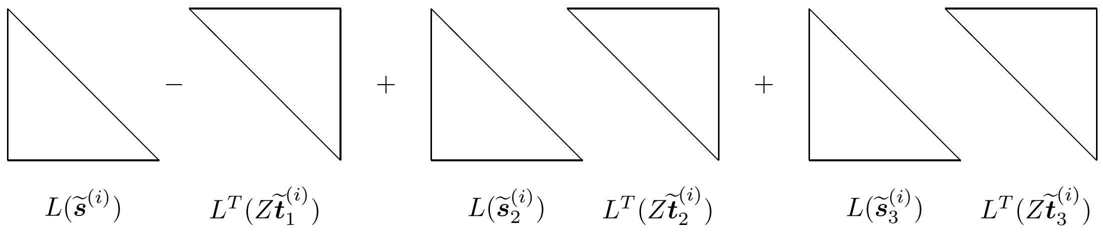
Matrix S_i^{-1} size $n \times n$ d-rank=3

$$\begin{aligned}
\Delta(S_i^{-1}) &= -S_i^{-1}\Delta(S_i)S_i^{-1}\\
&= \widetilde{\boldsymbol{s}}_1^{(i)}\widetilde{\boldsymbol{t}}_1^{(i)^T} + \widetilde{\boldsymbol{s}}_2^{(i)}\widetilde{\boldsymbol{t}}_2^{(i)^T} + \widetilde{\boldsymbol{s}}_3^{(i)}\widetilde{\boldsymbol{t}}_3^{(i)^T},
\end{aligned}$$

where

$$\begin{aligned}
\widetilde{\boldsymbol{s}}_1^{(i)} &= -S_i^{-1}\boldsymbol{s}_1^{(i)} = -\boldsymbol{e}_1, & \widetilde{\boldsymbol{t}}_1^{(i)^T} &= \boldsymbol{t}_1^{(i)T}S_i^{-1},\\
\widetilde{\boldsymbol{s}}_2^{(i)} &= -S_i^{-1}\boldsymbol{s}_2^{(i)}, & \widetilde{\boldsymbol{t}}_2^{(i)^T} &= \boldsymbol{t}_2^{(i)T}S_i^{-1},\\
\widetilde{\boldsymbol{s}}_3^{(i)} &= S_i^{-1}\boldsymbol{f}_{-1}, & \widetilde{\boldsymbol{t}}_3^{(i)^T} &= \boldsymbol{e}_n^T S_i^{-1}.
\end{aligned}$$

The first column of S_i^{-1} is $\widetilde{\boldsymbol{s}}^{(i)} = S_i^{-1}\boldsymbol{e}_1 = B_0^{-1}\big(\boldsymbol{e}_1 + U_{i-1}P_{i-1}^{-1}\widehat{\boldsymbol{t}}_2\big)$ and is not a d-vector. It must be computed by solving a system with matrix P_{i-1}. The d-representation of S_i^{-1} has the form


−
+
+
$L(\widetilde{\boldsymbol{s}}^{(i)})$
$L^T(Z\widetilde{\boldsymbol{t}}_1^{(i)})$
$L(\widetilde{\boldsymbol{s}}_2^{(i)})$
$L^T(Z\widetilde{\boldsymbol{t}}_2^{(i)})$
$L(\widetilde{\boldsymbol{s}}_3^{(i)})$
$L^T(Z\widetilde{\boldsymbol{t}}_3^{(i)})$

4.3 Cost of matrix by vector product

Table 1 lists the cost of computing matrix by vector products, for the matrices used in algorithm ge. As already said, this cost does not include the cost for the transformation of the vectors of the d-description of the matrix itself and the cost of transforming the input vector.

matrix	cost
B_1	π_n
B_{-1}	$2\pi_m$
B_0^{-1}	$5\pi_n$
V	$2\pi_n + 3\pi_m$
U_i	$4\pi_n + 2\pi_m$
U_i^T	$3\pi_n + 3\pi_m$
S_i^{-1}	$5\pi_m$

Table 1: Cost of computing matrix by vector products for algorithm ge.

Note. The product of B_{-1} by an n-vector $\boldsymbol{v}$ is computed multiplying matrix R by the subvector $\overline{\boldsymbol{v}}$ of the first m components of $\boldsymbol{v}$ and appending $n-m$ zeros in front of the result. Hence it requires transforming $\overline{\boldsymbol{v}}$ even if $\boldsymbol{v}$ has already been transformed. This explains why the cost for B_{-1} is $2\pi_m$ instead of π_m. Moreover, to multiply U_i by a m-vector $\boldsymbol{v}$ we need two transformed input vectors: the first one is obtained by transforming $\boldsymbol{v}$, the second one is obtained by transforming the vector $\boldsymbol{v}$ followed by $n-m$ zeros. This explains why the cost for U_i is $4\pi_n + 2\pi_m$ instead of $3\pi_n + 2\pi_m$.

4.4 Cost of algorithm ge

4.4.1 Transformation phase

(a) Initially the d-descriptions of matrices B_0^{-1}, V and U_0 are computed by solving three systems with matrix B_0 and computing the products

$$\widehat{\boldsymbol{f}}_1 = B_0^{-1}\boldsymbol{f}_{-1}, \qquad \widehat{\boldsymbol{f}}_2 = B_0^{-1}\boldsymbol{e}_1, \qquad \widehat{\boldsymbol{f}}_3 = B_0^{-1}\boldsymbol{e}_{n-m+1}, \qquad \widehat{\boldsymbol{t}}_1 = \underline{F}^T B_{-1}\widehat{\boldsymbol{f}}_1,$$
$$\widehat{\boldsymbol{t}}_2 = \underline{F}^T B_{-1}\widehat{\boldsymbol{f}}_2, \qquad \boldsymbol{u}_1^{(0)} = -\boldsymbol{f}_0 + B_1\widehat{\boldsymbol{f}}_1, \qquad \boldsymbol{u}_2^{(0)} = -B_1\widehat{\boldsymbol{f}}_2, \qquad \boldsymbol{u}^{(0)} = B_1\widehat{\boldsymbol{f}}_3,$$
$$\boldsymbol{v}_1^{(0)T} = \widehat{\boldsymbol{f}}_2^T J\underline{F}, \qquad \boldsymbol{v}_2^{(0)T} = \widehat{\boldsymbol{f}}_1^T J\underline{F}.$$

All these vectors are then transformed. Since $S_0^{-1} = B_0^{-1}$, we have also the d-description of S_0^{-1}

$$\widetilde{\boldsymbol{s}}_1^{(0)} = -\boldsymbol{e}_1, \qquad \widetilde{\boldsymbol{s}}_2^{(0)} = -\widehat{\boldsymbol{f}}_2, \qquad \widetilde{\boldsymbol{s}}_3^{(0)} = \widehat{\boldsymbol{f}}_1, \qquad \widetilde{\boldsymbol{s}}^{(0)} = \widehat{\boldsymbol{f}}_2,$$
$$\widetilde{\boldsymbol{t}}_1^{(0)} = \boldsymbol{0}, \qquad \widetilde{\boldsymbol{t}}_2^{(0)} = \widehat{\boldsymbol{f}}_1 J, \qquad \widetilde{\boldsymbol{t}}_3^{(0)} = \widehat{\boldsymbol{f}}_2 J.$$

As already said in Section 3.3, the cost of this initialization is not taken into consideration.

(b) Now the ith step begins, for $i = 0, \dots, p-1$. We assume that the d-vectors of S_i, S_i^{-1} and U_i are known and we compute the d-vectors of S_{i+1}, P_i, S_{i+1}^{-1} and U_{i+1}.

For matrix S_{i+1} we have

$\boldsymbol{s}_1^{(i+1)} = -\boldsymbol{u}_1^{(i)}$ and $\boldsymbol{s}_2^{(i+1)} = -\boldsymbol{u}_2^{(i)}$. It remains to compute

$\boldsymbol{t}_1^{(i+1)T} = \boldsymbol{e}_n^T S_i^{-1} B_{-1} = \widetilde{\boldsymbol{t}}_3^{(i)} B_{-1}$, the product by Toeplitz matrix R^T is performed with cost $2\pi_m$. The transformation of $\boldsymbol{t}_1^{(i+1)}$ costs π_n.

$\boldsymbol{t}_2^{(i+1)T} = \boldsymbol{t}_2^{(i)^T} S_i^{-1} B_{-1} = \widetilde{\boldsymbol{t}}_2^{(i)^T} B_{-1}$, as above the cost is $\pi_n + 2\pi_m$.

For matrix P_i we have to compute four d-vectors and the first column, but we do not transform them.

$\boldsymbol{p}_1^{(i)} = -V\boldsymbol{u}_1^{(i)}$, the product by matrix V with an input vector already transformed costs $2\pi_n + 3\pi_m$.

$\boldsymbol{p}_2^{(i)} = -V\boldsymbol{u}_2^{(i)}$, as above the cost is $2\pi_n + 3\pi_m$.

$\boldsymbol{p}^{(i)} = \underline{F}\boldsymbol{e}_1 - V\boldsymbol{u}^{(i)}$, as above the cost is $2\pi_n + 3\pi_m$.

$\boldsymbol{q}_3^{(i)T} = \boldsymbol{e}_n^T \underline{F} P_i = \boldsymbol{e}_n^T \underline{F} - \widehat{\boldsymbol{f}}_1^T J U_i$, the product by U_i^T with an input vector already transformed costs $3\pi_n + 3\pi_m$.

$\boldsymbol{q}_4^{(i)T} = \boldsymbol{e}_n^T B_0^{-1} U_i = \widehat{\boldsymbol{f}}_2^T J U_i$, as above the cost is $3\pi_n + 3\pi_m$.

Now the vectors

$$\boldsymbol{w}_1 = V\boldsymbol{s}_2^{(i+1)}, \quad \boldsymbol{w}_2^T = \boldsymbol{t}_1^{(i+1)T} B_0^{-1}, \quad \boldsymbol{w}_3^T = \boldsymbol{t}_2^{(i+1)T} B_0^{-1},$$
$$\boldsymbol{w}_4^T = \boldsymbol{w}_2^T U_i, \quad \boldsymbol{w}_5^T = \boldsymbol{w}_3^T U_i$$

are computed. The product by V with an input vector already transformed costs $2\pi_n + 3\pi_m$. The products by B_0^{-T} with input vectors already transformed cost $5\pi_n$ each. The products by U_i^T with input vectors not yet transformed cost $4\pi_n + 3\pi_m$ each.

Then the vectors

$$\boldsymbol{y}_1 = P_i^{-1}\boldsymbol{w}_1, \quad \boldsymbol{y}_2 = P_i^{-1}\widehat{\boldsymbol{t}}_1, \quad \boldsymbol{y}_3 = P_i^{-1}\widehat{\boldsymbol{t}}_2,$$
$$\boldsymbol{y}_4^T = \boldsymbol{w}_4^T P_i^{-1}, \quad \boldsymbol{y}_5^T = \boldsymbol{w}_5^T P_i^{-1}, \quad \boldsymbol{y}_6^T = \boldsymbol{q}_4^{(i)T} P_i^{-1}$$

are obtained by solving six systems, three with matrix P_i and three with matrix P_i^T. The cost is $2\sigma_{m,4}(3)$.

For matrix S_{i+1}^{-1} we have to compute

$\widetilde{\boldsymbol{s}}_2^{(i+1)} = -B_0^{-1}\big(\boldsymbol{s}_2^{(i+1)} + U_i\boldsymbol{y}_1\big)$, one product by matrix U_i and one product by matrix B_0^{-1} are required. The input vectors have not yet been transformed. The costs are $10\pi_n + 3\pi_m$. The transformation of $\widetilde{\boldsymbol{s}}_2^{(i+1)}$ costs π_n.

$\widetilde{\boldsymbol{s}}_3^{(i+1)} = B_0^{-1}\big(\boldsymbol{f}_{-1} + U_i\boldsymbol{y}_2\big)$, as above, the cost is $11\pi_n + 3\pi_m$.

$\widetilde{\boldsymbol{s}}^{(i+1)} = B_0^{-1}\big(\boldsymbol{e}_1 + U_i\boldsymbol{y}_3\big)$, as above, the cost is $11\pi_n + 3\pi_m$.

$\widetilde{\boldsymbol{t}}_1^{(i+1)T} = \boldsymbol{w}_2^T + \boldsymbol{y}_4^T V$, the product by matrix V^T with an input vector not yet transformed costs $3\pi_n + 3\pi_m$. The transformation of $\widetilde{\boldsymbol{t}}_1^{(i+1)}$ costs π_n.

$\widetilde{\boldsymbol{t}}_2^{(i+1)T} = \boldsymbol{w}_3^T + \boldsymbol{y}_5^T V$, as above the cost is $4\pi_n + 3\pi_m$.

$\widetilde{\boldsymbol{t}}_3^{(i+1)T} = \widehat{\boldsymbol{f}}_2^T J + \boldsymbol{y}_6^T V$, as above the cost is $4\pi_n + 3\pi_m$.

At this point the d-representation of S_{i+1}^{-1} is available and we have only to compute the d-representation of U_{i+1}. For the left d-vectors we have

$\boldsymbol{u}_1^{(i+1)} = -\big(B_0 - B_1 S_{i+1}^{-1} B_{-1}\big)\boldsymbol{e}_1 = -\boldsymbol{f}_0 + B_1\widetilde{\boldsymbol{s}}_3^{(i+1)}$, the product by B_1 with an input vector already transformed costs π_n. The transformation of $\boldsymbol{u}_1^{(i+1)}$ costs π_n.

$\boldsymbol{u}_2^{(i+1)} = -B_1 S_i^{-1}\boldsymbol{s}_2^{(i+1)} = B_1\widetilde{\boldsymbol{s}}_2^{(i+1)}$, as above the cost is $2\pi_n$.

The right d-vectors of U_{i+1} are obtained by truncating those of S_{i+1}^{-1} and must be transformed, with cost $2\pi_m$.

For the first column of U_{i+1}, i.e. $\boldsymbol{u}^{(i+1)} = B_1 S_{i+1}^{-1}\boldsymbol{e}_{n-m+1}$, we have to compute one product by S_{i+1}^{-1}, with input vector already transformed, and one product by B_1. The first one costs $5\pi_n$, the second one costs $2\pi_n$. The transformation of $\boldsymbol{u}^{(i+1)}$ costs π_n.

4.4.2 Substitution phase

For the forward phase $\boldsymbol{z}_i = S_{i-1}^{-1}(\boldsymbol{b}_i - B_1\boldsymbol{z}_{i-1})$, one product by matrix B_1 and one product by matrix S_{i-1}^{-1} are required. The input vectors have not yet been transformed. The costs are $2\pi_n$ and $6\pi_n$.

For backward phase $\boldsymbol{x}_i = \boldsymbol{z}_i - S_{i-1}^{-1}B_{-1}\boldsymbol{x}_{i+1}$, one product by matrix B_{-1} and one product by matrix S_{i-1}^{-1} are required. The input vectors have not yet been transformed. The costs are $2\pi_m$ and $6\pi_n$.

4.4.3 Overall cost

For the transformation phase the cost of the ith step is

$$t_i = 91\pi_n + 48\pi_m + 2\sigma_{m,4}(3),$$

for the substitution phase the cost is

$$s_i = 14\pi_n + 2\pi_m.$$

Both costs must be multiplied by N. According to (10) we assume $\sigma_{m,4}(3) = 27/2m^2$, we have

$$c_{\mathsf{ge}} = (t_i + s_i)N = (105\pi_n + 50\pi_m + 27m^2)N. \tag{16}$$

5 Algorithm cr

Various different methods go under the name of *cyclic reduction.* They all share the characteristic of building a sequence of problems having a progressively reduced size. When the size is sufficiently small, the reduced problem is solved and from its solution the solution of the original problem is reconstructed through backward steps. We describe here an implementation of the method proposed in [3], where the reduction of size is achieved by applying a block Gaussian elimination to a suitably permuted system.

Let $N = 2^p$, with $p \geq 2$. Denote by

$$A^{(0)}\boldsymbol{x}^{(0)} = \boldsymbol{b}^{(0)}$$

the given system (1), where $A^{(0)} = A$ is the block tridiagonal Toeplitz matrix (3) and $\boldsymbol{b}^{(0)} = \boldsymbol{b}$. The three blocks are renamed as follows

$$H_0^{(0)} = \widehat{H}_0^{(0)} = B_0, \quad H_1^{(0)} = B_1, \quad H_{-1}^{(0)} = B_{-1}. \tag{17}$$

For $i = 0, \ldots, p-1$, the matrices

$$\begin{aligned} H_0^{(i+1)} &= H_0^{(i)} - H_{-1}^{(i)} {H_0^{(i)}}^{-1} H_1^{(i)} - H_1^{(i)} {H_0^{(i)}}^{-1} H_{-1}^{(i)}, \\ H_1^{(i+1)} &= -H_1^{(i)} {H_0^{(i)}}^{-1} H_1^{(i)}, \\ H_{-1}^{(i+1)} &= -H_{-1}^{(i)} {H_0^{(i)}}^{-1} H_{-1}^{(i)}, \\ \widehat{H}^{(i+1)} &= \widehat{H}^{(i)} - H_{-1}^{(i)} {H_0^{(i)}}^{-1} H_1^{(i)}, \end{aligned} \tag{18}$$

are computed. In [3] it is proved that all these matrices are Toeplitz-like, with d-rank equal to 4 for $H_0^{(i)}$, to 2 for $H_1^{(i)}$ and $H_{-1}^{(i)}$, to 3 for $\widehat{H}^{(i)}$ when $i > 0$. Moreover $H_0^{(i)}$, $H_1^{(i)}$, $H_{-1}^{(i)}$ are persymmetric and $H_{-1}^{(i)}$ has nonzero elements only in a lower left square block $R^{(i)}$ of size m. Hence it can be written as

$$H_{-1}^{(i)} = \underline{F} R^{(i)} \overline{F}^T,$$

where

$$R^{(0)} = R \quad \text{and} \quad R^{(i+1)} = -R^{(i)} \overline{F}^T {H_0^{(i)}}^{-1} \underline{F} R^{(i)}.$$

The cyclic reduction is based on the sequence of systems

$$A^{(i)}\boldsymbol{x}^{(i)} = \boldsymbol{b}^{(i)}, \quad \text{for} \quad i = 1, \ldots, p, \tag{19}$$

of order $2^{p-i}n$, recursively obtained in this way:

1. let Π_i be the even/odd block permutation matrix of order 2^{p-i}, with blocks of order n. Problem (19) is equivalent to

$$\widetilde{A}^{(i)}\widetilde{\boldsymbol{x}}^{(i)} = \widetilde{\boldsymbol{b}}^{(i)}, \quad \widetilde{A}^{(i)} = \Pi_i A^{(i)} \Pi_i^T,\ \widetilde{\boldsymbol{x}}^{(i)} = \Pi_i \boldsymbol{x}^{(i)},\ \widetilde{\boldsymbol{b}}^{(i)} = \Pi_i \boldsymbol{b}^{(i)}. \tag{20}$$

2. Partition vectors $\widetilde{\boldsymbol{x}}^{(i)}$ and $\widetilde{\boldsymbol{b}}^{(i)}$ into two halves, as follows

$$\widetilde{\boldsymbol{x}}^{(i)} = \begin{bmatrix} \boldsymbol{x}_+^{(i)} \\ \boldsymbol{x}_-^{(i)} \end{bmatrix}, \qquad \widetilde{\boldsymbol{b}}^{(i)} = \begin{bmatrix} \boldsymbol{b}_+^{(i)} \\ \boldsymbol{b}_-^{(i)} \end{bmatrix}.$$

3. Apply one step of the block Gaussian elimination to (20) seen as a 2×2 block system, obtaining a problem of the form

$$\begin{bmatrix} D^{(i)} & U^{(i)} \\ O & A^{(i+1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_+^{(i)} \\ \boldsymbol{x}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_+^{(i)} \\ \boldsymbol{b}^{(i+1)} \end{bmatrix},$$

where
(a) $D^{(i)}$ and $U^{(i)}$ are block Toeplitz matrices of order 2^{p-i-1}, $D^{(i)}$ being a block diagonal matrix with blocks $H_0^{(i)}$ on the diagonal and $U^{(i)}$ being an upper block bidiagonal matrix with blocks $H_1^{(i)}$ on the diagonal and $H_{-1}^{(i)}$ on the upper parallel.
(b) $A^{(i+1)}$ is a block tridiagonal matrix of order 2^{p-i-1}. It has still a block Toeplitz structure except for the northwest corner block which is $\widehat{H}^{(i+1)}$. The other blocks are $H_0^{(i+1)}$ on the diagonal, $H_1^{(i+1)}$ on the lower parallel and $H_{-1}^{(i+1)}$ on the upper parallel.
(c) $\boldsymbol{b}^{(i+1)}$, $i = 0, \ldots, p-1$, is the vector whose jth subvector of size n is

$$\boldsymbol{b}_j^{(i+1)} = \boldsymbol{b}_{2j-1}^{(i)} - H_1^{(i)} {H_0^{(i)}}^{-1} \boldsymbol{b}_{2j-2}^{(i)} - H_{-1}^{(i)} {H_0^{(i)}}^{-1} \boldsymbol{b}_{2j}^{(i)}, \quad j = 1, \ldots, 2^{p-i-1}, \tag{21}$$

with $\boldsymbol{b}_j^{(i)} = \boldsymbol{0}$ for $j = 0$.

Once the vector $\boldsymbol{x}^{(i+1)}$ has been obtained by solving the system $A^{(i+1)}\boldsymbol{x}^{(i+1)} = \boldsymbol{b}^{(i+1)}$, the subvectors of $\boldsymbol{x}^{(i)}$, $i = p-1, \ldots, 0$, are given by

$$\begin{cases} \boldsymbol{x}_{2j-1}^{(i)} = \boldsymbol{x}_j^{(i+1)}, \\ \boldsymbol{x}_{2j}^{(i)} = {H_0^{(i)}}^{-1}\Big(\boldsymbol{b}_{2j}^{(i)} - H_1^{(i)} \boldsymbol{x}_j^{(i+1)} - H_{-1}^{(i)} \boldsymbol{x}_{j+1}^{(i+1)}\Big), \end{cases} \quad \text{for} \quad j = 1, \ldots, 2^{p-i-1}, \tag{22}$$

with $\boldsymbol{x}_{j+1}^{(i+1)} = \boldsymbol{0}$ for $j = 2^{p-i-1}$.

After p steps of reduction, system (19) has order n and is

$$\widehat{H}^{(p)}\boldsymbol{x}_1^{(p)} = \boldsymbol{b}_1^{(p)}. \tag{23}$$

Now $\boldsymbol{x}_1^{(p)}$ can be computed directly. Then $p-1$ backward steps are performed with (22), computing $\boldsymbol{x}_j^{(i)}$ for $i = p-1, \ldots, 0$, until $\boldsymbol{x}^{(0)} = \boldsymbol{x}$ is obtained. Hence method `cr` appears to consist of two phases: a transformation phase where the matrices $H_0^{(i)}$, $H_1^{(i)}$, $H_{-1}^{(i)}$ and $\widehat{H}^{(i)}$ are constructed, and a substitution phase where the vectors $\boldsymbol{b}_j^{(i)}$ are constructed in a forward way and the vectors $\boldsymbol{x}_j^{(i)}$ are constructed in a backward way.

Remark 5.1 Algorithm `cr` can be seen as a block Gaussian elimination applied to a suitable permuted matrix $\widetilde{A}$. In fact, let Γ_i be the matrix obtained by replacing in the identity matrix of order N the last principal block of order 2^{p-i} with Π_i. Defining

$$\Omega = \Gamma_{p-1} \ldots \Gamma_1 \Gamma_0,$$

the given system (1) is equivalent to

$$\widetilde{A}\,\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}, \quad \widetilde{A} = \Omega\, A\, \Omega^T, \quad \widetilde{\boldsymbol{x}} = \Omega\, \boldsymbol{x}, \quad \widetilde{\boldsymbol{b}} = \Omega\, \boldsymbol{b}.$$

Applying block Gaussian elimination we see that

$$\widetilde{A} = L\,U,$$

where L is lower block triangular and U is upper block triangular. Matrix L has identity principal blocks and one or two blocks of the form $H_1^{(i)} {H_0^{(i)}}^{-1}$ and $H_{-1}^{(i)} {H_0^{(i)}}^{-1}$ in the columns of the lower part (exactly elements with index i stay on 2^{p-i-1} consecutive columns). Matrix U has principal blocks equal to $H_0^{(i)}$ (exactly blocks with index i stay on 2^{p-i-1} consecutive places) and a last principal block equal to $\widehat{H}_0^{(p)}$ and blocks equal to $H_{-1}^{(i)}$ or $H_1^{(i)}$ in the upper part.

5.1 Sketchy implementation

The algorithm can be implemented in the following way. All the computations are carried out by using the d-descriptions of the involved matrices.

Function `cr` $(A, N, \boldsymbol{b})$

```
1. let $H_0^{(0)}$, $H_1^{(0)}$, $H_{-1}^{(0)}$ and $\widehat{H}^{(0)}$ be defined as in (17)
2. compute ${H_0^{(0)}}^{-1}$,
3. for $i = 0, \ldots, p-1$  compute $H_0^{(i+1)}$, $H_1^{(i+1)}$, $H_{-1}^{(i+1)}$, $\widehat{H}^{(i+1)}$
       by applying (18) and compute ${H_0^{(i+1)}}^{-1}$,
4. for $i = 0, \ldots, p-1$,  for $j = 1, \ldots 2^{p-i-1}$
       compute $\boldsymbol{b}_j^{(i+1)}$ by applying (21),
5. compute $\widehat{H}^{(p)-1}$,
6. compute $\boldsymbol{x}_1^{(p)} = \widehat{H}^{(p)-1}\boldsymbol{b}_1^{(p)}$,
7. for $i = p-1,\dots,0$,   for $j = 1,\dots 2^{p-i}$
       compute $\boldsymbol{x}_j^{(i)}$ by applying (22).
```

5.2 D-description of the matrices

For the d-description of the matrices of algorithm `cr` we need the vectors of length n

$\boldsymbol{h}_0^{(i)} = H_0^{(i)}\boldsymbol{e}_1$ is the first column of $H_0^{(i)}$,

$\boldsymbol{v}_0^{(i)} = \dfrac{1}{a_n}H_1^{(i)}\boldsymbol{e}_1$ is the first column of $\dfrac{1}{a_n}H_1^{(i)}$,

$\widehat{\boldsymbol{h}}^{(i)} = \widehat{H}^{(i)}\boldsymbol{e}_1$ is the first column of $\widehat{H}^{(i)}$,

$\boldsymbol{t}^{(i)} = J\widehat{H}^{(i)T}\boldsymbol{e}_n$, i.e. $\boldsymbol{t}^{(i)T}$ is the reversed last row of $\widehat{H}^{(i)}$,

and the vector of length m

$\boldsymbol{r}^{(i)} = R^{(i)}\overline{\boldsymbol{e}}_1$ is the first column of $R^{(i)}$.

The following recursive relations hold

$$\boldsymbol{h}_0^{(i+1)} = \boldsymbol{h}_0^{(i)} - a_n H_{-1}^{(i)} H_0^{(i)-1}\boldsymbol{v}_0^{(i)} - H_1^{(i)} H_0^{(i)-1}\underline{F}\,\boldsymbol{r}^{(i)}, \qquad \boldsymbol{h}_0^{(0)} = \boldsymbol{f}_0,$$

$$\boldsymbol{v}_0^{(i+1)} = -H_1^{(i)} H_0^{(i)-1}\boldsymbol{v}_0^{(i)}, \qquad \boldsymbol{v}_0^{(0)} = \boldsymbol{e}_1,$$

$$\widehat{\boldsymbol{h}}^{(i+1)} = \widehat{\boldsymbol{h}}^{(i)} - a_n H_{-1}^{(i)} H_0^{(i)-1}\boldsymbol{v}_0^{(i)}, \qquad \widehat{\boldsymbol{h}}^{(0)} = \boldsymbol{f}_0,$$

$$\boldsymbol{t}^{(i+1)} = \boldsymbol{t}^{(i)} - H_1^{(i)} H_0^{(i)-1}\underline{F}\,\boldsymbol{r}^{(i)}, \qquad \boldsymbol{t}^{(0)} = \boldsymbol{f}_0,$$

$$\boldsymbol{r}^{(i+1)} = -R^{(i)}\overline{F}^T H_0^{(i)-1}\underline{F}\,\boldsymbol{r}^{(i)}, \qquad \boldsymbol{r}^{(0)} = \underline{F}^T\boldsymbol{f}_{-1}.$$

For simplicity sake we introduce also the vector of length m

$$\boldsymbol{v}_1^{(i)} = \frac{1}{a_n}\underline{F}^T\left(\boldsymbol{h}_0^{(i)} - \boldsymbol{t}^{(i)}\right),$$

which satisfies the recursion

$$\boldsymbol{v}_1^{(i+1)} = \boldsymbol{v}_1^{(i)} - R^{(i)}\overline{F}^T H_0^{(i)-1}\boldsymbol{v}_0^{(i)}, \qquad \boldsymbol{v}_1^{(0)} = \boldsymbol{0}. \tag{24}$$

It is easy to show that $\boldsymbol{h}_0^{(i)}$ and $\boldsymbol{t}^{(i)}$ have the first $n-m$ entries in common. Hence, if $\boldsymbol{t}^{(i)}$ and $\boldsymbol{v}_1^{(i)}$ are known, there is no need to compute $\boldsymbol{h}_0^{(i)}$ using its recursive relation because it can be recovered in this way

$$\boldsymbol{h}_0^{(i)} = \boldsymbol{t}^{(i)} + a_n\underline{F}\,\boldsymbol{v}_1^{(i)}. \tag{25}$$

The d-description of $H_0^{(i)}$, $H_1^{(i)}$, $H_{-1}^{(i)}$ and $\widehat{H}^{(i)}$ can be easily proved by induction or derived from [3].

Matrix $H_0^{(i)}$ size $n \times n$ d-rank=4 (d-rank=2 if $i=0$)

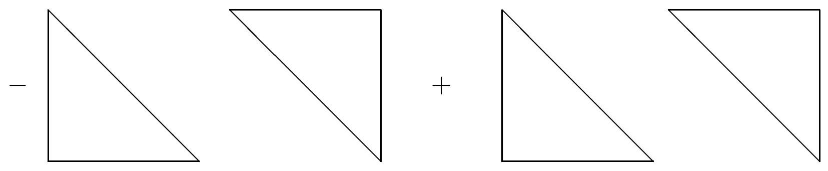
$$\Delta(H_0^{(i)}) = -\underline{F}\,\boldsymbol{r}^{(i)}\boldsymbol{v}_0^{(i)T}J - \boldsymbol{h}_0^{(i)}\boldsymbol{v}_1^{(i)T}\underline{F}^T J + \underline{F}\,\boldsymbol{v}_1^{(i)}\boldsymbol{h}_0^{(i)T}J + \boldsymbol{v}_0^{(i)}\boldsymbol{r}^{(i)T}\underline{F}^T J.$$

The first column of $H_0^{(i)}$ is the d-vector $\boldsymbol{h}_0^{(i)}$. The d-representation of this matrix is not shown here because it is not needed in the following.

Matrix $H_1^{(i)}$ size $n \times n$ d-rank=2

$$\Delta(H_1^{(i)}) = -\boldsymbol{t}^{(i)}\boldsymbol{v}_0^{(i)T}J + \boldsymbol{v}_0^{(i)}\boldsymbol{t}^{(i)T}J.$$

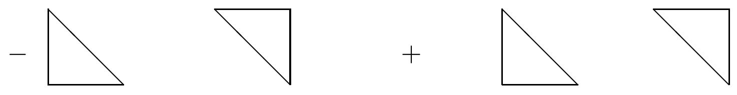
The first column of $H_1^{(i)}$ is the d-vector $a_n\boldsymbol{v}_0^{(i)}$. The d-representation of $H_1^{(i)}$ has the form


$L(\boldsymbol{t}^{(i)})$ $L^T(ZJ\boldsymbol{v}_0^{(i)})$ $L(\boldsymbol{v}_0^{(i)})$ $a_nI + L^T(ZJ\boldsymbol{t}^{(i)})$

Matrix $R^{(i)}$ size $m \times m$ d-rank=2

$$\Delta(R^{(i)}) = -\boldsymbol{r}^{(i)}\boldsymbol{v}_1^{(i)T}J + \boldsymbol{v}_1^{(i)}\boldsymbol{r}^{(i)T}J.$$

The first column of $R^{(i)}$ is the d-vector $\boldsymbol{r}^{(i)}$. The d-representation of $R^{(i)}$ has the form


$L(\boldsymbol{r}^{(i)})$ $-I + L^T(ZJ\boldsymbol{v}_1^{(i)})$ $L(\boldsymbol{v}_1^{(i)})$ $L^T(ZJ\boldsymbol{r}^{(i)})$

Matrix $\widehat{H}^{(i)}$ size $n \times n$ d-rank=3 (d-rank=2 if $i=0$)

$$\Delta(\widehat{H}^{(i)}) = \boldsymbol{e}_1\boldsymbol{f}_{-1}^T J - \underline{F}\,\boldsymbol{r}^{(i)}\boldsymbol{v}_0^{(i)T}J + \underline{F}\,\boldsymbol{v}_1^{(i)}\boldsymbol{t}^{(i)T}J.$$

The first column of $\widehat{H}^{(i)}$ is $\widehat{\boldsymbol{h}}^{(i)}$ and is different from the other vectors of the d-description. Matrix $\widehat{H}^{(i)}$ never enters into a multiplication. Hence the d-description of this matrix is not shown. Only the last one $\widehat{H}^{(p)}$ is used in (23).

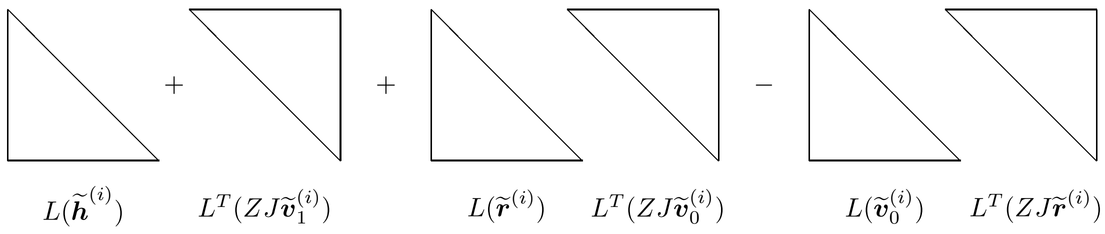
Matrix ${H_0^{(i)}}^{-1}$ size $n \times n$ d-rank=4 (d-rank=2 if $i = 0$)

$$\Delta({H_0^{(i)}}^{-1}) = \widetilde{\boldsymbol{r}}^{(i)}\, \widetilde{\boldsymbol{v}}_0^{(i)T} J + \boldsymbol{e}_1\, \widetilde{\boldsymbol{v}}_1^{(i)T} J - \widetilde{\boldsymbol{v}}_1^{(i)}\, \boldsymbol{e}_n^T - \widetilde{\boldsymbol{v}}_0^{(i)}\, \widetilde{\boldsymbol{r}}^{(i)T} J,$$

where

$$\widetilde{\boldsymbol{r}}^{(i)} = {H_0^{(i)}}^{-1} \underline{F}\, \boldsymbol{r}^{(i)}, \qquad \widetilde{\boldsymbol{v}}_0^{(i)} = {H_0^{(i)}}^{-1} \boldsymbol{v}_0^{(i)}, \qquad \widetilde{\boldsymbol{v}}_1^{(i)} = {H_0^{(i)}}^{-1} \underline{F}\, \boldsymbol{v}_1^{(i)}.$$

The first column of ${H_0^{(i)}}^{-1}$ is $\widetilde{\boldsymbol{h}}^{(i)} = {H_0^{(i)}}^{-1} \boldsymbol{e}_1$ and is different from the other vectors of the d-description. The d-representation of ${H_0^{(i)}}^{-1}$ has the form


$+$
$+$
$-$
$L(\widetilde{\boldsymbol{h}}^{(i)})$
$L^T(ZJ\widetilde{\boldsymbol{v}}_1^{(i)})$
$L(\widetilde{\boldsymbol{r}}^{(i)})$
$L^T(ZJ\widetilde{\boldsymbol{v}}_0^{(i)})$
$L(\widetilde{\boldsymbol{v}}_0^{(i)})$
$L^T(ZJ\widetilde{\boldsymbol{r}}^{(i)})$

5.3 Cost of matrix by vector product

Table 2 lists the cost of computing matrix by vector products, for the matrices used in algorithm `cr`. As already said, this cost does not include the cost for the transformation of the vectors of the d-description of the matrix itself and the cost of transforming the input vector.

matrix	cost
${H_0^{(i)}}^{-1}$	$5\pi_n$
$H_1^{(i)}$	$5\pi_n$
$R^{(i)}$	$5\pi_m$

Table 2: Cost of computing matrix by vector products for algorithm `cr`.

5.4 Cost of algorithm `cr`

5.4.1 Transformation phase

(a) Initially we set

$$\boldsymbol{h}_0^{(0)} = \widehat{\boldsymbol{h}}^{(0)} = \boldsymbol{t}^{(0)} = \boldsymbol{f}_0, \quad \boldsymbol{v}_0^{(0)} = \boldsymbol{e}_1, \quad \boldsymbol{r}^{(0)} = \underline{F}^T \boldsymbol{f}_{-1}, \quad \boldsymbol{v}_1^{(0)} = \boldsymbol{0}.$$

(b) We assume that at the ith step, $i = 1, \ldots, p$, the vectors $\boldsymbol{v}_0^{(i-1)}$, $\boldsymbol{r}^{(i-1)}$, $\boldsymbol{v}_1^{(i-1)}$ and $\boldsymbol{t}^{(i-1)}$ have already been computed and transformed, while vectors $\boldsymbol{h}_0^{(i-1)}$ and $\widehat{\boldsymbol{h}}^{(i-1)}$ have been computed but not transformed, since their transformations are not needed.

First the vectors

$$\widetilde{\boldsymbol{h}}^{(i-1)} = H_0^{(i-1)^{-1}} \boldsymbol{e}_1, \qquad \widetilde{\boldsymbol{r}}^{(i-1)} = H_0^{(i-1)^{-1}} \underline{F}\, \boldsymbol{r}^{(i-1)},$$
$$\widetilde{\boldsymbol{v}}_0^{(i-1)} = H_0^{(i-1)^{-1}} \boldsymbol{v}_0^{(i-1)}, \qquad \widetilde{\boldsymbol{v}}_1^{(i-1)} = H_0^{(i-1)^{-1}} \underline{F}\, \boldsymbol{v}_1^{(i-1)},$$

are computed by solving four systems with the same matrix $H_0^{(i-1)}$, whose complete d-description is available. The corresponding cost is $\sigma_{n,4}(4)$. Now we have the d-representation of $H_0^{(i-1)^{-1}}$. The four vectors are transformed with cost $4\pi_n$.

Then the vectors

$$\boldsymbol{v}_0^{(i)} = -H_1^{(i-1)} \widetilde{\boldsymbol{v}}_0^{(i-1)}, \qquad \boldsymbol{z}_1 = H_1^{(i-1)} \widetilde{\boldsymbol{r}}^{(i-1)},$$
$$\boldsymbol{z}_2 = R^{(i-1)} \overline{F}^T \widetilde{\boldsymbol{v}}_0^{(i-1)}, \qquad \boldsymbol{r}^{(i)} = -R^{(i-1)} \overline{F}^T \widetilde{\boldsymbol{r}}^{(i-1)},$$

are computed by multiplication. The input vectors of length n have already been transformed, those of length m have not yet been transformed. Hence the products by $H_1^{(i-1)}$ cost $5\pi_n$ each and the products by $R^{(i-1)}$ cost $6\pi_m$ each. We get

$$\boldsymbol{t}^{(i)} = \boldsymbol{t}^{(i-1)} - \boldsymbol{z}_1, \qquad \boldsymbol{v}_1^{(i)} = \boldsymbol{v}_1^{(i-1)} - \boldsymbol{z}_2, \qquad \widehat{\boldsymbol{h}}^{(i)} = \widehat{\boldsymbol{h}}^{(i-1)} - a_n \underline{F}\, \boldsymbol{z}_2.$$

Vectors $\boldsymbol{v}_0^{(i)}$ and $\boldsymbol{t}^{(i)}$ are transformed with cost $2\pi_n$, vectors $\boldsymbol{r}^{(i)}$ and $\boldsymbol{v}_1^{(i)}$ are transformed with cost $2\pi_m$.

Vector $\boldsymbol{h}_0^{(i)}$ is computed using (25).

5.4.2 Substitution phase

At the ith step first the vectors

$$w_j^{(i)} = H_0^{(i)^{-1}} \boldsymbol{b}_{2j}^{(i)}$$

are computed for $j = 1, \ldots, 2^{p-i-1}$. The input vectors have not yet been transformed. Hence the cost is $6\pi_n$ for each j.

Then the vectors

$$\boldsymbol{b}_j^{(i+1)} = \boldsymbol{b}_{2j-1}^{(i)} - H_1^{(i)} \boldsymbol{w}_{j-1}^{(i)} - \underline{F}\, R^{(i)} \overline{F}^T \boldsymbol{w}_j^{(i)}$$

are computed. The input vectors have not yet been transformed. Hence the product by $H_1^{(i)}$ costs $6\pi_n$ and the product by $R^{(i)}$ costs $6\pi_m$, i.e. the cost is $6\pi_n + 6\pi_m$ for each j.

The same cost holds for the computation of

$$\boldsymbol{x}_{2j}^{(i)} = {H_0^{(i)}}^{-1}\Big(\boldsymbol{b}_{2j}^{(i)} - H_1^{(i)}\boldsymbol{x}_j^{(i+1)} - \underline{F}\,R^{(i)}\overline{F}^T\boldsymbol{x}_{j+1}^{(i+1)}\Big).$$

At the first step $\boldsymbol{x}_1^{(i)}$ is computed by solving system (23) with matrix $\widehat{H}^{(p)}$ whose d-description is available. The corresponding cost is not considered.

5.4.3 Overall cost

For the transformation phase the cost of the ith step is

$$t_i = 16\pi_n + 14\pi_m + \sigma_{n,4}(4) \tag{26}$$

and must be multiplied by $\log N$. For the substitution phase the cost is

$$s = (24\pi_n + 12\pi_m)N.$$

According to (10) we assume $\sigma_{n,4}(4) = 14.5n^2$. We have

$$c_{\mathtt{cr}} = t_i \log N + s = (16\pi_n + 14\pi_m + 14.5n^2)\log N + (24\pi_n + 12\pi_m)N. \tag{27}$$

6 Algorithm `mcr`

It is evident that the main part of the computational cost of the transformation phase of algorithm `cr` lies in the computation of ${H_0^{(i)}}^{-1}$. This cost can be reduced if we exploit the fact that each matrix $H_0^{(i+1)}$ differs from $H_0^{(i)}$ by a correction of low rank (at most $2m$). Using SMW formula, as suggested in [17], we propose the following modification.

Let us consider the matrices

$$U_1^{(i)} = H_1^{(i)}{H_0^{(i)}}^{-1}\underline{F}R^{(i)} \quad \text{and} \quad V_1^{(i)} = R^{(i)}\overline{F}^T{H_0^{(i)}}^{-1}H_1^{(i)},$$

and the matrices

$$U^{(i)} = \left[\,\underline{F} \mid U_1^{(i)}\,\right], \quad \text{and} \quad V^{(i)} = \left[\begin{array}{c} V_1^{(i)} \\ \overline{F}^T \end{array}\right].$$

Then we can write

$$H_0^{(i+1)} = H_0^{(i)} - U^{(i)}V^{(i)},$$

and by SMW formula we have

$${H_0^{(i+1)}}^{-1} = {H_0^{(i)}}^{-1} + {H_0^{(i)}}^{-1}U^{(i)}{P^{(i)}}^{-1}V^{(i)}{H_0^{(i)}}^{-1},$$

where

$$P^{(i)} = I - V^{(i)}{H_0^{(i)}}^{-1}U^{(i)}. \tag{28}$$

In this way the computation of the inverse of the $n\times n$ matrix $H_0^{(i+1)}$ is replaced by the computation of the inverse of the $2m\times 2m$ matrix $P^{(i)}$. This modification of algorithm `cr` will be called `mcr`.

6.1 D-description of the matrices

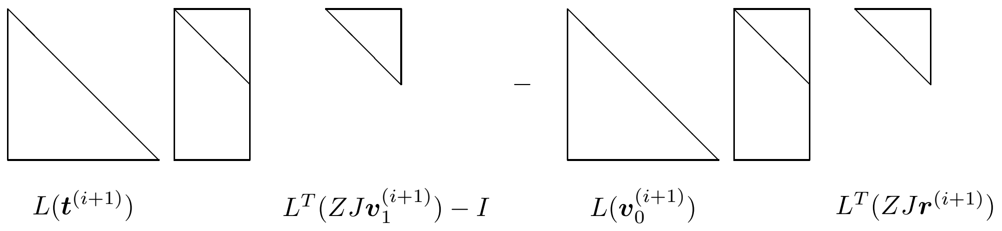
Matrix $U_1^{(i)} = H_1^{(i)} {H_0^{(i)}}^{-1} \underline{F} R^{(i)}$ size $n \times m$ d-rank=4

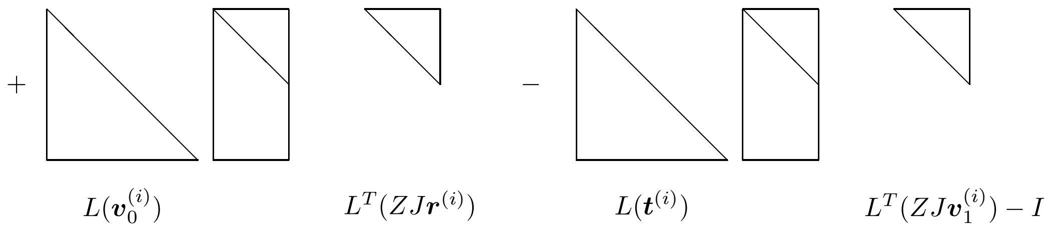
$$\Delta(U_1^{(i)}) = \left[\boldsymbol{t}^{(i+1)} \boldsymbol{v}_1^{(i+1)T} - \boldsymbol{v}_0^{(i+1)} \boldsymbol{r}^{(i+1)T} + \boldsymbol{v}_0^{(i)} \boldsymbol{r}^{(i)T} - \boldsymbol{t}^{(i)} \boldsymbol{v}_1^{(i)T} \right] J.$$

The first column of $U_1^{(i)}$ is

$$U_1^{(i)} \overline{\boldsymbol{e}}_1 = \boldsymbol{t}^{(i)} - \boldsymbol{t}^{(i+1)}$$

and is not a d-vector, but both $\boldsymbol{t}^{(i)}$ and $\boldsymbol{t}^{(i+1)}$ are d-vectors, and this fact can be exploited for the d-representation of $U_1^{(i)}$ which has the form


$-$
$L(\boldsymbol{t}^{(i+1)})$
$L^T(ZJ\boldsymbol{v}_1^{(i+1)}) - I$
$L(\boldsymbol{v}_0^{(i+1)})$
$L^T(ZJ\boldsymbol{r}^{(i+1)})$


$+$
$-$
$L(\boldsymbol{v}_0^{(i)})$
$L^T(ZJ\boldsymbol{r}^{(i)})$
$L(\boldsymbol{t}^{(i)})$
$L^T(ZJ\boldsymbol{v}_1^{(i)}) - I$

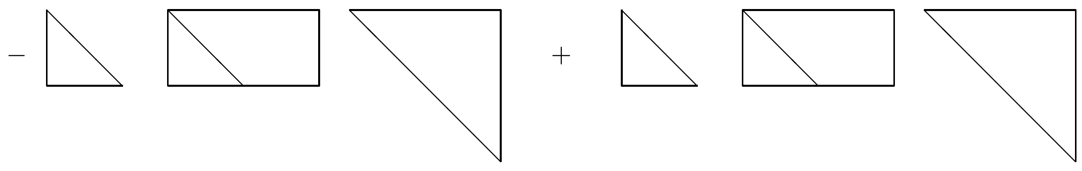
Matrix $V_1^{(i)} = R^{(i)} \overline{F}^T {H_0^{(i)}}^{-1} H_1^{(i)} = J U_1^{(i)T} J$ size $m \times n$ d-rank=4

$$\Delta(V_1^{(i)}) = -J\big[\Delta(U_1^{(i)})\big]^T J = \left[-\boldsymbol{v}_1^{(i+1)} \boldsymbol{t}^{(i+1)T} + \boldsymbol{r}^{(i+1)} \boldsymbol{v}_0^{(i+1)T} - \boldsymbol{r}^{(i)} \boldsymbol{v}_0^{(i)T} + \boldsymbol{v}_1^{(i)} \boldsymbol{t}^{(i)T} \right] J.$$

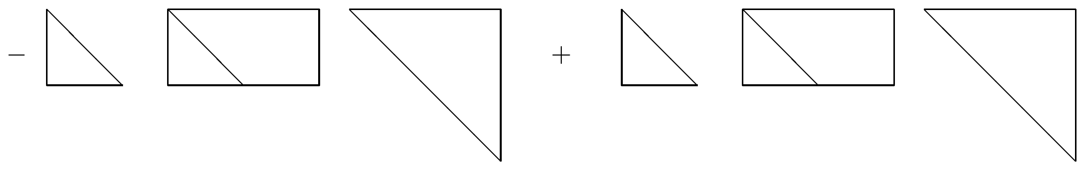
From (24) it follows that the first column of $V_1^{(i)}$ is

$$V_1^{(i)} \boldsymbol{e}_1 = R^{(i)} \overline{F}^T {H_0^{(i)}}^{-1} H_1^{(i)} \boldsymbol{e}_1 = a_n \big(\boldsymbol{v}_1^{(i)} - \boldsymbol{v}_1^{(i+1)} \big).$$

It is not a d-vector, but both $\boldsymbol{v}_1^{(i)}$ and $\boldsymbol{v}_1^{(i+1)}$ are d-vectors, and this fact can be exploited for the d-representation of $V_1^{(i)}$ which has the form



$-$
$+$
$L(\boldsymbol{v}_1^{(i+1)})$
$a_nI + L^T(ZJ\boldsymbol{t}^{(i+1)})$
$L(\boldsymbol{r}^{(i+1)})$
$L^T(ZJ\boldsymbol{v}_0^{(i+1)})$



$-$
$+$
$L(\boldsymbol{r}^{(i)})$
$L^T(ZJ\boldsymbol{v}_0^{(i)})$
$L(\boldsymbol{v}_1^{(i)})$
$a_nI + L^T(ZJ\boldsymbol{t}^{(i)})$

Matrix $P^{(i)} = I - V^{(i)}{H_0^{(i)}}^{-1}U^{(i)}$ size $2m \times 2m$ d-rank=8 (d-rank=6 if $i = 0$).

For $i > 0$ we have

$$\Delta(P^{(i)}) = \sum_{j=1}^{8} \boldsymbol{p}_j^{(i)}\boldsymbol{q}_j^{(i)T},$$

where

$$\begin{aligned}
\boldsymbol{p}_1^{(i)} &= -V^{(i)}{H_0^{(i)}}^{-1}\boldsymbol{v}_0^{(i+1)}, & \boldsymbol{q}_1^{(i)T} &= \boldsymbol{p}_2^{(i)T}J,\\
\boldsymbol{p}_2^{(i)} &= -\overline{F}_{2m}\boldsymbol{r}^{(i+1)}, & \boldsymbol{q}_2^{(i)T} &= -\boldsymbol{p}_1^{(i)T}J,\\
\boldsymbol{p}_3^{(i)} &= -V^{(i)}\widetilde{\boldsymbol{v}}_0^{(i)}, & \boldsymbol{q}_3^{(i)T} &= \boldsymbol{p}_4^{(i)T}J,\\
\boldsymbol{p}_4^{(i)} &= \overline{F}_{2m}\boldsymbol{r}^{(i)} - V^{(i)}\widetilde{\boldsymbol{r}}^{(i)}, & \boldsymbol{q}_4^{(i)T} &= -\boldsymbol{p}_3^{(i)T}J,\\
\boldsymbol{p}_5^{(i)} &= -V^{(i)}{H_0^{(i)}}^{-1}H_1^{(i)}\widetilde{\boldsymbol{r}}^{(i)}, & \boldsymbol{q}_5^{(i)T} &= \boldsymbol{p}_6^{(i)T}J,\\
\boldsymbol{p}_6^{(i)} &= -\overline{F}_{2m}\boldsymbol{v}_1^{(i+1)} + \underline{F}_{2m}\overline{\boldsymbol{e}}_1, & \boldsymbol{q}_6^{(i)T} &= -\boldsymbol{p}_5^{(i)T}J,\\
\boldsymbol{p}_7^{(i)} &= \overline{F}_{2m}\big(\boldsymbol{v}_1^{(i+1)} - \boldsymbol{v}_1^{(i)}\big) + V^{(i)}\widetilde{\boldsymbol{v}}_1^{(i)}, & \boldsymbol{q}_7^{(i)T} &= \boldsymbol{p}_8^{(i)T}J,\\
\boldsymbol{p}_8^{(i)} &= \underline{F}_{2m}\overline{\boldsymbol{e}}_1, & \boldsymbol{q}_8^{(i)T} &= -\boldsymbol{p}_7^{(i)T}J.
\end{aligned}$$

The first column of $P^{(i)}$ is $P^{(i)}\overline{F}_{2m}\overline{\boldsymbol{e}}_1 = \overline{F}_{2m}\overline{\boldsymbol{e}}_1 - V^{(i)}{H_0^{(i)}}^{-1}\boldsymbol{e}_{n-m+1}$ and is not a d-vector.

For $i = 0$ the last four pairs of vectors are replaced by

$$\begin{aligned}
\boldsymbol{p}_5^{(0)} &= -V^{(0)}{H_0^{(0)}}^{-1}H_1^{(0)}\widetilde{\boldsymbol{r}}^{(0)} + \underline{F}_{2m}\overline{\boldsymbol{e}}_1, & \boldsymbol{q}_5^{(0)T} &= -\boldsymbol{p}_6^{(0)T}J,\\
\boldsymbol{p}_6^{(0)} &= -\overline{F}_{2m}\boldsymbol{v}_1^{(1)} + \underline{F}_{2m}\overline{\boldsymbol{e}}_1, & \boldsymbol{q}_6^{(0)T} &= \boldsymbol{p}_5^{(0)T}J.
\end{aligned}$$

6.2 Cost of algorithm `mcr`

Table 3 lists the cost of computing matrix by vector products, for the matrices used in algorithm `mcr` which were not introduced for algorithm `cr`. As already said, this cost does not include the cost for the transformation of the vectors of the d-description of the matrix itself and the cost of transforming the input vector.

matrix	cost
$U^{(i)}$	$5\pi_n + 4\pi_m$
$V^{(i)}$	$4\pi_n + 5\pi_m$

Table 3: Cost of computing matrix by vector products for algorithm `mcr`.

The cost of the ith step of algorithm `mcr` is the same as for `cr`, except for what concerns the computation of the d-description of ${H_0^{(i)}}^{-1}$, which in algorithm `cr` is directly computed by solving four systems of order n with cost $\sigma_{n,4}(4)$, while here is computed by using SMW formula. Hence to find the cost of algorithm `mcr` we have to estimate the cost of this formula.

No computation is required to find the d-description of $U^{(i-1)}$ and of $V^{(i-1)}$. The vectors $\boldsymbol{v}_0^{(i-1)}$, $\boldsymbol{v}_0^{(i)}$, $\boldsymbol{t}^{(i-1)}$, $\boldsymbol{t}^{(i)}$, $\boldsymbol{v}_1^{(i-1)}$, $\boldsymbol{v}_1^{(i)}$, $\boldsymbol{r}^{(i-1)}$ and $\boldsymbol{r}^{(i)}$ have already been transformed.

The d-description of $P^{(i-1)}$ is computed as follows. The resulting vectors are not transformed, since Levinson algorithm does not require them.

For $\boldsymbol{p}_1^{(i-1)} = -V^{(i-1)}{H_0^{(i-1)}}^{-1}\boldsymbol{v}_0^{(i)}$, one product by ${H_0^{(i-1)}}^{-1}$ and one product by $V^{(i-1)}$ are required. For the first product the input vector has already been transformed. The costs are $5\pi_n$ and $5\pi_n + 5\pi_m$.

For $\boldsymbol{p}_3^{(i-1)} = -V^{(i-1)}\widetilde{\boldsymbol{v}}_0^{(i-1)}$, the product by $V^{(i-1)}$ of a vector already transformed costs $4\pi_n + 5\pi_m$.

For $\boldsymbol{p}_4^{(i-1)} = \overline{F}_{2m}\boldsymbol{r}^{(i-1)} - V^{(i-1)}\widetilde{\boldsymbol{r}}^{(i-1)}$, as above, the cost is $4\pi_n + 5\pi_m$.

For $\boldsymbol{p}_5^{(i-1)} = -V^{(i-1)}{H_0^{(i-1)}}^{-1}H_1^{(i-1)}\widetilde{\boldsymbol{r}}^{(i-1)}$. The vector $z_1 = H_1^{(i-1)}\widetilde{\boldsymbol{r}}^{(i-1)}$ has already been computed in `cr`, hence only two products are required: one by ${H_0^{(i-1)}}^{-1}$ and one by $V^{(i-1)}$. No input vector has yet been transformed. The costs are $6\pi_n$ and $5\pi_n + 5\pi_m$.

For $\boldsymbol{p}_7^{(i-1)} = \overline{F}_{2m}\big(\boldsymbol{v}_1^{(i)} - \boldsymbol{v}_1^{(i-1)}\big) + V^{(i-1)}\widetilde{\boldsymbol{v}}_1^{(i-1)}$, the product by $V^{(i-1)}$ of a vector already transformed costs $4\pi_n + 5\pi_m$.

For $P^{(i-1)}\overline{F}_{2m}\overline{\boldsymbol{e}}_1 = \overline{F}_{2m}\overline{\boldsymbol{e}}_1 - V^{(i-1)}{H_0^{(i-1)}}^{-1}\boldsymbol{e}_{n-m+1}$, one product by ${H_0^{(i-1)}}^{-1}$ and one product by $V^{(i-1)}$ are required. For the first product the input vector has already been transformed. The costs are $5\pi_n$ and $5\pi_n + 5\pi_m$.

Hence the cost required for computing the d-vectors of $P^{(i-1)}$ is $c_P = 43\pi_n + 30\pi_m$.

To find the representation of ${H_0^{(i)}}^{-1}$, we compute by SMW formula the expression

$$ {H_0^{(i)}}^{-1}\boldsymbol{z} = {H_0^{(i-1)}}^{-1}\boldsymbol{z} + {H_0^{(i-1)}}^{-1}U^{(i-1)}{P^{(i-1)}}^{-1}V^{(i-1)}{H_0^{(i-1)}}^{-1}\boldsymbol{z} \qquad (29) $$

for selected vectors $\boldsymbol{z}$. We compute first $\boldsymbol{z}_1 = {H_0^{(i-1)}}^{-1}\boldsymbol{z}$. If $\boldsymbol{z}$ has already been transformed, it costs $5\pi_n$. Then we compute $\boldsymbol{z}_2 = V^{(i-1)}\boldsymbol{z}_1$ as previously explained, with cost $5\pi_n + 5\pi_m$. The vector $\boldsymbol{z}_3 = {P^{(i-1)}}^{-1}\boldsymbol{z}_2$ is obtained by solving a system of order $2m$ applying Levinson algorithm. Now the product $U^{(i-1)}\boldsymbol{z}_3$ is computed as follows

$$ \boldsymbol{z}_4 = U^{(i-1)}\boldsymbol{z}_3 = \left[\ \underline{F} \mid U_1^{(i-1)}\ \right]\boldsymbol{z}_3 = \underline{F}\overline{F}^T_{2m}\boldsymbol{z}_3 + U_1^{(i-1)}\underline{F}^T_{2m}\boldsymbol{z}_3. $$

The product $U_1^{(i-1)}\underline{F}^T_{2m}\boldsymbol{z}_3$ has cost $5\pi_n + 5\pi_m$. Finally vector $\boldsymbol{z}_4$ is multiplied by ${H_0^{(i-1)}}^{-1}$ with cost $6\pi_n$. The total cost for the computation of (29) is $c_t = 21\pi_n + 10\pi_m$, plus the cost of Levinson algorithm.

To obtain the d-description of ${H_0^{(i)}}^{-1}$ expression (29) is computed for the vectors $\boldsymbol{z} = \underline{F}\boldsymbol{r}^{(i)}$, $\boldsymbol{z} = \boldsymbol{v}_0^{(i)}$, $\boldsymbol{z} = \underline{F}\boldsymbol{v}_1^{(i)}$ and $\boldsymbol{z} = \boldsymbol{e}_1$. The vectors $\boldsymbol{z} = \underline{F}\boldsymbol{r}^{(i)}$ and $\boldsymbol{z} = \underline{F}\boldsymbol{v}_1^{(i)}$ are transformed with cost $2\pi_n$. Then the cost of this step is

$$ c_S = 4c_t + 2\pi_n + \sigma_{2m,9}(4) = 86\pi_n + 40\pi_m + \sigma_{2m,9}(4) = 86\pi_n + 40\pi_m + 108m^2. $$

The sum $c_S + c_P = 129\pi_n + 70\pi_m + 108m^2$ is the quantity we have to substitute to $\sigma_{n,4}(4)$ into formula (26) to get the cost of algorithm `mcr`, which turns out to be

$$ c_{\texttt{mcr}} = (145\pi_n + 84\pi_m + 108m^2)\log N + (24\pi_n + 12\pi_m)N. \qquad (30) $$

7 Algorithm `st`

The last method, which will be called `st`, is based on a doubling algorithm devised by Stewart for solving block Hessenberg systems. It has been modified in [2] for matrices with Toeplitz structure, in order to lower its computational cost. Our goal is to show that a further decrease of the cost can be achieved with band matrices. The basic idea is the following.

Let $N = 2^p$, with $p \geq 2$. For $i = 0, \ldots, p$, let $n_i = 2^i n$ and denote by A_i the $n_i \times n_i$ leading principal submatrix of A. We consider the sequence

$$ A_0 = B_0, \qquad A_{i+1} = \begin{bmatrix} A_i & R_i \\ U_i & A_i \end{bmatrix}, \qquad A_p = A, $$

with

$$ U_i = \overline{E}B_1\underline{E}^T,\ R_i = \underline{E}B_{-1}\overline{E}^T = \underline{E}\underline{F}R\overline{F}^T\overline{E}^T. $$

At the ith level of recursion we put into relation the solution of the linear system

$$A_{i+1}\boldsymbol{z} = \boldsymbol{t}, \quad \text{with} \quad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}' \\ \boldsymbol{z}'' \end{bmatrix}, \quad \boldsymbol{t} = \begin{bmatrix} \boldsymbol{t}' \\ \boldsymbol{t}'' \end{bmatrix}, \tag{31}$$

(where $'$ and $''$ indicate the first half and the second half (resp.) of the vector) with the solution of the system

$$\begin{bmatrix} A_i & O \\ U_i & A_i \end{bmatrix} \begin{bmatrix} \boldsymbol{y}' \\ \boldsymbol{y}'' \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}' \\ \boldsymbol{t}'' \end{bmatrix}.$$

We have

$$\begin{cases} \boldsymbol{y}' = A_i^{-1}\boldsymbol{t}' \\ \boldsymbol{y}'' = A_i^{-1}\big(\boldsymbol{t}'' - U_i\,\boldsymbol{y}'\big) \end{cases} \tag{32}$$

and

$$\begin{cases} \boldsymbol{z}' = \boldsymbol{y}' - A_i^{-1}R_i\,\boldsymbol{z}'' \\ \boldsymbol{z}'' = T_i^{-1}\boldsymbol{y}'' \end{cases} \tag{33}$$

with

$$T_i = I - A_i^{-1}U_i\,A_i^{-1}R_i = I - G_i\underline{E}^T H_i\overline{F}^T\overline{E}^T,$$

where

$$G_i = A_i^{-1}\overline{E}B_1 \quad \text{and} \quad H_i = A_i^{-1}\underline{E}\underline{F}R.$$

Due to the band structure of matrix A_{i+1}, matrix $G_i\underline{E}^T H_i\overline{F}^T\overline{E}^T$ is a low rank (exactly m) correction to the n_i dimensional identity. We can exploit this fact by using the SMW formula. So we have

$$T_i^{-1} = I + G_i\underline{E}^T H_i P_i^{-1}\overline{F}^T\overline{E}^T, \tag{34}$$

where matrix

$$P_i = I - \overline{F}^T\overline{E}^T G_i\underline{E}^T H_i$$

has size m. In this way the inverse of the $n_i \times n_i$ matrix T_i is obtained by inverting the $m \times m$ matrix P_i. Replacing (34) into (33) we see that the solution of system (31) of size n_{i+1} can be expressed as an additive correction of the two solutions (32) of size n_i in the following way

$$\begin{bmatrix} \boldsymbol{z}' \\ \boldsymbol{z}'' \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}' \\ \boldsymbol{y}'' \end{bmatrix} + \begin{bmatrix} -H_iP_i^{-1}\overline{F}^T\overline{E}^T\boldsymbol{y}'' \\ G_i\underline{E}^T H_iP_i^{-1}\overline{F}^T\overline{E}^T\boldsymbol{y}'' \end{bmatrix}. \tag{35}$$

The method `st` is based on the recursive application of this procedure. At the highest level the solution of the system $A_p\boldsymbol{x} = \boldsymbol{b}$ leads to solving two systems with half sized matrix A_{p-1} and suitable right hand-sides. Each of these two systems recursively leads to other two systems, and so on, until N elementary sized systems, all having matrix B_0, are obtained. They are solved and the successive application of (35) allows the reconstruction of $\boldsymbol{x}$.

To turn this outlined procedure into a really efficient method, we need a compact notation of the matrices which appear in (35) and formulas to compute them recursively. Their computation is the object of the transformation phase.

7.1 Transformation phase

We consider the upper and lower blocks (called *angle* blocks) of the matrices G_i and H_i

$$\overline{G}_i = \overline{F}^T\overline{E}^T G_i, \quad \underline{G}_i = \underline{E}^T G_i, \quad \overline{H}_i = \overline{F}^T\overline{E}^T H_i, \quad \underline{H}_i = \underline{E}^T H_i.$$

The sizes are $m \times n$ for $\overline{G}_i$, $n \times n$ for $\underline{G}_i$, $m \times m$ for $\overline{H}_i$ and $n \times m$ for $\underline{H}_i$. For the index $i = 0$ we have

$$\begin{aligned} G_0 &= \underline{G}_0 = B_0^{-1}B_1, \quad \overline{G}_0 = \overline{F}^T\underline{G}_0, \\ H_0 &= \underline{H}_0 = B_0^{-1}\underline{F}R, \quad \overline{H}_0 = \overline{F}^T\underline{H}_0. \end{aligned} \tag{36}$$

Then

$$A_{i+1}^{-1} = \left[\begin{array}{c|c} (I + A_i^{-1}R_iT_i^{-1}A_i^{-1}U_i)A_i^{-1} & -A_i^{-1}R_iT_i^{-1}A_i^{-1} \\ \hline -T_i^{-1}A_i^{-1}U_iA_i^{-1} & T_i^{-1}A_i^{-1} \end{array}\right]$$

$$= \left[\begin{array}{c|c} (I + H_iP_i^{-1}\overline{G}_i\underline{E}^T)A_i^{-1} & -H_iP_i^{-1}\overline{F}^T\overline{E}^T A_i^{-1} \\ \hline -G_i(I + \underline{H}_iP_i^{-1}\overline{G}_i)\underline{E}^T A_i^{-1} & (I + G_i\underline{H}_iP_i^{-1}\overline{F}^T\overline{E}^T)A_i^{-1} \end{array}\right],$$

where

$$P_i = I - \overline{F}^T\overline{E}^T G_i\underline{H}_i = I - \overline{G}_i\,\underline{H}_i. \tag{37}$$

We have now the recursive relations for G_i and H_i

$$\begin{aligned} G_{i+1} &= A_{i+1}^{-1}\begin{bmatrix} \overline{E} \\ O \end{bmatrix} B_1 = \begin{bmatrix} G_i + H_iP_i^{-1}\overline{G}_i\underline{G}_i \\ -G_i(\underline{G}_i + \underline{H}_iP_i^{-1}\overline{G}_i\underline{G}_i) \end{bmatrix}, \\ H_{i+1} &= A_{i+1}^{-1}\begin{bmatrix} O \\ \underline{E} \end{bmatrix} \underline{F}\,R = \begin{bmatrix} -H_iP_i^{-1}\overline{H}_i \\ G_i\underline{H}_iP_i^{-1}\overline{H}_i + H_i \end{bmatrix}. \end{aligned} \tag{38}$$

From (38) we get the recursive relations for the angle blocks

$$\begin{aligned} \overline{G}_{i+1} &= \overline{G}_i + \overline{H}_iP_i^{-1}\overline{G}_i\underline{G}_i, & \overline{H}_{i+1} &= -\overline{H}_iP_i^{-1}\overline{H}_i, \\ \underline{G}_{i+1} &= -\underline{G}_i(\underline{G}_i + \underline{H}_iP_i^{-1}\overline{G}_i\underline{G}_i), & \underline{H}_{i+1} &= \underline{G}_i\underline{H}_iP_i^{-1}\overline{H}_i + \underline{H}_i. \end{aligned} \tag{39}$$

In the substitution phase we use also the matrices

$$M_i = \underline{G}_i\underline{H}_iP_i^{-1}, \quad K_i = P_i^{-1}\overline{H}_i \quad \text{and} \quad L_i = P_i^{-1}\overline{G}_i\underline{G}_i. \tag{40}$$

In the following the "over" and "under" line notation is used also for vectors. Given a vector $\boldsymbol{v}$ of any size, we denote by $\overline{\boldsymbol{v}} = \overline{F}^T\overline{E}^T\boldsymbol{v}$ and $\underline{\boldsymbol{v}} = \underline{E}^T\boldsymbol{v}$ the subvectors of the first m components and of the last n components of $\boldsymbol{v}$ (resp.).

7.2 Substitution phase

For $i = 0, \ldots, p$ and $j = 1, \ldots, 2^{p-i}$ let $\boldsymbol{b}_j^{(i)}$ be the jth subvector of size n_i of $\boldsymbol{b}$. Then

$$\boldsymbol{b}_j^{(i+1)'} = \boldsymbol{b}_{2j-1}^{(i)}, \qquad \boldsymbol{b}_j^{(i+1)''} = \boldsymbol{b}_{2j}^{(i)}.$$

We define now recursively a sequence of vectors $\boldsymbol{y}_j^{(i)}$ of size n_i by means of the systems

$$A_i \boldsymbol{y}_j^{(i)} = \boldsymbol{t}_j^{(i)}, \quad \text{for} \quad i = p, \ldots, 0, \quad \text{and} \quad j = 1, \ldots, 2^{p-i}, \tag{41}$$

where $\boldsymbol{t}_1^{(p)} = \boldsymbol{b}$ and

$$\boldsymbol{t}_{2j-1}^{(i)} = \boldsymbol{t}_j^{(i+1)'}, \qquad \boldsymbol{t}_{2j}^{(i)} = \boldsymbol{t}_j^{(i+1)''} - U_i A_i^{-1} \boldsymbol{t}_j^{(i+1)'}. \tag{42}$$

Hence

$$\boldsymbol{y}_{2j-1}^{(i)} = A_i^{-1} \boldsymbol{t}_{2j-1}^{(i)} = A_i^{-1} \boldsymbol{t}_j^{(i+1)'}$$

and

$$\boldsymbol{y}_{2j}^{(i)} = A_i^{-1} \boldsymbol{t}_{2j}^{(i)} = A_i^{-1} \big(\boldsymbol{t}_j^{(i+1)''} - U_i \, \boldsymbol{y}_{2j-1}^{(i)} \big).$$

From (35) we have

$$\boldsymbol{y}_j^{(i+1)} = A_{i+1}^{-1} \begin{bmatrix} \boldsymbol{t}_j^{(i+1)'} \\ \boldsymbol{t}_j^{(i+1)''} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_{2j-1}^{(i)} - H_i \, P_i^{-1} \, \overline{\boldsymbol{y}}_{2j}^{(i)} \\ \boldsymbol{y}_{2j}^{(i)} + G_i \underline{H}_i \, P_i^{-1} \, \overline{\boldsymbol{y}}_{2j}^{(i)} \end{bmatrix}.$$

Then the vectors of the first m components and of the last n components of $\boldsymbol{y}_j^{(i+1)}$ are given by

$$\overline{\boldsymbol{y}}_j^{(i+1)} = \overline{\boldsymbol{y}}_{2j-1}^{(i)} - \overline{H}_i \, P_i^{-1} \, \overline{\boldsymbol{y}}_{2j}^{(i)}, \quad \underline{\boldsymbol{y}}_j^{(i+1)} = \underline{\boldsymbol{y}}_{2j}^{(i)} + \underline{G_i} \underline{H}_i \, P_i^{-1} \, \overline{\boldsymbol{y}}_{2j}^{(i)}. \tag{43}$$

The vectors $\overline{\boldsymbol{y}}_j^{(i+1)}$ and $\underline{\boldsymbol{y}}_j^{(i+1)}$ are constructed in the forward substitution phase. They can be computed recursively, but we prefer to give the equivalent iterative version in the next sketchy implementation.

Due to the structure of U_i, the vector $\boldsymbol{t}_j^{(i)}$ differs from $\boldsymbol{b}_j^{(i)}$ only in the first n components. More precisely, we show recursively that

$$\begin{aligned} &\boldsymbol{t}_1^{(i)} = \boldsymbol{b}_1^{(i)}, \qquad \boldsymbol{t}_{2j-1}^{(i)} = \boldsymbol{b}_{2j-1}^{(i)} - \overline{E} B_1 \underline{\boldsymbol{y}}_k^{(i+h)}, \quad \text{for} \quad j > 1, \\ &\boldsymbol{t}_{2j}^{(i)} = \boldsymbol{b}_{2j}^{(i)} - \overline{E} B_1 \underline{\boldsymbol{y}}_{2j-1}^{(i)}, \end{aligned} \tag{44}$$

where h and k are defined by the relation

$$2j = 2 + 2^h \cdot k, \quad \text{with } k \text{ odd}. \tag{45}$$

In fact, in the even case we have

$$\boldsymbol{t}_{2j}^{(i)} = \boldsymbol{b}_j^{(i+1)''} - U_i A_i^{-1} \boldsymbol{t}_{2j-1}^{(i)} = \boldsymbol{b}_{2j}^{(i)} - U_i \, \boldsymbol{y}_{2j-1}^{(i)} = \boldsymbol{b}_{2j}^{(i)} - \overline{E} B_1 \, \underline{\boldsymbol{y}}_{2j-1}^{(i)}.$$

In the odd case we have

$$\boldsymbol{t}_1^{(i)} = \boldsymbol{t}_1^{(i+1)'} = \boldsymbol{b}_1^{(i+1)'} = \boldsymbol{b}_1^{(i)}$$

and, if j is even

$$\boldsymbol{t}_{2j-1}^{(i)} = \boldsymbol{t}_j^{(i+1)'} = \boldsymbol{b}_j^{(i+1)'} - \overline{E}B_1\,\underline{\boldsymbol{y}}_{j-1}^{(i+1)} = \boldsymbol{b}_{2j-1}^{(i)} - \overline{E}B_1\,\underline{\boldsymbol{y}}_k^{(i+h)},$$

with $k = j-1$, $h = 1$ since $2j-1 = 1+2(j-1)$, and if j is odd

$$\boldsymbol{t}_{2j-1}^{(i)} = \boldsymbol{t}_j^{(i+1)'} = \boldsymbol{b}_j^{(i+1)'} - \overline{E}B_1\,\underline{\boldsymbol{y}}_k^{(i+1+h)} = \boldsymbol{b}_{2j-1}^{(i)} - \overline{E}B_1\,\underline{\boldsymbol{y}}_k^{(i+1+h)},$$

with $j = 1 + 2^h \cdot k$, k odd. Hence $2j-1 = 1 + 2^{h+1}\cdot k$.

In the backward substitution phase the solution $\boldsymbol{x}$ is reconstructed from the vectors $\overline{\boldsymbol{y}}_{2j}^{(i)}$ and $\underline{\boldsymbol{y}}_{2j}^{(i)}$. For $i = 0,\ldots,p$ and $j = 1,\ldots,2^{p-i}$ let $\boldsymbol{x}_j^{(i)}$ be the jth subvector of size n_i of $\boldsymbol{x}$. The computation starts with

$$\boldsymbol{x}_1^{(p)} = \boldsymbol{x} = A^{-1}\boldsymbol{b} = A_p^{-1}\boldsymbol{t}_1^{(p)} = \boldsymbol{y}_1^{(p)},$$

i.e.

$$\overline{\boldsymbol{x}}_1^{(p)} = \overline{\boldsymbol{y}}_1^{(p)} \quad \text{and} \quad \underline{\boldsymbol{x}}_1^{(p)} = \underline{\boldsymbol{y}}_1^{(p)},$$

and proceeds with $\overline{\boldsymbol{x}}_j^{(i)}$ and $\underline{\boldsymbol{x}}_j^{(i)}$ for $i = p-1,\ldots,0$ and $j = 1,\ldots,2^{p-i}$. As a matter of fact, only $\underline{\boldsymbol{x}}_j^{(0)}$ are of interest at the last recursion level.

The even components $\underline{\boldsymbol{x}}_{2j}^{(i)}$ and the odd components $\overline{\boldsymbol{x}}_{2j-1}^{(i)}$ are immediately given by

$$\underline{\boldsymbol{x}}_{2j}^{(i)} = \underline{\boldsymbol{x}}_j^{(i+1)}, \qquad \overline{\boldsymbol{x}}_{2j-1}^{(i)} = \overline{\boldsymbol{x}}_j^{(i+1)}, \tag{46}$$

but of course we need also a rule for the other components. We have also

$$\underline{\boldsymbol{x}}_{2j-2}^{(i)} = \underline{\boldsymbol{x}}_k^{(i+h)}, \tag{47}$$

where h and k are defined in (45).

From the block tridiagonal structure of A we have

$$U_i\,\boldsymbol{x}_{j-1}^{(i)} + A_i\,\boldsymbol{x}_j^{(i)} + R_i\,\boldsymbol{x}_{j+1}^{(i)} = \boldsymbol{b}_j^{(i)}, \quad \text{for} \quad j = 1,\ldots,J, \quad J = 2^{p-i},$$

with $\boldsymbol{x}_0^{(i)} = \boldsymbol{x}_{J+1}^{(i)} = \boldsymbol{0}$. Then for the odd rows, except the first one, we have

$$A_i\,\boldsymbol{x}_{2j-1}^{(i)} = \boldsymbol{b}_{2j-1}^{(i)} - \overline{E}B_1\,\underline{\boldsymbol{x}}_{2j-2}^{(i)} - \underline{E}\,\underline{F}R\,\overline{\boldsymbol{x}}_{2j}^{(i)},$$

and from (47)

$$A_i\,\boldsymbol{x}_{2j-1}^{(i)} = \boldsymbol{b}_{2j-1}^{(i)} - \overline{E}B_1\,\underline{\boldsymbol{x}}_k^{(i+h)} - \underline{E}\,\underline{F}R\,\overline{\boldsymbol{x}}_{2j}^{(i)}.$$

Replacing $\boldsymbol{b}^{(i)}_{2j-1}$ from (44) and using (41) we have

$$A_i\left(\boldsymbol{x}^{(i)}_{2j-1}-\boldsymbol{y}^{(i)}_{2j-1}\right) = -\overline{E}B_1\left(\underline{\boldsymbol{x}}^{(i+h)}_k - \underline{\boldsymbol{y}}^{(i+h)}_k\right) - \underline{E}\,\underline{F}R\,\overline{\boldsymbol{x}}^{(i)}_{2j},$$

hence

$$\underline{\boldsymbol{x}}^{(i)}_{2j-1} = \underline{\boldsymbol{y}}^{(i)}_{2j-1} - \underline{G}_i\left(\underline{\boldsymbol{x}}^{(i+h)}_k - \underline{\boldsymbol{y}}^{(i+h)}_k\right) - \underline{H}_i\,\overline{\boldsymbol{x}}^{(i)}_{2j}. \tag{48}$$

Proceeding in a similar way we get

$$\overline{\boldsymbol{x}}^{(i)}_{2j} = \overline{\boldsymbol{y}}^{(i)}_{2j} - \overline{G}_i\left(\underline{\boldsymbol{x}}^{(i)}_{2j-1} - \underline{\boldsymbol{y}}^{(i)}_{2j-1}\right) - \overline{H}_i\,\overline{\boldsymbol{x}}^{(i+1)}_{j+1}. \tag{49}$$

Relations (48) and (49) form a linear system from which $\underline{\boldsymbol{x}}^{(i)}_{2j-1}$ and $\overline{\boldsymbol{x}}^{(i)}_{2j}$ can be computed. In fact, replacing $\underline{\boldsymbol{x}}^{(i)}_{2j-1}$ from (48) into (49) we get

$$P_i\overline{\boldsymbol{x}}^{(i)}_{2j} = \overline{\boldsymbol{y}}^{(i)}_{2j} + \overline{G}_i\underline{G}_i\left(\underline{\boldsymbol{x}}^{(i+h)}_k - \underline{\boldsymbol{y}}^{(i+h)}_k\right) - \overline{H}_i\,\overline{\boldsymbol{x}}^{(i+1)}_{j+1},$$

and finally

$$\overline{\boldsymbol{x}}^{(i)}_{2j} = P_i^{-1}\overline{\boldsymbol{y}}^{(i)}_{2j} + P_i^{-1}\overline{G}_i\underline{G}_i\left(\underline{\boldsymbol{x}}^{(i+h)}_k - \underline{\boldsymbol{y}}^{(i+h)}_k\right) - P_i^{-1}\overline{H}_i\,\overline{\boldsymbol{x}}^{(i+1)}_{j+1}. \tag{50}$$

Hence we compute $\overline{\boldsymbol{x}}^{(i)}_{2j}$ by applying (50) and $\underline{\boldsymbol{x}}^{(i)}_{2j-1}$ by applying (48).

Special subvectors are given by

$$\begin{aligned} &\underline{\boldsymbol{x}}^{(i)}_1 = \underline{\boldsymbol{y}}^{(i)}_1 - \underline{H}_i\overline{\boldsymbol{x}}^{(i)}_2, \qquad \overline{\boldsymbol{x}}^{(i)}_2 = \boldsymbol{z}^{(i)}_2 - P_i^{-1}\overline{H}_i\,\overline{\boldsymbol{x}}^{(i+1)}_2, \\ &\overline{\boldsymbol{x}}^{(i)}_J = P_i^{-1}\overline{\boldsymbol{y}}^{(i)}_J + P_i^{-1}\overline{G}_i\underline{G}_i\left(\underline{\boldsymbol{x}}^{(i+1)}_{J/2-1} - \underline{\boldsymbol{y}}^{(i+1)}_{J/2-1}\right). \end{aligned} \tag{51}$$

At the last recursive level the vectors $\overline{\boldsymbol{x}}^{(0)}_{2j}$ need not be computed, because $\overline{\boldsymbol{x}}^{(0)}_{2j} = \overline{F}^T\underline{\boldsymbol{x}}^{(0)}_{2j} = \overline{F}^T\underline{\boldsymbol{x}}^{(1)}_j$ and (48) can be applied directly in the following way

$$\begin{aligned} &\underline{\boldsymbol{x}}^{(0)}_1 = \underline{\boldsymbol{y}}^{(0)}_1 - \underline{H}_0\,\overline{F}^T\underline{\boldsymbol{x}}^{(1)}_1, \\ &\underline{\boldsymbol{x}}^{(0)}_{2j-1} = \underline{\boldsymbol{y}}^{(0)}_{2j-1} - \underline{G}_0\left(\underline{\boldsymbol{x}}^{(h)}_k - \underline{\boldsymbol{y}}^{(h)}_k\right) - \underline{H}_0\,\overline{F}^T\underline{\boldsymbol{x}}^{(1)}_j, \text{ for } j = 2, \ldots, N/2. \end{aligned} \tag{52}$$

7.3 Sketchy implementation

All the computations are carried out by using the d-descriptions of the involved matrices. A first function `blocks` computes the angle blocks and P_i from (37) for any i, starting with the initial positions (36) and using (39).

```
Function blocks $(A)$
  1. compute $B_0^{-1}$,
  2. compute $\overline{G}_0$, $\underline{G}_0$, $\overline{H}_0$ and $\underline{H}_0$ from (36),
  3. for $i = 0, \ldots, p-1$
```

```
4.      compute $P_i$ from (37),
5.      compute $P_i^{-1}$,
6.      compute $M_i$, $K_i$ and $L_i$ from (40),
7.      compute $\overline{G}_{i+1}$, $\underline{G}_{i+1}$, $\overline{H}_{i+1}$ and $\underline{H}_{i+1}$ from (39).
```

The function `forward` computes iteratively the vectors needed in the last backward phase by exploiting (43). It makes use of a function `takem` $(\boldsymbol{u})$ which take the first m components of a vector $\boldsymbol{u}$, and of the logical function `even` (r) which returns `True` if r is even.

Function `forward` $(\boldsymbol{b})$

```
1. $\boldsymbol{c} = \boldsymbol{0}$,
2. for $k = 1, \ldots, N$
3.      $\boldsymbol{c} = \underline{\boldsymbol{y}}_k^{(0)} = B_0^{-1}(\boldsymbol{b}_k^{(0)} - B_1\boldsymbol{c})$,    $\overline{\boldsymbol{y}}_k^{(0)} = $ takem $(\boldsymbol{c})$,
4.      if even $(k)$ then
5.          $j = k$,    $i = 0$,
6.          while even$(j)$
7.              $j = j/2$,
8.              $\boldsymbol{z}_{2j}^{(i)} = P_i^{-1}\overline{\boldsymbol{y}}_{2j}^{(i)}$,
9.              $\overline{\boldsymbol{y}}_j^{(i+1)} = \overline{\boldsymbol{y}}_{2j-1}^{(i)} - \overline{H}_i\boldsymbol{z}_{2j}^{(i)}$,
10.             $\boldsymbol{c} = \underline{\boldsymbol{y}}_j^{(i+1)} = \underline{\boldsymbol{y}}_{2j}^{(i)} + M_i\overline{\boldsymbol{y}}_{2j}^{(i)}$,
11.             $i = i + 1$.
```

Now function `st` implements the whole method.

Function `st` $(A, N, \boldsymbol{b})$

```
1. call blocks $(A)$,
2. call forward$(\boldsymbol{b})$ to compute the vectors $\overline{\boldsymbol{y}}_j^{(i)}$, $\underline{\boldsymbol{y}}_j^{(i)}$ and $\boldsymbol{z}_{2j}^{(i)}$,
3. let $\overline{\boldsymbol{x}}_1^{(p)} = \overline{\boldsymbol{y}}_1^{(p)}$, $\underline{\boldsymbol{x}}_1^{(p)} = \underline{\boldsymbol{y}}_1^{(p)}$,
4. for $i = p-1, \ldots, 1$,    for $j = 1, \ldots, 2^{p-i-1}$,
5.      set $\underline{\boldsymbol{x}}_{2j}^{(i)} = \underline{\boldsymbol{x}}_j^{(i+1)}$ and $\overline{\boldsymbol{x}}_{2j-1}^{(i)} = \overline{\boldsymbol{x}}_j^{(i+1)}$,
6.      compute $\overline{\boldsymbol{x}}_{2j}^{(i)}$ and $\underline{\boldsymbol{x}}_{2j-1}^{(i)}$ by applying (50), (51) and (48),
7. for $j = N/2, \ldots, 1$ compute $\underline{\boldsymbol{x}}_{2j}^{(0)}$ and $\underline{\boldsymbol{x}}_{2j-1}^{(0)}$ by applying (46) and (52).
```

7.4 Useful vectors

To give the d-description of the matrices of algorithm `st` we must introduce some new vectors and give some formulas connecting them.

$\widetilde{\boldsymbol{e}}_j$ is the jth canonical vector of size n_i,

$\boldsymbol{r}^T = \overline{\boldsymbol{e}}_n^T R$ is the last row of R,

$\overline{\boldsymbol{x}}_i = \overline{H}_i \overline{\boldsymbol{e}}_1$ is the first column of $\overline{H}_i$,

$\underline{\boldsymbol{x}}_i = \underline{H}_i \overline{\boldsymbol{e}}_1$ is the first column of $\underline{H}_i$,

$\overline{\boldsymbol{u}}_i^T = \boldsymbol{r}^T \overline{G}_i$ is the last row of $R\overline{G}_i$,

$\underline{\boldsymbol{u}}_i^T = \boldsymbol{e}_n^T \underline{G}_i$ is the last row of $\underline{G}_i$,

$\overline{\boldsymbol{v}}_i^T = \boldsymbol{r}^T \overline{H}_i$ is the last row of $R\overline{H}_i$,

$\underline{\boldsymbol{v}}_i^T = \boldsymbol{e}_n^T \underline{H}_i$ is the last row of $\underline{H}_i$,

$\overline{\boldsymbol{\ell}}_i = \overline{F}^T \overline{E}^T A_i^{-1} \widetilde{\boldsymbol{e}}_1$ is the vector of the first m entries of the first column of A_i^{-1},

$\underline{\boldsymbol{\ell}}_i = \underline{E}^T A_i^{-1} \widetilde{\boldsymbol{e}}_1$ is the vector of the last n entries of the first column of A_i^{-1},

$\boldsymbol{g}_i^T = \widetilde{\boldsymbol{e}}_{n_i - n}^T G_i$ is the $(n_i - n)$th row of G_i for $i \geq 1$,

$\boldsymbol{h}_i^T = \widetilde{\boldsymbol{e}}_{n_i - n}^T H_i$ is the $(n_i - n)$th row of H_i for $i \geq 1$,

$\overline{\boldsymbol{w}}_i = J\boldsymbol{f}_0 - \overline{\boldsymbol{u}}_i$.

From (39) we get

$$\overline{\boldsymbol{x}}_{i+1} = -\overline{H}_i P_i^{-1} \overline{H}_i \overline{\boldsymbol{e}}_1 = -\overline{H}_i P_i^{-1} \overline{\boldsymbol{x}}_i, \qquad \overline{\boldsymbol{x}}_0 = \overline{F}^T \widehat{\boldsymbol{f}}_1,$$

$$\underline{\boldsymbol{x}}_{i+1} = (\underline{H}_i + \underline{G}_i \underline{H}_i P_i^{-1} \overline{H}_i) \overline{\boldsymbol{e}}_1 = \underline{\boldsymbol{x}}_i + \underline{G}_i \underline{H}_i P_i^{-1} \overline{\boldsymbol{x}}_i, \qquad \underline{\boldsymbol{x}}_0 = \widehat{\boldsymbol{f}}_1,$$

$$\overline{\boldsymbol{u}}_{i+1}^T = \overline{\boldsymbol{u}}_i^T + \boldsymbol{r}^T \overline{H}_i P_i^{-1} \overline{G}_i \underline{G}_i = \overline{\boldsymbol{u}}_i^T + \overline{\boldsymbol{v}}_i^T P_i^{-1} \overline{G}_i \underline{G}_i, \qquad \overline{\boldsymbol{u}}_0^T = \widehat{\boldsymbol{f}}_1^T J B_1,$$

$$\underline{\boldsymbol{u}}_{i+1}^T = -\boldsymbol{e}_n^T \big(\underline{G}_i^2 + \underline{G}_i \underline{H}_i P_i^{-1} \overline{G}_i \underline{G}_i \big) = -\underline{\boldsymbol{u}}_i^T \underline{G}_i - \underline{\boldsymbol{u}}_i^T \underline{H}_i P_i^{-1} \overline{G}_i \underline{G}_i, \qquad \underline{\boldsymbol{u}}_0^T = \widehat{\boldsymbol{f}}_2^T J B_1,$$

$$\overline{\boldsymbol{v}}_{i+1}^T = -\boldsymbol{r}^T \overline{H}_i P_i^{-1} \overline{H}_i = -\overline{\boldsymbol{v}}_i^T P_i^{-1} \overline{H}_i, \qquad \overline{\boldsymbol{v}}_0^T = \widehat{\boldsymbol{f}}_1^T J \underline{F} R,$$

$$\underline{\boldsymbol{v}}_{i+1}^T = \underline{\boldsymbol{v}}_i^T + \boldsymbol{e}_n^T \underline{G}_i \underline{H}_i P_i^{-1} \overline{H}_i = \underline{\boldsymbol{v}}_i^T + \underline{\boldsymbol{u}}_i^T \underline{H}_i P_i^{-1} \overline{H}_i, \qquad \underline{\boldsymbol{v}}_0^T = \widehat{\boldsymbol{f}}_2^T J \underline{F} R,$$

$$\underline{\boldsymbol{\ell}}_{i+1} = -\underline{G}_i (I + \underline{H}_i P_i^{-1} \overline{G}_i) \underline{\boldsymbol{\ell}}_i, \qquad \underline{\boldsymbol{\ell}}_0 = \widehat{\boldsymbol{f}}_2,$$

$$\overline{\boldsymbol{\ell}}_{i+1} = \overline{\boldsymbol{\ell}}_i + \overline{H}_i P_i^{-1} \overline{G}_i \underline{\boldsymbol{\ell}}_i, \qquad \overline{\boldsymbol{\ell}}_0 = \overline{F}^T \underline{\boldsymbol{\ell}}_0,$$

$$\boldsymbol{g}_{i+1}^T = -\boldsymbol{g}_i^T \big(\underline{G}_i + \underline{H}_i P_i^{-1} \overline{G}_i \underline{G}_i \big), \qquad \boldsymbol{g}_0^T = -\boldsymbol{e}_n^T$$

$$\boldsymbol{h}_{i+1}^T = \boldsymbol{h}_i^T + \boldsymbol{g}_i^T \underline{H}_i P_i^{-1} \overline{H}_i, \qquad \boldsymbol{h}_0^T = \boldsymbol{0}^T.$$

From $B_1 \boldsymbol{e}_1 = a_n \boldsymbol{e}_1$ it follows that

$$\overline{G}_i \boldsymbol{e}_1 = a_n \overline{\boldsymbol{\ell}}_i, \qquad \underline{G}_i \boldsymbol{e}_1 = a_n \underline{\boldsymbol{\ell}}_i.$$

From

$$\boldsymbol{f}_0^T J \underline{E}^T + \widetilde{e}_{n_i-n}^T a_n = \left[0, \ldots, 0, a_n, a_{n-1}, \ldots, a_0\right] = \widetilde{e}_{n_i}^T A_i,$$

it follows that

$$(\boldsymbol{f}_0^T J \underline{E}^T + \widetilde{\boldsymbol{e}}_{n_i-n}^T a_n) A_i^{-1} \underline{EF} R = \widetilde{\boldsymbol{e}}_{n_i} \underline{EF} R = \boldsymbol{e}_n^T \underline{F} R = \boldsymbol{r}^T,$$

hence

$$\overline{\boldsymbol{w}}_i^T \underline{H}_i + a_n \boldsymbol{h}_i^T = (\boldsymbol{f}_0^T J - \boldsymbol{r}^T \overline{G}_i) \underline{H}_i + a_n \boldsymbol{h}_i^T = \boldsymbol{r}^T P_i,$$

and analogously

$$\boldsymbol{f}_0^T J \underline{G}_i + a_n \boldsymbol{g}_i^T = \boldsymbol{0}^T.$$

7.5 D-description of the matrices

Matrix R size $m \times m$ d-rank=0.

$$\Delta(R) = O.$$

Matrix A_i

$$\Delta(A_i) = \left(\overline{E}\boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T B_{-1} \overline{E}^T\right) - \left(\underline{E} B_{-1} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{E}^T\right).$$

Matrix $G_i = A_i^{-1} \overline{E} B_1$ size $n_i \times n$ d-rank=3

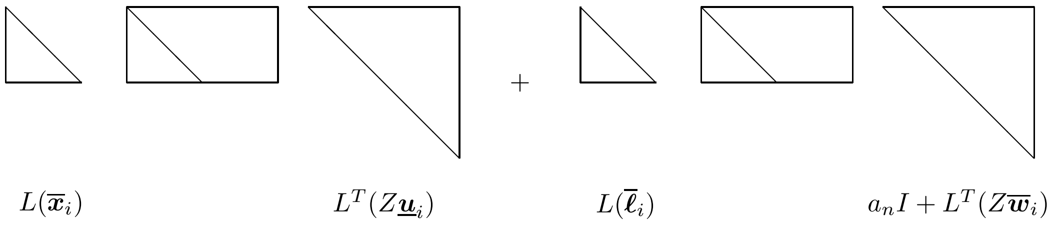
$$\begin{aligned}
\Delta(G_i) &= -A_i^{-1} \Delta(A_i) A_i^{-1} \overline{E} B_1 + A_i^{-1} \Delta(\overline{E}) B_1 + A_i^{-1} \overline{E} \Delta(B_1) \\
&= -A_i^{-1} \Big[\left(\overline{E}\boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T B_{-1} \overline{E}^T\right) - \left(\underline{E} B_{-1} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{E}^T\right)\Big] G_i - A_i^{-1} \widetilde{\boldsymbol{e}}_{n+1} \boldsymbol{e}_n^T B_1 \\
&\quad + A_i^{-1} \overline{E} \left(\boldsymbol{e}_1 \boldsymbol{e}_n^T B_0 - B_0 \boldsymbol{e}_1 \boldsymbol{e}_n^T\right) \\
&= \left(A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{E}^T G_i\right) + \left(A_i^{-1} \overline{E} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T (B_0 - B_{-1} \overline{E}^T G_i)\right) \\
&\quad - A_i^{-1} \left(a_n \widetilde{\boldsymbol{e}}_{n+1} + \overline{E} B_0 \boldsymbol{e}_1\right) \boldsymbol{e}_n^T \\
&= \left(A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{G}_i\right) + \left(A_i^{-1} \overline{E}\, \boldsymbol{e}_1\right)\left(\boldsymbol{f}_0^T J - \boldsymbol{r}^T \overline{G}_i\right) - \widetilde{\boldsymbol{e}}_1 \boldsymbol{e}_n^T,
\end{aligned}$$

since $\boldsymbol{e}_n^T B_1 = a_n \boldsymbol{e}_n^T$ and $a_n \widetilde{\boldsymbol{e}}_{n+1} + \overline{E} B_0 \boldsymbol{e}_1 = A_i \widetilde{\boldsymbol{e}}_1$.

Matrix $\overline{G}_i = \overline{F}^T \overline{E}^T G_i$ size $m \times n$ d-rank=3

$$\begin{aligned}
&\Delta(\overline{G}_i) = \overline{F}^T \overline{E}^T \Delta(G_i) \\
&= \left(\overline{F}^T \overline{E}^T A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{G}_i\right) + \left(\overline{F}^T \overline{E}^T A_i^{-1} \overline{E}\, \boldsymbol{e}_1\right)\left(\boldsymbol{f}_0^T J - \boldsymbol{r}^T \overline{G}_i\right) - \overline{F}^T \boldsymbol{e}_1 \boldsymbol{e}_n^T \\
&= \left(\overline{H}_i \overline{F}^T \boldsymbol{e}_1\right)\left(\boldsymbol{e}_n^T \underline{G}_i\right) + \overline{\boldsymbol{\ell}}_i \left(\boldsymbol{f}_0^T J - \boldsymbol{r}^T \overline{G}_i\right) - \overline{\boldsymbol{e}}_1 \boldsymbol{e}_n^T \\
&= \overline{\boldsymbol{x}}_i\, \underline{\boldsymbol{u}}_i^T + \overline{\boldsymbol{\ell}}_i\, \overline{\boldsymbol{w}}_i^T - \overline{\boldsymbol{e}}_1\, \boldsymbol{e}_n^T.
\end{aligned}$$

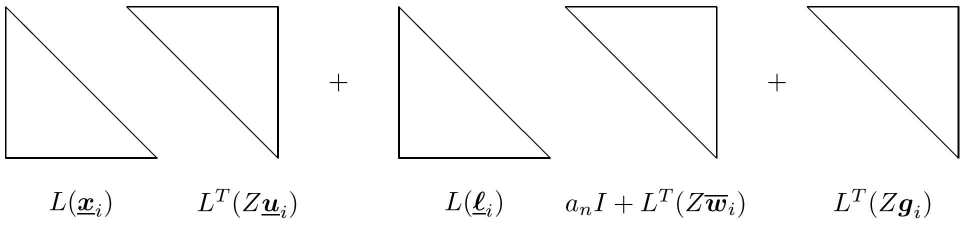
The first column of $\overline{G}_i$ is the d-vector $a_n \overline{\boldsymbol{\ell}}_i$. The d-representation of $\overline{G}_i$ has the form


$L(\overline{\boldsymbol{x}}_i)$
$L^T(Z\underline{\boldsymbol{u}}_i)$
$+$
$L(\overline{\boldsymbol{\ell}}_i)$
$a_nI + L^T(Z\overline{\boldsymbol{w}}_i)$

Matrix $\underline{G}_i = \underline{E}^T G_i$ size $n \times n$ d-rank=3 (since $\underline{E}^T \widetilde{\boldsymbol{e}}_1 = \mathbf{0}$)

$$\begin{aligned}
\Delta(\underline{G}_i) &= \underline{E}^T \Delta(G_i) + \Delta(\underline{E}^T) G_i \\
&= \big(\underline{E}^T A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\big)\big(\boldsymbol{e}_n^T \underline{G}_i\big) + \big(\underline{E}^T A_i^{-1} \overline{E} \boldsymbol{e}_1\big)\, \big(\boldsymbol{f}_0^T J - \boldsymbol{r}^T \overline{G}_i\big) \\
&\quad - \underline{E}^T \widetilde{\boldsymbol{e}}_1 \boldsymbol{e}_n^T + \big(\boldsymbol{e}_1 \widetilde{\boldsymbol{e}}^T_{n_i - n}\big) G_i \\
&= \underline{\boldsymbol{x}}_i\, \underline{\boldsymbol{u}}_i^T + \underline{\boldsymbol{\ell}}_i\, \overline{\boldsymbol{w}}_i^T + \boldsymbol{e}_1\, \boldsymbol{g}_i^T .
\end{aligned}$$

The first column of $\underline{G}_i$ is the d-vector $a_n \underline{\boldsymbol{\ell}}_i$. The d-representation of $\underline{G}_i$ has the form


$L(\underline{\boldsymbol{x}}_i)$
$L^T(Z\underline{\boldsymbol{u}}_i)$
$+$
$L(\underline{\boldsymbol{\ell}}_i)$
$a_nI + L^T(Z\overline{\boldsymbol{w}}_i)$
$+$
$L^T(Z\boldsymbol{g}_i)$

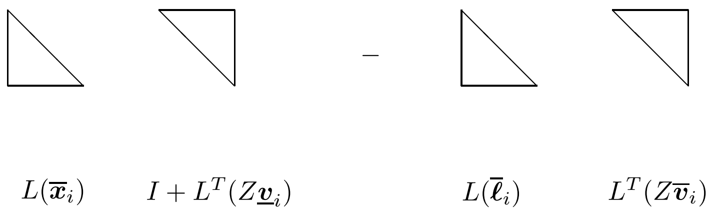
Matrix $H_i = A_i^{-1} \underline{E}\underline{F} R$ size $n_i \times m$

$$\begin{aligned}
\Delta(H_i) &= -A_i^{-1} \Delta(A_i) A_i^{-1} \underline{E}\underline{F} R \\
&= \big(-A_i^{-1} \overline{E} \boldsymbol{e}_1\big)\, \big(\boldsymbol{e}_n^T B_{-1} \overline{E}^{\,T} H_i\big) + \big(A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\big)\, \big(\boldsymbol{e}_n^T \underline{E}^T H_i\big) \\
&= \big(A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\big)\, \big(\boldsymbol{e}_n^T \underline{H}_i\big) - \big(A_i^{-1} \overline{E} \boldsymbol{e}_1\big)\, \big(\boldsymbol{r}^T \overline{H}_i\big).
\end{aligned}$$

Matrix $\overline{H}_i = \overline{F}^T \overline{E}^{\,T} H_i$ size $m \times m$ d-rank=2

$$\begin{aligned}
\Delta(\overline{H}_i) &= \overline{F}^T \overline{E}^{\,T} \Delta(H_i) \\
&= \big(\overline{F}^T \overline{E}^{\,T} A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\big)\, \big(\boldsymbol{e}_n^T \underline{H}_i\big) - \big(\overline{F}^T \overline{E}^{\,T} A_i^{-1} \overline{E} \boldsymbol{e}_1\big)\, \big(\boldsymbol{r}^T \overline{H}_i\big) \\
&= \big(\overline{H}_i \overline{F}^T \boldsymbol{e}_1\big)\, \big(\boldsymbol{e}_n^T \underline{H}_i\big) - \overline{\boldsymbol{\ell}}_i\, \big(\boldsymbol{r}^T \overline{H}_i\big) \\
&= \overline{\boldsymbol{x}}_i\, \underline{\boldsymbol{v}}_i^T - \overline{\boldsymbol{\ell}}_i\, \overline{\boldsymbol{v}}_i^T ,
\end{aligned}$$

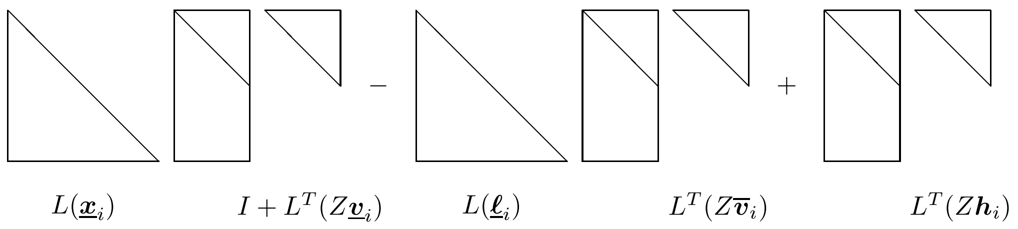
The first column of $\overline{H}_i$ is the d-vector $\overline{\boldsymbol{x}}_i$. The d-representation of $\overline{H}_i$ has the form


$-$
$L(\overline{\boldsymbol{x}}_i)$
$I + L^T(Z\underline{\boldsymbol{v}}_i)$
$L(\overline{\boldsymbol{\ell}}_i)$
$L^T(Z\overline{\boldsymbol{v}}_i)$

Matrix $\underline{H}_i = \underline{E}^T H_i$ size $n \times m$ d-rank=3 (d-rank=2 for $i = 0$)

$$
\begin{aligned}
&\Delta(\underline{H}_i) = \underline{E}^T \Delta(H_i) + \Delta(\underline{E}^T) H_i \\
&= \left(\underline{E}^T A_i^{-1} \underline{E} B_{-1} \boldsymbol{e}_1\right) \left(\boldsymbol{e}_n^T \underline{H}_i\right) - \left(\underline{E}^T A_i^{-1} \overline{E} \boldsymbol{e}_1\right) \left(\boldsymbol{r}^T \overline{H}_i\right) + \boldsymbol{e}_1 \widetilde{\boldsymbol{e}}_{n_i - n}^T H_i \\
&= \left(\underline{H}_i \overline{F}^T \boldsymbol{e}_1\right) \left(\boldsymbol{e}_n^T \underline{H}_i\right) - \underline{\boldsymbol{\ell}}_i \left(\boldsymbol{r}^T \overline{H}_i\right) + \boldsymbol{e}_1 \boldsymbol{h}_i^T \\
&= \underline{\boldsymbol{x}}_i \, \underline{\boldsymbol{v}}_i^T - \underline{\boldsymbol{\ell}}_i \, \overline{\boldsymbol{v}}_i^T + \boldsymbol{e}_1 \, \boldsymbol{h}_i^T .
\end{aligned}
$$

The first column of $\underline{H}_i$ is the d-vector $\underline{\boldsymbol{x}}_i$. The d-representation of $\underline{H}_i$ has the form


$-$
$+$
$L(\underline{\boldsymbol{x}}_i)$
$I + L^T(Z\underline{\boldsymbol{v}}_i)$
$L(\underline{\boldsymbol{\ell}}_i)$
$L^T(Z\overline{\boldsymbol{v}}_i)$
$L^T(Z\boldsymbol{h}_i)$

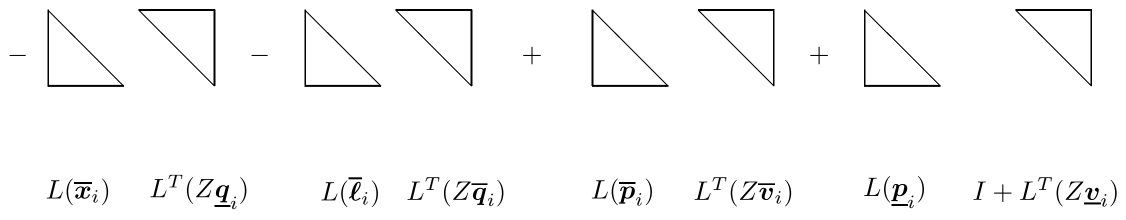
Matrix $P_i = I - \overline{G}_i \underline{H}_i$ size $m \times m$ d-rank=4

$$
\begin{aligned}
&\Delta(P_i) = \Delta(I) - \Delta(\overline{G}_i)\underline{H}_i - \overline{G}_i \Delta(\underline{H}_i) = -\Delta(\overline{G}_i)\underline{H}_i - \overline{G}_i \Delta(\underline{H}_i) \\
&= -\overline{\boldsymbol{x}}_i \, \underline{\boldsymbol{u}}_i^T \underline{H}_i - \overline{\boldsymbol{\ell}}_i \, \overline{\boldsymbol{w}}_i^T \underline{H}_i + \overline{\boldsymbol{e}}_1 \, \boldsymbol{e}_n^T \underline{H}_i \\
&\quad - \overline{G}_i \underline{\boldsymbol{x}}_i \, \underline{\boldsymbol{v}}_i^T + \overline{G}_i \underline{\boldsymbol{\ell}}_i \, \overline{\boldsymbol{v}}_i^T - \overline{G}_i \boldsymbol{e}_1 \, \boldsymbol{h}_i^T \\
&= -\overline{\boldsymbol{x}}_i \, \underline{\boldsymbol{u}}_i^T \underline{H}_i - \overline{\boldsymbol{\ell}}_i \left(\overline{\boldsymbol{w}}_i^T \underline{H}_i + a_n \boldsymbol{h}_i^T\right) + \overline{G}_i \underline{\boldsymbol{\ell}}_i \, \overline{\boldsymbol{v}}_i^T + \left(\overline{\boldsymbol{e}}_1 - \overline{G}_i \underline{\boldsymbol{x}}_i\right) \underline{\boldsymbol{v}}_i^T \\
&= -\overline{\boldsymbol{x}}_i \, \underline{\boldsymbol{q}}_i^T - \overline{\boldsymbol{\ell}}_i \, \overline{\boldsymbol{q}}_i^T + \overline{\boldsymbol{p}}_i \, \overline{\boldsymbol{v}}_i^T + \underline{\boldsymbol{p}}_i \, \underline{\boldsymbol{v}}_i^T ,
\end{aligned}
$$

where

$$
\begin{aligned}
\underline{\boldsymbol{q}}_i^T &= \underline{\boldsymbol{u}}_i^T \underline{H}_i, \qquad & \overline{\boldsymbol{q}}_i^T &= \boldsymbol{r}^T P_i = \boldsymbol{r}^T - \overline{\boldsymbol{u}}_i^T \underline{H}_i, \\
\overline{\boldsymbol{p}}_i &= \overline{G}_i \underline{\boldsymbol{\ell}}_i, \qquad & \underline{\boldsymbol{p}}_i &= P_i \, \overline{\boldsymbol{e}}_1 = \overline{\boldsymbol{e}}_1 - \overline{G}_i \underline{\boldsymbol{x}}_i.
\end{aligned}
$$

The first column of P_i is the d-vector $\underline{\boldsymbol{p}}_i$. The d-representation of P_i has the form


−
−
+
+
$L(\overline{\boldsymbol{x}}_i)$
$L^T(Z\underline{\boldsymbol{q}}_i)$
$L(\overline{\boldsymbol{\ell}}_i)$
$L^T(Z\overline{\boldsymbol{q}}_i)$
$L(\overline{\boldsymbol{p}}_i)$
$L^T(Z\overline{\boldsymbol{v}}_i)$
$L(\underline{\boldsymbol{p}}_i)$
$I + L^T(Z\underline{\boldsymbol{v}}_i)$

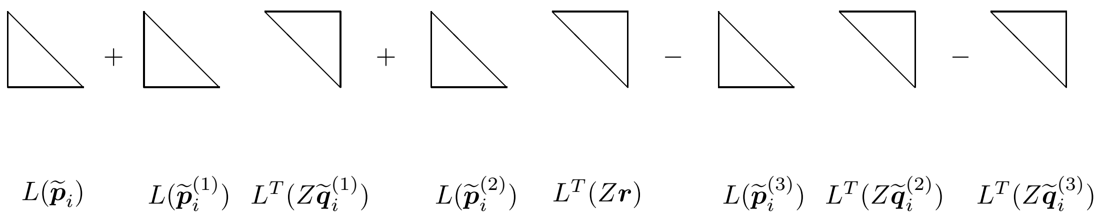
Matrix P_i^{-1} size $m \times m$ d-rank=4

$$\begin{aligned}
&\Delta(P_i^{-1}) = -P_i^{-1}\Delta(P_i^{-1})P_i^{-1} \\
&= \left(P_i^{-1}\overline{\boldsymbol{x}}_i\right)\left(\underline{\boldsymbol{q}}_i^T P_i^{-1}\right) + \left(P_i^{-1}\overline{\boldsymbol{\ell}}_i\right)\boldsymbol{r}^T - \left(P_i^{-1}\overline{\boldsymbol{p}}_i\right)\left(\overline{\boldsymbol{v}}_i^T P_i^{-1}\right) - \overline{\boldsymbol{e}}_1\left(\underline{\boldsymbol{v}}_i^T P_i^{-1}\right) \\
&= \widetilde{\boldsymbol{p}}_i^{(1)}\,\widetilde{\boldsymbol{q}}_i^{(1)T} + \widetilde{\boldsymbol{p}}_i^{(2)}\,\boldsymbol{r}^T - \widetilde{\boldsymbol{p}}_i^{(3)}\,\widetilde{\boldsymbol{q}}_i^{(2)T} - \overline{\boldsymbol{e}}_1\,\widetilde{\boldsymbol{q}}_i^{(3)T},
\end{aligned}$$

where

$$\begin{aligned}
\widetilde{\boldsymbol{p}}_i^{(1)} &= P_i^{-1}\overline{\boldsymbol{x}}_i, & \widetilde{\boldsymbol{q}}_i^{(1)T} &= \underline{\boldsymbol{q}}_i^T P_i^{-1}, \\
\widetilde{\boldsymbol{p}}_i^{(2)} &= P_i^{-1}\overline{\boldsymbol{\ell}}_i, & \widetilde{\boldsymbol{q}}_i^{(2)T} &= \overline{\boldsymbol{v}}_i^T P_i^{-1}, \\
\widetilde{\boldsymbol{p}}_i^{(3)} &= P_i^{-1}\overline{\boldsymbol{p}}_i, & \widetilde{\boldsymbol{q}}_i^{(3)T} &= \underline{\boldsymbol{v}}_i^T P_i^{-1}.
\end{aligned}$$

The first column of P_i^{-1} is $\widetilde{\boldsymbol{p}}_i = P_i^{-1}\overline{\boldsymbol{e}}_1$ and is not a d-vector. The d-representation of P_i^{-1} has the form


+
+
−
−
$L(\widetilde{\boldsymbol{p}}_i)$
$L(\widetilde{\boldsymbol{p}}_i^{(1)})$
$L^T(Z\widetilde{\boldsymbol{q}}_i^{(1)})$
$L(\widetilde{\boldsymbol{p}}_i^{(2)})$
$L^T(Z\boldsymbol{r})$
$L(\widetilde{\boldsymbol{p}}_i^{(3)})$
$L^T(Z\widetilde{\boldsymbol{q}}_i^{(2)})$
$L^T(Z\widetilde{\boldsymbol{q}}_i^{(3)})$

Matrix $M_i = \underline{G}_i\underline{H}_i P_i^{-1}$ size $n \times m$ d-rank=4

$$\begin{aligned}
&\Delta(M_i) = \Delta(\underline{G}_i)\underline{H}_i P_i^{-1} + \underline{G}_i\Delta(\underline{H}_i)P_i^{-1} + \underline{G}_i\underline{H}_i\Delta(P_i^{-1}) \\
&= \left(\underline{\boldsymbol{x}}_i\,\underline{\boldsymbol{u}}_i^T + \underline{\boldsymbol{\ell}}_i\,\overline{\boldsymbol{w}}_i^T + \boldsymbol{e}_1\,\boldsymbol{g}_i^T\right)\underline{H}_i P_i^{-1} + \underline{G}_i\left(\underline{\boldsymbol{x}}_i\,\underline{\boldsymbol{v}}_i^T - \underline{\boldsymbol{\ell}}_i\,\overline{\boldsymbol{v}}_i^T + \boldsymbol{e}_1\,\boldsymbol{h}_i^T\right)P_i^{-1} \\
&\qquad + \underline{G}_i\underline{H}_i\left(\widetilde{\boldsymbol{p}}_i^{(1)}\,\widetilde{\boldsymbol{q}}_i^{(1)T} + \widetilde{\boldsymbol{p}}_i^{(2)}\,\boldsymbol{r}^T - \widetilde{\boldsymbol{p}}_i^{(3)}\,\widetilde{\boldsymbol{q}}_i^{(2)T} - \overline{\boldsymbol{e}}_1\,\widetilde{\boldsymbol{q}}_i^{(3)T}\right).
\end{aligned}$$

Since

$$\begin{aligned}
&\underline{\boldsymbol{x}}_i\,\underline{\boldsymbol{u}}_i^T\,\underline{H}_iP_i^{-1} + \underline{G}_i\underline{H}_i\widetilde{\boldsymbol{p}}_i^{(1)}\,\widetilde{\boldsymbol{q}}_i^{(1)T} = \underline{\boldsymbol{x}}_{i+1}\,\widetilde{\boldsymbol{q}}_i^{(1)T},\\
&\underline{\boldsymbol{\ell}}_i\,\overline{\boldsymbol{w}}_i^T\,\underline{H}_iP_i^{-1} + \underline{G}_i\boldsymbol{e}_1\,\boldsymbol{h}_i^TP_i^{-1} + \underline{G}_i\underline{H}_i\,\widetilde{\boldsymbol{p}}_i^{(2)}\,\boldsymbol{r}^T = \big(\underline{\boldsymbol{\ell}}_i + \underline{G}_i\underline{H}_i\,\widetilde{\boldsymbol{p}}_i^{(2)}\big)\,\boldsymbol{r}^T,\\
&-\underline{G}_i\underline{\boldsymbol{\ell}}_i\,\overline{\boldsymbol{v}}_i^TP_i^{-1} - \underline{G}_i\underline{H}_i\,\widetilde{\boldsymbol{p}}_i^{(3)}\,\widetilde{\boldsymbol{q}}_i^{(2)T} = \underline{\boldsymbol{\ell}}_{i+1}\,\widetilde{\boldsymbol{q}}_i^{(2)T},\\
&\underline{G}_i\underline{\boldsymbol{x}}_i\,\underline{\boldsymbol{v}}_i^TP_i^{-1} - \underline{G}_i\underline{H}_i\overline{\boldsymbol{e}}_1\,\widetilde{\boldsymbol{q}}_i^{(3)T} = \boldsymbol{0},
\end{aligned}$$

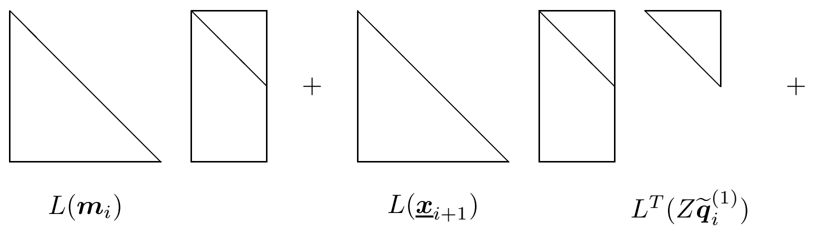
we have

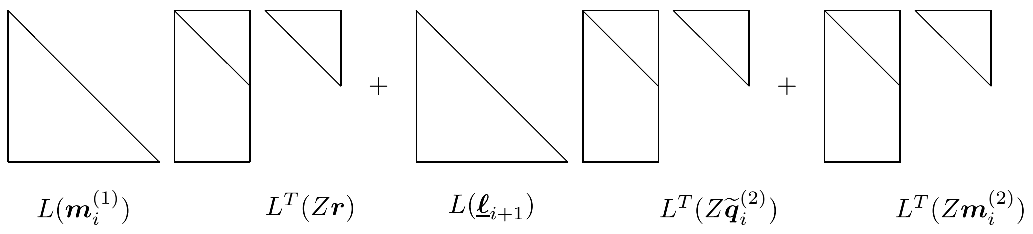
$$\Delta(M_i) = \underline{\boldsymbol{x}}_{i+1}\,\widetilde{\boldsymbol{q}}_i^{(1)T} + \boldsymbol{m}_i^{(1)}\,\boldsymbol{r}^T + \underline{\boldsymbol{\ell}}_{i+1}\,\widetilde{\boldsymbol{q}}_i^{(2)T} + \boldsymbol{e}_1\,\boldsymbol{m}_i^{(2)T},$$

where

$$\boldsymbol{m}_i^{(1)} = \underline{\boldsymbol{\ell}}_i + \underline{G}_i\underline{H}_i\,\widetilde{\boldsymbol{p}}_i^{(2)}, \qquad \boldsymbol{m}_i^{(2)T} = \boldsymbol{g}_i^T\,\underline{H}_iP_i^{-1}.$$

The first column of M_i is $\boldsymbol{m}_i = \underline{G}_i\underline{H}_i\widetilde{\boldsymbol{p}}_i$ and is not a d-vector. The d-representation of M_i has the form


+
+
$L(\boldsymbol{m}_i)$
$L(\underline{\boldsymbol{x}}_{i+1})$
$L^T(Z\widetilde{\boldsymbol{q}}_i^{(1)})$


+
+
$L(\boldsymbol{m}_i^{(1)})$
$L^T(Z\boldsymbol{r})$
$L(\underline{\boldsymbol{\ell}}_{i+1})$
$L^T(Z\widetilde{\boldsymbol{q}}_i^{(2)})$
$L^T(Z\boldsymbol{m}_i^{(2)})$

Note that the first three right d-vectors of M_i coincide with the first three right d-vectors of P_i^{-1}.

Matrix $K_i = P_i^{-1}\overline{H}_i$ size $m \times m$ d-rank=3

$$\begin{aligned}
&\Delta(K_i) = \Delta(P_i^{-1})\overline{H}_i + P_i^{-1}\Delta(\overline{H}_i)\\
&= \big(\widetilde{\boldsymbol{p}}_i^{(1)}\,\widetilde{\boldsymbol{q}}_i^{(1)T} + \widetilde{\boldsymbol{p}}_i^{(2)}\,\boldsymbol{r}^T - \widetilde{\boldsymbol{p}}_i^{(3)}\,\widetilde{\boldsymbol{q}}_i^{(2)T} - \overline{\boldsymbol{e}}_1\,\widetilde{\boldsymbol{q}}_i^{(3)T}\big)\overline{H}_i + P_i^{-1}\big(\overline{\boldsymbol{x}}_i\,\underline{\boldsymbol{v}}_i^T - \overline{\boldsymbol{\ell}}_i\,\overline{\boldsymbol{v}}_i^T\big).
\end{aligned}$$

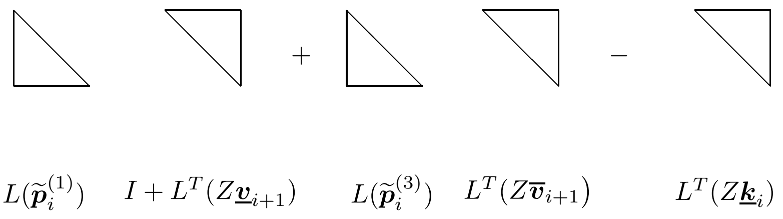
Since

$$\begin{aligned}
&\widetilde{\boldsymbol p}_i^{(1)}\,\widetilde{\boldsymbol q}_i^{(1)T}\overline{H}_i + P_i^{-1}\overline{\boldsymbol x}_i\,\underline{\boldsymbol v}_i^T = \widetilde{\boldsymbol p}_i^{(1)}\,\underline{\boldsymbol v}_{i+1}^T,\\
&\widetilde{\boldsymbol p}_i^{(2)}\,\boldsymbol r^T\overline{H}_i - P_i^{-1}\overline{\boldsymbol\ell}_i\,\overline{\boldsymbol v}_i^T = \boldsymbol 0,\\
&-\widetilde{\boldsymbol p}_i^{(3)}\,\widetilde{\boldsymbol q}_i^{(2)T}\overline{H}_i = \widetilde{\boldsymbol p}_i^{(3)}\,\overline{\boldsymbol v}_{i+1}^T,
\end{aligned}$$

we have

$$\Delta(K_i) = \widetilde{\boldsymbol p}_i^{(1)}\underline{\boldsymbol v}_{i+1}^T + \widetilde{\boldsymbol p}_i^{(3)}\overline{\boldsymbol v}_{i+1}^T - \overline{\boldsymbol e}_1\underline{\boldsymbol k}_i^T, \quad \text{where} \quad \underline{\boldsymbol k}_i^T = \widetilde{\boldsymbol q}_i^{(3)T}\overline{H}_i.$$

The first column of K_i is the d-vector $P_i^{-1}\overline{\boldsymbol x}_i = \widetilde{\boldsymbol p}_i^{(1)}$. The d-representation of K_i has the form


+
−
$L(\widetilde{\boldsymbol p}_i^{(1)})$
$I + L^T(Z\underline{\boldsymbol v}_{i+1})$
$L(\widetilde{\boldsymbol p}_i^{(3)})$
$L^T(Z\overline{\boldsymbol v}_{i+1})$
$L^T(Z\underline{\boldsymbol k}_i)$

Note that the first two right d-vectors of K_i coincide with the two right d-vectors of $\overline{H}_{i+1}$.

Matrix $L_i = P_i^{-1}\overline{G}_i\underline{G}_i$ size $m \times n$ d-rank=3

$$\begin{aligned}
\Delta(L_i) &= \Delta(P_i^{-1})\overline{G}_i\underline{G}_i + P_i^{-1}\Delta(\overline{G}_i)\underline{G}_i + P_i^{-1}\overline{G}_i\Delta(\underline{G}_i)\\
&= \big(\widetilde{\boldsymbol p}_i^{(1)}\,\widetilde{\boldsymbol q}_i^{(1)T} + \widetilde{\boldsymbol p}_i^{(2)}\,\boldsymbol r^T - \widetilde{\boldsymbol p}_i^{(3)}\,\widetilde{\boldsymbol q}_i^{(2)T} - \overline{\boldsymbol e}_1\,\widetilde{\boldsymbol q}_i^{(3)T}\big)\overline{G}_i\underline{G}_i\\
&\quad + P_i^{-1}\big(\overline{\boldsymbol x}_i\,\underline{\boldsymbol u}_i^T + \overline{\boldsymbol\ell}_i\,\overline{\boldsymbol w}_i^T - \overline{\boldsymbol e}_1\,\boldsymbol e_n^T\big)\underline{G}_i + P_i^{-1}\overline{G}_i\big(\underline{\boldsymbol x}_i\,\underline{\boldsymbol u}_i^T + \underline{\boldsymbol\ell}_i\,\overline{\boldsymbol w}_i^T + \boldsymbol e_1\,\boldsymbol g_i^T\big).
\end{aligned}$$

Since

$$\begin{aligned}
&\widetilde{\boldsymbol p}_i^{(1)}\,\widetilde{\boldsymbol q}_i^{(1)T}\overline{G}_i\underline{G}_i + P_i^{-1}\overline{\boldsymbol x}_i\,\underline{\boldsymbol u}_i^T\underline{G}_i = -\widetilde{\boldsymbol p}_i^{(1)}\,\underline{\boldsymbol u}_{i+1}^T,\\
&\widetilde{\boldsymbol p}_i^{(2)}\,\boldsymbol r^T\overline{G}_i\underline{G}_i + P_i^{-1}\overline{\boldsymbol\ell}_i\,\overline{\boldsymbol w}_i^T\underline{G}_i + P_i^{-1}\overline{G}_i\boldsymbol e_1\,\boldsymbol g_i^T = \boldsymbol 0,\\
&-\widetilde{\boldsymbol p}_i^{(3)}\,\widetilde{\boldsymbol q}_i^{(2)T}\overline{G}_i\underline{G}_i + P_i^{-1}\overline{G}_i\underline{\boldsymbol\ell}_i\,\overline{\boldsymbol w}_i^T = \widetilde{\boldsymbol p}_i^{(3)}\,\overline{\boldsymbol w}_{i+1}^T,\\
&-\overline{\boldsymbol e}_1\,\widetilde{\boldsymbol q}_i^{(3)T}\overline{G}_i\underline{G}_i - P_i^{-1}\overline{\boldsymbol e}_1\,\boldsymbol e_n^T\underline{G}_i + P_i^{-1}\overline{G}_i\,\underline{\boldsymbol x}_i\,\underline{\boldsymbol u}_i^T = -\overline{\boldsymbol e}_1\big(\underline{\boldsymbol u}_i^T + \widetilde{\boldsymbol q}_i^{(3)T}\overline{G}_i\underline{G}_i\big),
\end{aligned}$$

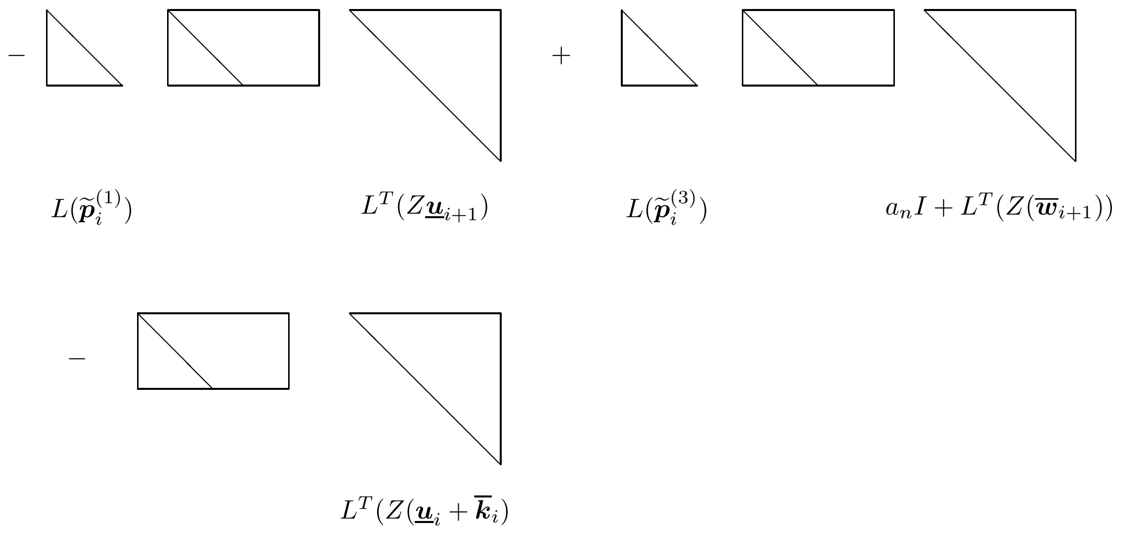
we have

$$\Delta(L_i) = -\widetilde{\boldsymbol p}_i^{(1)}\,\underline{\boldsymbol u}_{i+1}^T + \widetilde{\boldsymbol p}_i^{(3)}\,\overline{\boldsymbol w}_{i+1}^T - \overline{\boldsymbol e}_1\,\big(\underline{\boldsymbol u}_i^T + \overline{\boldsymbol k}_i^T\big), \quad \text{where} \quad \overline{\boldsymbol k}_i^T = \widetilde{\boldsymbol q}_i^{(3)T}\overline{G}_i\underline{G}_i.$$

The first column of L_i is the d-vector

$$P_i^{-1}\overline{G}_i\underline{G}_i\boldsymbol e_1 = a_nP_i^{-1}\overline{G}_i\underline{\boldsymbol\ell}_i = a_nP_i^{-1}\overline{\boldsymbol p}_i = a_n\widetilde{\boldsymbol p}_i^{(3)}.$$

The d-representation of L_i has the form


$-$
$+$
$L(\widetilde{\boldsymbol{p}}_i^{(1)})$
$L^T(Z\underline{\boldsymbol{u}}_{i+1})$
$L(\widetilde{\boldsymbol{p}}_i^{(3)})$
$a_n I + L^T(Z(\overline{\boldsymbol{w}}_{i+1}))$
$-$
$L^T(Z(\underline{\boldsymbol{u}}_i + \overline{\boldsymbol{k}}_i)$

Note that the first two right d-vectors of L_i coincide with the first two right d-vectors of $\underline{G}_{i+1}$.

7.6 Cost of matrix by vector product

Table 4 lists the cost of computing matrix by vector products, for the matrices used in algorithm `st`. As already said, this cost does not include the cost for the transformation of the vectors of the d-description of the matrix itself and the cost of transforming the input vector.

matrix	cost
$\overline{G}_i$	$2\pi_n + 3\pi_m$
$\overline{G}_i^T$	$3\pi_n + 2\pi_m$
$\underline{G}_i$	$5\pi_n$
$\overline{H}_i$	$5\pi_m$
$\underline{H}_i$	$3\pi_n + 3\pi_m$
$\underline{H}_i^T$	$2\pi_n + 4\pi_m$
P_i^{-1}	$7\pi_m$
M_i	$5\pi_n + 4\pi_m$
K_i	$5\pi_m$
L_i	$3\pi_n + 3\pi_m$

Table 4: Cost of computing matrix by vector products for algorithm `st`.

We observe that frequently in the recursive relations of the vectors we have to compute the products $T_1\boldsymbol{z}$ and $T_2\boldsymbol{z}$ of two different matrices by the same input vector $\boldsymbol{z}$. If T_1 and T_2 have some right d-vectors in common, for the product which is made after we can exploit some intermediate results of the product which is made before. For example, if T_1 and T_2 have two right d-vectors in common and the length of $\boldsymbol{z}$ is n, the cost of the second product decreases by $2\pi_n$. An analogous decrease holds for the products $\boldsymbol{z}^T T_1$ and $\boldsymbol{z}^T T_2$ if T_1 and T_2 have some left d-vectors in common.

7.7 Cost of algorithm `st`

7.7.1 Transformation phase

(a) Initially the d-description of matrix B_0^{-1} is computed by solving two systems with matrix B_0

$$\widehat{\boldsymbol{f}}_1 = B_0^{-1}\boldsymbol{f}_{-1}, \qquad \widehat{\boldsymbol{f}}_2 = B_0^{-1}\boldsymbol{e}_1.$$

Hence the vectors $\overline{\boldsymbol{x}}_0 = \overline{F}^T\widehat{\boldsymbol{f}}_1$, $\underline{\boldsymbol{x}}_0 = \widehat{\boldsymbol{f}}_1$, $\underline{\boldsymbol{\ell}}_0 = \boldsymbol{s}_2$, $\overline{\boldsymbol{\ell}}_0 = \overline{F}^T\widehat{\boldsymbol{f}}_2$, $\boldsymbol{g}_0^T = -\boldsymbol{e}_n^T$ and $\boldsymbol{h}_0^T = \boldsymbol{0}^T$ are available. The vectors $\overline{\boldsymbol{u}}_0^T = \widehat{\boldsymbol{f}}_1^T JB_1$, $\underline{\boldsymbol{u}}_0^T = \widehat{\boldsymbol{f}}_2^T JB_1$, $\overline{\boldsymbol{v}}_0^T = \boldsymbol{s}_1^T J\underline{F}R$, $\underline{\boldsymbol{v}}_0^T = \boldsymbol{s}_2^T J\underline{F}R$ are computed. All these vectors are transformed. The cost of this initialization is not considered, since we are interested into the cost of the general ith step of the algorithm.

(b) Now the ith step begins, for $i = 0, \dots, p-1$. First matrices $P_i = I - \overline{G}_i\underline{H}_i$ and P_i^{-1} are computed:

1. for the vector $\overline{\boldsymbol{p}}_i = \overline{G}_i\underline{\boldsymbol{\ell}}_i$, the product by matrix $\overline{G}_i$ of an input vector already transformed costs $2\pi_n + 3\pi_m$.
2. for the vector $\underline{\boldsymbol{p}}_i = \overline{F}^T\boldsymbol{e}_1 - \overline{G}_i\underline{\boldsymbol{x}}_i$, as above, the cost is $2\pi_n + 3\pi_m$.
3. for the vector $\underline{\boldsymbol{q}}_i^T = \underline{\boldsymbol{u}}_i^T\underline{H}_i$, one product by matrix $\underline{H}_i^T$ and input vector already transformed, so the cost is $2\pi_n + 4\pi_m$.
4. for the vector $\overline{\boldsymbol{q}}_i^T = \boldsymbol{r}^T - \overline{\boldsymbol{u}}_i^T\underline{H}_i$, as above, the cost is $2\pi_n + 4\pi_m$.
5. for the vector $\underline{\boldsymbol{g}}_i^T = \boldsymbol{g}_i^T\underline{H}_i$, as above, the cost is $2\pi_n + 4\pi_m$.
6. The vectors $\widetilde{\boldsymbol{p}}_i^{(1)} = P_i^{-1}\overline{\boldsymbol{x}}_i$, $\widetilde{\boldsymbol{p}}_i^{(2)} = P_i^{-1}\overline{\boldsymbol{\ell}}_i$, $\widetilde{\boldsymbol{p}}_i^{(3)} = P_i^{-1}\overline{\boldsymbol{p}}_i$, $\widetilde{\boldsymbol{p}}_i = P_i^{-1}\overline{\boldsymbol{e}}_1$, $\widetilde{\boldsymbol{q}}_i^{(1)T} = \underline{\boldsymbol{q}}_i^T P_i^{-1}$, $\widetilde{\boldsymbol{q}}_i^{(2)T} = \overline{\boldsymbol{v}}_i^T P_i^{-1}$, $\widetilde{\boldsymbol{q}}_i^{(3)T} = \underline{\boldsymbol{v}}_i^T P_i^{-1}$, $\boldsymbol{m}_i^{(2)T} = \underline{\boldsymbol{g}}_i^T P_i^{-1}$ (actually this last vector belongs to the d-description of M_i) are computed by solving four systems with matrix P_i and four systems with matrix P_i^T. The cost is $2\sigma_{m,4}(4)$. The vectors are then transformed with cost $8\pi_m$.

Now the vectors which enter into the d-description of $\overline{G}_{i+1}$, $\underline{G}_{i+1}$, $\overline{H}_{i+1}$, $\underline{H}_{i+1}$, K_i and L_i are computed.

Cost of the pair

$$\begin{aligned}\overline{\boldsymbol{x}}_{i+1} &= -\overline{H}_i P_i^{-1}\overline{\boldsymbol{x}}_i = -\overline{H}_i\widetilde{\boldsymbol{p}}_i^{(1)},\\ \underline{\boldsymbol{x}}_{i+1} &= \underline{\boldsymbol{x}}_i + \underline{G}_i\underline{H}_i P_i^{-1}\overline{\boldsymbol{x}}_i = \underline{\boldsymbol{x}}_i + \underline{G}_i\underline{H}_i\widetilde{\boldsymbol{p}}_i^{(1)}.\end{aligned}$$

The input vector $\widetilde{\boldsymbol{p}}_i^{(1)}$ has already been transformed. Matrices $\overline{H}_i$ and $\underline{H}_i$ have two right d-vectors in common, so the cost of the second product decreases by $2\pi_m$. The result of $\underline{H}_i\widetilde{\boldsymbol{p}}_i^{(1)}$ must be transformed, then the product by $\underline{G}_i$ is performed. All this procedure costs $9\pi_n + 6\pi_m$.

The same cost holds for the pair

$$\begin{aligned}\overline{\boldsymbol{\ell}}_{i+1} &= \overline{\boldsymbol{\ell}}_i + \overline{H}_i P_i^{-1}\overline{G}_i\underline{\boldsymbol{\ell}}_i = \overline{\boldsymbol{\ell}}_i + \overline{H}_i\widetilde{\boldsymbol{p}}_i^{(3)},\\ \underline{\boldsymbol{\ell}}_{i+1} &= -\underline{G}_i\underline{\boldsymbol{\ell}}_i - \underline{G}_i\underline{H}_i P_i^{-1}\overline{G}_i\underline{\boldsymbol{\ell}}_i = -\underline{G}_i\big(\underline{\boldsymbol{\ell}}_i + \underline{H}_i\widetilde{\boldsymbol{p}}_i^{(3)}\big).\end{aligned}$$

Cost of the pair

$$\begin{aligned}\underline{\boldsymbol{u}}_{i+1}^T &= -\underline{\boldsymbol{u}}_i^T\underline{G}_i - \underline{\boldsymbol{u}}_i^T\underline{H}_i P_i^{-1}\overline{G}_i\underline{G}_i = -\big(\underline{\boldsymbol{u}}_i^T + \widetilde{\boldsymbol{q}}_i^{(1)T}\,\overline{G}_i\big)\underline{G}_i,\\ \underline{\boldsymbol{v}}_{i+1}^T &= \underline{\boldsymbol{v}}_i^T + \underline{\boldsymbol{u}}_i^T\underline{H}_i P_i^{-1}\overline{H}_i = \underline{\boldsymbol{v}}_i^T + \widetilde{\boldsymbol{q}}_i^{(1)T}\,\overline{H}_i,\end{aligned}$$

The input vector $\widetilde{\boldsymbol{q}}_i^{(1)}$ has already been transformed. Matrices $\overline{G}_i$ and $\overline{H}_i$ have two left d-vectors in common, so the cost of the second product decreases by $2\pi_m$. The result of $\widetilde{\boldsymbol{q}}_i^{(1)T}\overline{G}_i$ must be transformed, then the product by $\underline{G}_i$ is performed. All this procedure costs $9\pi_n + 5\pi_m$.

The same cost holds for the pair

$$\begin{aligned}\overline{\boldsymbol{u}}_{i+1}^T &= \overline{\boldsymbol{u}}_i^T + \overline{\boldsymbol{v}}_i^T P_i^{-1}\overline{G}_i\underline{G}_i = \overline{\boldsymbol{u}}_i^T + \widetilde{\boldsymbol{q}}_i^{(2)T}\,\overline{G}_i\underline{G}_i,\\ \overline{\boldsymbol{v}}_{i+1}^T &= -\overline{\boldsymbol{v}}_i^T P_i^{-1}\overline{H}_i = -\widetilde{\boldsymbol{q}}_i^{(2)T}\,\overline{H}_i,\end{aligned}$$

for the pair

$$\begin{aligned}\boldsymbol{g}_{i+1}^T &= -\boldsymbol{g}_i^T\underline{G}_i - \boldsymbol{g}_i^T\underline{H}_i P_i^{-1}\overline{G}_i\underline{G}_i = -\big(\boldsymbol{g}_i^T + \boldsymbol{m}_i^{(2)T}\,\overline{G}_i\big)\underline{G}_i,\\ \boldsymbol{h}_{i+1}^T &= \boldsymbol{h}_i^T + \boldsymbol{g}_i^T\underline{H}_i P_i^{-1}\overline{H}_i = \boldsymbol{h}_i^T + \boldsymbol{m}_i^{(2)T}\,\overline{H}_i,\end{aligned}$$

and for the pair

$$\overline{\boldsymbol{k}}_i^T = \widetilde{\boldsymbol{q}}_i^{(3)T}\overline{G}_i\underline{G}_i \quad\text{and}\quad \underline{\boldsymbol{k}}_i^T = \widetilde{\boldsymbol{q}}_i^{(3)T}\overline{H}_i.$$

The transformation of the 12 output vectors so computed costs $6\pi_n + 6\pi_m$.

For matrix M_i we have still to compute

$$\boldsymbol{m}_i^{(1)} = \underline{\boldsymbol{\ell}}_i + \underline{G}_i\underline{H}_i\,\widetilde{\boldsymbol{p}}_i^{(2)} \quad\text{and}\quad \boldsymbol{m}_i = \underline{G}_i\underline{H}_i\widetilde{\boldsymbol{p}}_i.$$

Both input vectors have already been transformed, so the two multiplications by $\underline{H}_i$ and by $\underline{G}_i$ and the transformations of the results cost $20\pi_n + 6\pi_m$.

No cost is required for vector $\overline{\boldsymbol{w}}_{i+1}$.

7.7.2 Substitution phase

For the forward phase

1. $\underline{\boldsymbol{y}}_i^{(0)} = B_0^{-1}(\boldsymbol{b}_i^{(0)} - B_1\boldsymbol{c})$ two products are required, by B_1 and by B_0^{-1}, both input vectors have not yet been transformed, so the cost is $2\pi_n$ and $6\pi_n$. For $i = 1, \dots, N$ the cost is $8\pi_n N$.

2. $\boldsymbol{z}_{2j}^{(i)} = P_i^{-1}\overline{\boldsymbol{y}}_{2j}^{(i)}$, the product by matrix P_i^{-1} of an input vector not yet transformed costs $8\pi_m$. Vector $\boldsymbol{z}_{2j}^{(i)}$ is then transformed with cost π_m. For the N vectors to be computed the cost is $9\pi_m N$.

3. Cost of the pair

$$\overline{\boldsymbol{y}}_j^{(i+1)} = \overline{\boldsymbol{y}}_{2j-1}^{(i)} - \overline{H}_i\, \boldsymbol{z}_{2j}^{(i)} \quad \text{and} \quad \underline{\boldsymbol{y}}_j^{(i+1)} = \underline{\boldsymbol{y}}_{2j}^{(i)} + M_i\, \overline{\boldsymbol{y}}_{2j}^{(i)}.$$

The product by $\overline{H}_i$ of an already transformed vector costs $5\pi_m$. The product by M_i costs $5\pi_n + \pi_m$, since M_i has three right d-vectors in common with P_i^{-1} and the product of P_i^{-1} with the same input vector has just been made. For the N pairs to be computed the cost is $(5\pi_n + 6\pi_m)N$.

For the backward phase

1. When they are computed, the input vectors $\underline{\boldsymbol{x}}_k^{(i)} - \underline{\boldsymbol{y}}_k^{(i)}$, with $i = p-1, \dots, 1$, must be transformed for all odd indices k and $\overline{\boldsymbol{x}}_{2j}^{(i)}$ must be transformed for all even indices $2j$. Moreover $\overline{F}^T \underline{\boldsymbol{x}}_j^{(1)}$ must be transformed for $j = 1, \dots, N/2$. The cost is $(\pi_n/2 + \pi_m)N$.

2. By (48) and (52) the N vectors $\underline{\boldsymbol{x}}_{2j-1}^{(i)}$ for $i = p-1, \dots, 1$ are computed performing products by $\underline{G}_i$ and by $\underline{H}_i$. All input vectors have already been transformed. By the linearity, the last backtransformation can be counted only once. Hence the cost for all these vectors is $(7\pi_n + 3\pi_m)N$.

3. By (50) the $N/2$ vectors $\overline{\boldsymbol{x}}_{2j}^{(i)} = \boldsymbol{z}_{2j}^{(i)} + L_i\left(\underline{\boldsymbol{x}}_k^{(i+h)} - \underline{\boldsymbol{y}}_k^{(i+h)}\right) - K_i\, \overline{\boldsymbol{x}}_{j+1}^{(i+1)}$ are computed performing products by L_i and by K_i. We take into account that the products $K_i\, \overline{\boldsymbol{x}}_{j+1}^{(i+1)}$ for odd indices j correspond to products $\underline{H}_{i+1}\, \overline{\boldsymbol{x}}_{2j}^{(i+1)}$ already considered for formula (48). Since K_i and $\underline{H}_{i+1}$ have two right d-vectors in common, the cost of $K_i\, \overline{\boldsymbol{x}}_{j+1}^{(i+1)}$ with j odd is $3\pi_m$. The other half of the products by K_i corresponding to even indices have full cost $5\pi_m$. The same argument holds for L_i. Hence half of the products cost $\pi_n + 3\pi_m$ and the other half cost $3\pi_n + 3\pi_m$. As in the previous item 2. by the linearity, the last backtransformation can be counted only once. Hence the cost for all the vectors to be computed by (50) is $(\pi_n + 3\pi_m)N$.

7.7.3 Overall cost

For the transformation phase the cost of the ith step is

$$t_i = 90\pi_n + 70\pi_m + 2\sigma_{m,4}(4)$$

and must be multiplied by $\log N$. For the substitution phase the cost is

$$s = (21.5\pi_n + 22\pi_m)N.$$

Since $2\sigma_{m,4}(4) = 29m^2$, we have

$$c_{\texttt{st}} = t_i \log N + s = (90\pi_n + 70\pi_m + 29m^2)\log N + (21.5\pi_n + 22\pi_m)N. \quad (53)$$

8 Analysis of stability

Even if our primary interest in studying the behaviour of the introduced algorithms concerns the computational costs, the stability issue is also briefly taken into consideration. We do not perform here a detailed stability analysis of the single methods and refer to known results presented in the literature.

1. For algorithm `ge` we refer to the discussion presented in [14]. The hypothesis of block diagonal dominance, expressed in our case as

$$\|B_0^{-1}\|^{-1} \geq \|B_1\| + \|B_{-1}\|, \quad (54)$$

guarantees that block Gaussian elimination is stable, i.e. that the matrix $\widehat{L}_i$ effectively computed in place of $B_1 S_i^{-1}$ satisfies

$$\widehat{L}_i S_i = B_1 + E_i, \quad \text{with} \quad \|E_i\| \leq c(n,i)\,\epsilon\,\|\widehat{L}_i\|\,\|S_i\|,$$

where ϵ is the machine precision and $c(n,i)$ is a slowly increasing function of the size n and the recursive index i. In practice, this means that the stability of `ge` is guaranteed if the Schur complement S_i^{-1} defined in (9) is well computed for any i. However, also the simpler point diagonal dominance by columns (in our case the detail "by columns" can be dropped) guarantees that block Gaussian elimination is stable.

2. For algorithms `cr` and `mcr` we refer to [24], where the stability of the cyclic reduction applied to block tridiagonal systems is studied under the assumption that the matrix is block diagonally dominant, expressed in our case as

$$s = \beta + \gamma \leq 1, \quad \text{where} \quad \beta = \|B_1 B_0^{-1}\| \quad \text{and} \quad \gamma = \|B_{-1} B_0^{-1}\|, \quad (55)$$

for some multiplicative norm. By forward error analysis the following result is shown in [24]:

$$\frac{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq f(n,N)\,\kappa\,\epsilon,$$

where $\| \ \|$ is an infinite block norm, $\kappa = \|B_0^{-1}\|(\|B_0\|\|\boldsymbol{x}\| + \|\boldsymbol{b}\|)/\|\boldsymbol{x}\|$ depends on the condition number of B_0, $\widetilde{\boldsymbol{x}}$ is the computed solution, and

$$f(n,N) = \begin{cases} c_n N^2 \log N & \text{if} \quad s = 1, \\ c_n g(s) \log N & \text{if} \quad s < 1, \end{cases}$$

c_n being a constant depending linearly on n and $g(s)$ being a fast increasing function of s alone. If s is too close to 1, then $g(s)$ should be replaced by N^2. For example, if $s = 0.9$ we have $g(s) \sim 100$ and the first bound should be taken for $N \leq 10$, while for $s = 0.99$ we have $g(s) \sim 10000$ and the first bound should be taken for $N \leq 100$.

3. For algorithm `st` we refer to [18], which analyzes the stability of the divide-and-conquer algorithm described by Stewart in [22] and gives an upper bound to the residual. More precisely, the general case of the system $AX = B$ is considered and the computed X is shown to satisfy $AX = B + \Delta B$, with $\|\Delta B\| \leq \eta \|A\| \, \|X\|$, η being a small multiple of ϵ. In the case where the right-hand side B is a vector, this residual stability is the same as the backward stability. Two important classes of matrices, i.e. the block diagonally dominant matrices (satisfying (54)) and the M-matrices, are identified for which the residual stability is guaranteed.

All the previous results are obtained under the hypothesis that the rounding errors which arise in multiplying and inverting matrices at block size are under control. Then our attention on the stability issue is shifted to the techniques used for computing the inverse matrices, i.e. to the SMW formula, used for reducing the size of matrices to be inverted, and to the exploitation of the Toeplitz structure for computing products.

Regarding SMW formula, its stability depends on the conditioning of the matrix effectively inverted [25] (examined separately for each algorithm in the next subsection) and on the stability of the method actually used for the inversion at block size. As already said, we suggest for this inversion a generalization [19] of Levinson algorithm for Toeplitz like matrices. The stability of Levinson algorithm has been analyzed in many papers (see for example [6], [10]). It has been proved that the algorithm is stable for positive definite Toeplitz matrices. There are variants of Levinson algorithm which embed look-ahead techniques and guarantee weak stability even in the nonsymmetric case [7]. Anyway, our numerical experiments have shown that if all the matrices to be inverted enjoy some diagonally dominance property, the errors arising during the computations remain of the order of the machine precision.

8.1 Conditioning of the Sherman-Morrison-Woodbury matrix

We investigate now the conditioning $\kappa(P) = \|P\| \, \|P^{-1}\|$ of the matrix P inverted at block size when the SMW formula is applied by the different methods.

SMW formula updates the inverse of the matrix

$$T = B - UV$$

in terms of the inverse of B

$$T^{-1} = B^{-1} + B^{-1}UP^{-1}VB^{-1}, \quad \text{where} \quad P = I - VB^{-1}U.$$

In the following we use also the relation

$$P^{-1} = I + VT^{-1}U.$$

Since SMW formula is just an intermediate step in a larger algorithm whose stability cannot be expected unless some dominance property holds, we will restrict our analysis to the case of point diagonal dominance. Our aim is to estimate the growth of $\kappa(P)$ with respect to $\kappa(A)$. The norm $\| \ \|$ used will be determined for each method by simplicity consideration.

To this aim, we need some upper bounds on norms which can be derived considering suitable block factorizations. Let M be a partitioned matrix of the form

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}.$$

Let $S_0 = M_{11}$ and $S_1 = M_{22} - M_{21}M_{11}^{-1}M_{12}$ (the Schur complement of M_{11}). The following properties are useful for stability analysis.

1. If M is point diagonally dominant by columns, then

$$\|M_{21}M_{11}^{-1}\|_1 \leq 1.$$

2. If M is point diagonally dominant by rows, then

$$\|M_{11}^{-1}M_{12}\|_\infty \leq 1.$$

These bounds follow (see [14], probl. 12.5 p. 258) from the two factorizations (the first one is the block LU decomposition of M and the second one is the transpose of the block LU decomposition of M^T)

$$M = \begin{bmatrix} I & O \\ M_{21}M_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ O & S_1 \end{bmatrix},$$

$$M = \begin{bmatrix} M_{11} & O \\ M_{21} & S_1 \end{bmatrix} \begin{bmatrix} I & M_{11}^{-1}M_{12} \\ O & I \end{bmatrix}.$$

In both cases S_1 maintains the same diagonal dominance of M. In the case of a Toeplitz matrix, which is persymmetric, both bounds hold and we have also $\|M_{12}M_{11}^{-1}\|_1 \leq 1$ and $\|M_{11}^{-1}M_{21}\|_\infty \leq 1$.

Analogous bounds can be derived if we apply a block factorization procedure to a block tridiagonal matrix M. Then matrix L of its LU decomposition is lower block bidiagonal with identity principal blocks and subdiagonal blocks having 1-norm upper bounded by 1, while matrix U is upper block bidiagonal with principal blocks equal to the Schur complements S_i. The Schur complements S_i maintain the same dominance of M.

Since $U^{-1} = M^{-1}L$, we have

$$\|S_i^{-1}\| \leq \|U^{-1}\| \leq \|M^{-1}\| \, \|L\|. \tag{56}$$

8.1.1 Algorithm `ge`

For algorithm `ge` we consider the matrix defined in (15)

$$P_i = I - VU_i, \quad \text{where} \quad V = R\overline{F}^T B_0^{-1}, \qquad U_i = B_1 S_i^{-1} \underline{F},$$

and we have

$$S_{i+1} = (I - U_i V)B_0, \quad S_{i+1}^{-1} = B_0^{-1}(I + U_i P_i^{-1} V), \quad P_i^{-1} = I + VB_0 S_{i+1}^{-1} U_i.$$

The norm used in this case is the 1-norm.

Under the hypothesis of point diagonal dominance for the Toeplitz matrix A, applying block LU factorization to A or A^T from the considerations made above it follows that both block matrices $B_1 S_i^{-1}$ and $B_{-1} S_i^{-1}$ have 1-norm upper bounded by 1. Then we have

$$\|U_i\|_1 = \|B_1 S_i^{-1} \underline{F}\|_1 \leq \|B_1 S_i^{-1}\|_1 \leq 1.$$

and

$$\|V\|_1 = \|R\overline{F}^T B_0^{-1}\|_1 \leq \|B_{-1} B_0^{-1}\|_1 \leq 1,$$

hence

$$\|P_i\|_1 \leq 1 + \|V\|_1 \, \|U_i\|_1 \leq 2.$$

Now

$$\|P_i^{-1}\|_1 \leq 1 + \|VB_0 S_{i+1}^{-1} U_i\|_1 \leq 1 + \|V\|_1 \, \|B_0\|_1 \, \|S_{i+1}^{-1}\|_1 \, \|U_i\|_1.$$

From (56) we have

$$\|S_{i+1}^{-1}\|_1 \leq \|A^{-1}\|_1 \, (1 + \|B_1 S_i^{-1}\|_1) \leq 2 \, \|A^{-1}\|_1.$$

Then

$$\|P_i^{-1}\|_1 \leq 1 + 2 \, \|A\|_1 \, \|A^{-1}\|_1$$

and

$$\kappa_1(P_i) \leq 2 + 4 \, \kappa_1(A).$$

We conclude that the conditioning of P_i is not worse than the conditioning of A.

8.1.2 Algorithm `mcr`

For algorithm `mcr` we consider the matrix defined in (28)

$$P^{(i)} = I - V^{(i)} {H_0^{(i)}}^{-1} U^{(i)}, \quad \text{where} \quad V^{(i)} = \begin{bmatrix} V_1^{(i)} \\ \overline{F}^T \end{bmatrix}, \quad U^{(i)} = \left[\, \underline{F} \mid U_1^{(i)} \,\right]$$

and

$$V_1^{(i)} = \underline{F}^T H_{-1}^{(i)} {H_0^{(i)}}^{-1} H_1^{(i)}, \quad \text{and} \quad U_1^{(i)} = H_1^{(i)} {H_0^{(i)}}^{-1} H_{-1}^{(i)} \overline{F}.$$

We have

$$H_0^{(i+1)} = H_0^{(i)} - U^{(i)} V^{(i)}, \quad {H_0^{(i+1)}}^{-1} = {H_0^{(i)}}^{-1} + {H_0^{(i)}}^{-1} U^{(i)} {P_i^{(i)}}^{-1} V^{(i)} {H_0^{(i)}}^{-1}$$

and

$${P^{(i)}}^{-1} = I + V^{(i)} {H_0^{(i+1)}}^{-1} U^{(i)}.$$

The norm used in this case is the 1-norm and for simplicity we assume that $\|A\|_1 \geq 1$.

As already noted in Remark 5.1, matrices $H_{-1}^{(i)} {H_0^{(i)}}^{-1}$ and $H_1^{(i)} {H_0^{(i)}}^{-1}$ appear as blocks of the lower triangular factor L of the block LU factorization of the matrix $\widetilde{A} = \Omega\, A\, \Omega^T$ and matrices $H_0^{(i)}$ appear as diagonal blocks of the upper triangular factor U. Under the hypothesis of point diagonal dominance for A also $\widetilde{A}$ is point diagonally dominant and we have

$$\tau_{-1}^{(i)} = \|H_{-1}^{(i)} {H_0^{(i)}}^{-1}\|_1 \leq 1 \quad \text{and} \quad \tau_1^{(i)} = \|H_1^{(i)} {H_0^{(i)}}^{-1}\|_1 \leq 1.$$

Moreover matrices $H_{-1}^{(i)}$ and $H_1^{(i)}$ appear as blocks of the upper triangular factor U of the block LU factorization and we have

$$\|H_{-1}^{(i)}\|_1 \leq \|U\|_1 \leq \|L^{-1}\|_1 \, \|\widetilde{A}\|_1 = \|L^{-1}\|_1 \, \|A\|_1 \leq 2\, \|A\|_1,$$

and analogously

$$\|H_1^{(i)}\|_1 \leq 2\, \|A\|_1.$$

Replacing the expression of $V^{(i)}$ and $U^{(i)}$ into $P^{(i)}$ we get

$$P^{(i)} = I - F^T H^{-1} G\, H\, F,$$

where

$$F = \begin{bmatrix} \underline{F} & O \\ O & \overline{F} \end{bmatrix}, \quad H = \begin{bmatrix} I & O \\ O & H_0^{(i)} \end{bmatrix}$$

and

$$G = \begin{bmatrix} H_{-1}^{(i)} {H_0^{(i)}}^{-1} H_1^{(i)} {H_0^{(i)}}^{-1} \\ I \end{bmatrix} \left[\, I \;\middle|\; H_1^{(i)} {H_0^{(i)}}^{-1} H_{-1}^{(i)} {H_0^{(i)}}^{-1} \,\right].$$

Now we have

$$\|G\|_1 \leq \left(1 + \tau_{-1}^{(i)}\, \tau_1^{(i)}\right) \max\left\{1, \tau_1^{(i)}\, \tau_{-1}^{(i)}\right\} \leq 2,$$

$$\begin{aligned}\|P^{(i)}\|_1 &\leq 1 + \|H^{-1}\, G\, H\|_1 \leq 1 + \sqrt{2n}\; \|H^{-1}\, G\, H\|_2 = 1 + \sqrt{2n}\; \|G\|_2 \\ &\leq 1 + 2n\, \|G\|_1 \leq 1 + 4n.\end{aligned}$$

For what concerns ${P^{(i)}}^{-1}$, we have

$$\|{P^{(i)}}^{-1}\|_1 \leq 1 + \|V^{(i)}\|_1\; \|{H_0^{(i+1)}}^{-1}\|_1\; \|U^{(i)}\|_1,$$

$$\|V^{(i)}\|_1 \leq 1 + \|H_{-1}^{(i)}\, {H_0^{(i)}}^{-1}\, H_1^{(i)}\|_1 \leq 1 + \tau_{-1}^{(i)}\; \|H_1^{(i)}\|_1 \leq 1 + 2\; \|A\|_1,$$

$$\|U^{(i)}\|_1 \leq \max\left\{1, \|H_1^{(i)} {H_0^{(i)}}^{-1}\, H_{-1}^{(i)}\|_1\right\} \leq \max\left\{1, \tau_1^{(i)}\; \|H_{-1}^{(i)}\|_1\right\} \leq 2\; \|A\|_1$$

and

$$\|{H_0^{(i+1)}}^{-1}\|_1 \leq \|A^{-1}\|_1\; \|L\|_1 \leq 2\; \|A^{-1}\|_1.$$

Hence

$$\kappa_1(P^{(i)}) \leq (1 + 4n)\, \left(1 + 4(1 + 2\, \|A\|_1)\kappa_1(A)\,\right) \leq c\, \kappa_1(A),$$

where c is a slowly increasing function of n and $\|A\|_1$. We conclude that the conditioning of P_i is not worse than the conditioning of A.

8.1.3 Algorithm `st`

For algorithm `st` we consider the matrix defined in (37)

$$P_i = I - \overline{G}_i\, \underline{H}_i, \quad \overline{G}_i = \overline{F}^T \overline{E}^T A_i^{-1} U_i \underline{E}, \quad \underline{H}_i = \underline{E}^T A_i^{-1} R_i \overline{EF}.$$

The norm used in this case is the ∞-norm.

From the block LU factorization of A_{i+1}^T we get

$$A_{i+1} = \begin{bmatrix} A_i & R_i \\ U_i & A_i \end{bmatrix} = \begin{bmatrix} A_i & O \\ U_i & A_i Q_i \end{bmatrix} \begin{bmatrix} I & A_i^{-1} R_i \\ O & I \end{bmatrix}, \qquad (57)$$

where

$$Q_i = I - A_i^{-1} U_i A_i^{-1} R_i.$$

It is easy to verify that P_i is the leading principal minor of order m of Q_i, i.e.

$$P_i = \overline{F}^T \overline{E}^T Q_i \overline{EF},$$

and that Q_i is block lower triangular. Under the hypothesis of point diagonal dominance for the Toeplitz matrix A (hence of A_{i+1}), we have

$$\|A_i^{-1} R_i\|_\infty \leq 1 \quad \text{and} \quad \|A_i^{-1} U_i\|_\infty \leq 1,$$

and

$$\|P_i\|_\infty \leq \|Q_i\|_\infty \leq 1 + \|A_i^{-1}U_i\|_\infty \, \|A_i^{-1}R_i\|_\infty \leq 2.$$

For what concerns ${P^{(i)}}^{-1}$, from (57) we have that the right lower block of A_{i+1}^{-1} is $Q_i^{-1}A_i^{-1}$. Hence

$$\|P_i^{-1}\|_\infty \leq \|Q_i^{-1}\|_\infty \leq \|Q_i^{-1}A_i^{-1}\|_\infty \|A_i\|_\infty \leq \|A_{i+1}^{-1}\|_\infty \|A\|_\infty.$$

To put into relation the norm of A_{i+1}^{-1} with that of A^{-1} we consider the decomposition

$$A = \begin{bmatrix} A_{i+1} & \overline{R}_{i+1} \\ \overline{U}_{i+1} & B \end{bmatrix} = \overline{L}\,\overline{U}, \quad \overline{L} = \begin{bmatrix} A_{i+1} & O \\ \overline{U}_{i+1} & S \end{bmatrix}, \quad \overline{U} = \begin{bmatrix} I & A_{i+1}^{-1}\overline{R}_{i+1} \\ O & I \end{bmatrix},$$

where B and S are suitable matrices. We see that A_{i+1}^{-1} is the leading principal minor of order n_{i+1} of $\overline{L}^{\,-1}$. Hence

$$\|A_{i+1}^{-1}\|_\infty \leq \|\overline{L}^{\,-1}\|_\infty \leq \|\overline{U}\|_\infty \, \|A^{-1}\|_\infty \leq 2 \, \|A^{-1}\|_\infty,$$

and

$$\kappa_\infty(P_i) \leq 2 \, \|A_{i+1}^{-1}\|_\infty \|A\|_\infty \leq 4 \, \kappa_\infty(A).$$

We conclude that the conditioning of P_i is not worse than the conditioning of A.

8.2 Stability of the computation of products

The product of Toeplitz-like matrices by vectors is performed by multiplying lower and upper triangular Toeplitz matrices by means of discrete Fourier transforms, as described in Section 3.2. We have then to find error bounds for the operations listed in (9).

Let $fl(\cdot)$ denote the result of a floating point computation with machine precision ϵ. In a first order analysis of the error we have for the multiplication

$$fl(\widetilde{x} \cdot \widetilde{y}) \sim x\,y + \delta(x,y), \quad \text{where} \quad \delta(x,y) = x\,y\,\widetilde{\epsilon} + y\,\delta(x) + x\,\delta(y), \tag{58}$$

where x and y are two complex numbers, $\widetilde{x}$ and $\widetilde{y}$ their machine approximations, $\delta(x) = \widetilde{x} - x$ and $\delta(y) = \widetilde{y} - y$ their errors, and $\widetilde{\epsilon}$ is the local error of the complex multiplication, with $|\widetilde{\epsilon}| < c_1\,\epsilon$, for a small constant c_1.

For the FFT we use a componentwise error bound given in [1]: for a vector $\boldsymbol{x}$ of length n (power of 2) we have

$$|\delta(\mathcal{F}(\boldsymbol{x})_j)| = |fl\big(\mathcal{F}(\boldsymbol{x})_j\big) - \mathcal{F}(\boldsymbol{x})_j| \leq c_2\,\sqrt{n}\,\epsilon\,\|\mathcal{F}(\boldsymbol{x})\|_\infty \tag{59}$$

where c_2 is a constant. A similar bound holds for the inverse transformation. Referring to (9) we have from (58) and (59)

$$|\delta(w_j)| \leq c_2\,\sqrt{2n}\,\epsilon\,\|\boldsymbol{w}\|_\infty, \qquad |\delta(g_j)| \leq c_2\,\sqrt{2n}\,\epsilon\,\|\boldsymbol{g}\|_\infty,$$

$$\widetilde{h}_j = fl\big(\widetilde{g}_j \cdot \widetilde{w}_j\big) \sim h_j + \delta(h_j), \quad \text{where} \quad \delta(h_j) = h_j\,\widetilde{\epsilon}_j + w_j\,\delta(g_j) + g_j\,\delta(w_j).$$

Since $|g_j| \le \|\boldsymbol{g}\|_\infty \le \|\boldsymbol{s}\|_1$ (and similarly $|w_j| \le \|\boldsymbol{w}\|_\infty \le \|\boldsymbol{v}\|_1$) we have

$$\begin{aligned}
&|h_j| \le \|\boldsymbol{s}\|_1 \|\boldsymbol{v}\|_1,\\
&|\delta(h_j)| \le c_1 |h_j|\,\epsilon + |w_j|\,|\delta(g_j)| + |g_j|\,\delta|(w_j)| \le (c_1 + 2c_2\sqrt{2n})\,\epsilon\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1,\\
&|\widetilde{h}_j| \le \big[1 + (c_1 + 2c_2\sqrt{2n})\,\epsilon\big]\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1.
\end{aligned}$$

Now

$$\begin{aligned}
|\delta(z_j)| &= |fl\big(\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}})_j\big) - \mathcal{F}^{-1}(\boldsymbol{h})_j|\\
&\le |fl\big(\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}})_j\big) - \mathcal{F}^{-1}(\widetilde{\boldsymbol{h}})_j| + |\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}})_j - \mathcal{F}^{-1}(\boldsymbol{h})_j|\\
&\le c_2\,\sqrt{2n}\,\epsilon\,\|\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}})\|_\infty + |\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}} - \boldsymbol{h})_j|\\
&\le c_2\,\sqrt{2n}\,\epsilon\,\|\widetilde{\boldsymbol{h}}\|_\infty + \|\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}} - \boldsymbol{h})\|_\infty,
\end{aligned}$$

but

$$\|\mathcal{F}^{-1}(\widetilde{\boldsymbol{h}} - \boldsymbol{h})\|_\infty \le \|\widetilde{\boldsymbol{h}} - \boldsymbol{h}\|_\infty = \max_j |\delta(h_j)| \le (c_1 + 2c_2\sqrt{2n})\,\epsilon\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1,$$

hence

$$|\delta(z_j)| \le c_2\,\sqrt{2n}\,\epsilon\,\|\widetilde{\boldsymbol{h}}\|_\infty + (c_1 + 2c_2\sqrt{2n})\,\epsilon\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1.$$

In the first order analysis of the error we have

$$|\delta(z_j)| \le c_2\,\sqrt{2n}\,\epsilon\,\|\boldsymbol{h}\|_\infty + (c_1 + 2c_2\sqrt{2n})\,\epsilon\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1 \le (c_1 + 3c_2\sqrt{2n})\,\epsilon\,\|\boldsymbol{s}\|_1\|\boldsymbol{v}\|_1.$$

This means that the computed components of $\boldsymbol{z}$ are affected by an error slowly increasing with the size of the problem. It follows that the procedure for multiplying Toeplitz-like matrices by vectors is stable.

9 Analysis of the costs

In this section the multiplicative costs (16), (27), (30) and (53) are compared. Without loss of generality the bandwidths n and m are taken as powers of 2. To simplify the discussion we rewrite these theoretical costs in the form

$$c_k = \alpha_k\,\log N + \beta_k\,N,$$

where k indicates the method and its coefficients α_k and β_k are functions of n and m. It is evident that method `cr` outperforms the others when m is equal to n, but we are interested in comparing the costs for values of n greater than m, i.e. when the bandwidths are unbalanced.

We must note that values of n and N too "small" are not of interest, since the corresponding computational costs are not significant. Two other reasons suggest not to accept unquestioningly the conclusions we can draw from the analysis of the theoretical costs: first, the approximations made for π_n, which

method	k	α_k	β_k
`ge`	1	0	$105\,\pi_n + 50\pi_m + 27m^2$
`cr`	2	$16\pi_n + 14\pi_m + 14.5\,n^2$	$24\pi_n + 12\pi_m$
`mcr`	3	$145\pi_n + 84\pi_m + 108\,m^2$	$24\pi_n + 12\pi_m$
`st`	4	$90\pi_n + 70\pi_m + 29\,m^2$	$21.5\pi_n + 22\pi_m$

Table 5: Theoretical costs.

take into consideration only the terms in $n \log n$ and neglect the terms in n, hold only for large n, and second, the influence of the difference between the cost of the first step and the other steps of the transformation phase, lowers with increasing N.

Asymptotically with N, the cost depends mainly on β_k. Hence method `ge` has the greatest cost (even in the order as a function of m). For this reason it will be ignored in the following discussion.

Let $\rho = n/m$ be the ratio between the bandwidths. We expect that for finite fixed value of N the cost of `cr`, seen as a function of ρ, is a more steep function than the cost of both `mcr` and `st`, due to the presence in α_2 of a term in n^2, compared with terms in m^2 for the other two methods. We consider the cases $\rho = 2$, $\rho = 4$ and $\rho \geq 8$. For each case a figure shows the graphs of the costs of `cr`, `mcr` and `st` as functions of m, for the fixed value $N = 2^8$.

$\rho = 2$, mildly unbalanced (Figure 1): in this case we have $\beta_2 = \beta_3 < \beta_4$ and $\alpha_2 < \alpha_3$ for any m. Then we expect `cr` to outperform `mcr` for any m. A crossing between the graphs of `cr` and `st` (solid and dashed lines) suggests that the two methods have a different behaviour for small and large values of m and that `st` may outperform `cr` for large values of m.

$\rho = 4$, moderately unbalanced (Figure 2): in this case we have $\beta_2 = \beta_3 > \beta_4$ for large m and $\alpha_4 < \alpha_3$ for any m. The closeness of the graphs of `cr` and `mcr` (solid and dotted lines) suggests a substantial equivalence of the two methods, at least for small values of m. Moreover we expect `st` to outperforms both `cr` and `mcr` for large values of m.

$\rho \geq 8$, severely unbalanced (Figure 3): in this case we have $\beta_2 = \beta_3 > \beta_4$ and $\alpha_4 < \alpha_3 < \alpha_2$ for any m. We expect `st` to outperform both `cr` and `mcr` for any m.

Asymptotically with N, methods `cr` and `mcr` have a common behaviour. For $\rho = 2$ `st` is outperformed by `cr` and `mcr`, for $\rho = 4$ the three methods have the same behaviour, for $\rho \geq 8$ `st` slightly outperforms `cr` and `mcr`.

Summarizing, we expect that when the bandwidths are at least moderately unbalanced, the use of SMW formula leads to an improvement of the performance, i.e. `mcr` and `st` outperform `cr` for m sufficiently large. Moreover, in

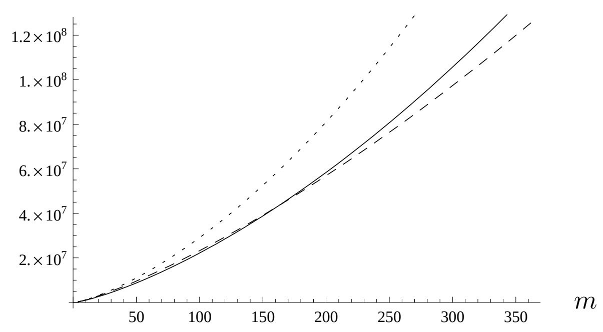


Figure 1: Graphs of the cost c_2 (method `cr`, solid line), c_3 (method `mcr`, dotted line), c_4 (method `st`, dashed line), for $\rho = 2$ and $N = 2^8$.

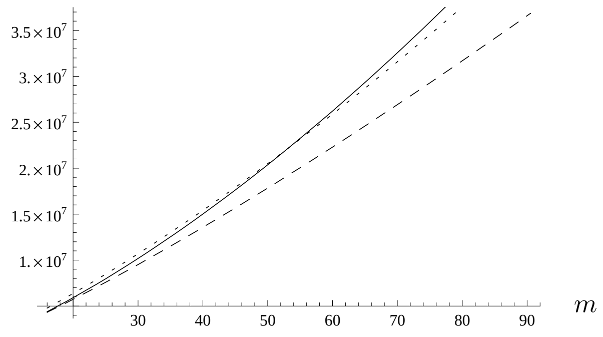


Figure 2: Graphs of the cost c_2 (method `cr`, solid line), c_3 (method `mcr`, dotted line), c_4 (method `st`, dashed line), for $\rho = 4$ and $N = 2^8$.

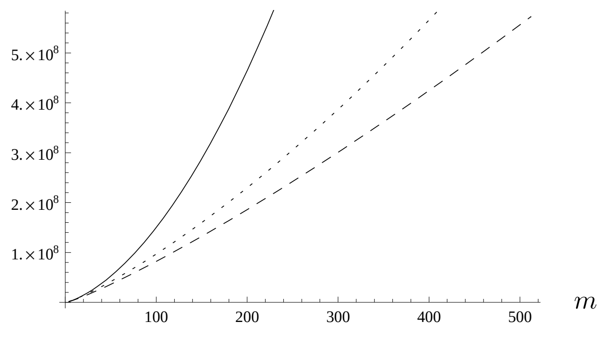


Figure 3: Graphs of the cost c_2 (method `cr`, solid line), c_3 (method `mcr`, dotted line), c_4 (method `st`, dashed line), for $\rho = 8$ and $N = 2^8$.

this case `st` appears to be always better than `mcr`. Also in the case of mildly unbalanced bandwidths `st` appears to be better than `cr` for m sufficiently large. However, there are cases where `cr` appears to be better than `st` for small values of m and this region shrinks for increasing values of ρ.

In next section an experimentation is carried out on methods `cr`, `mcr` and `st` to validate the previous results, investigating the costs also for small values of the parameters. This is interesting, since the approximations used to derive the theoretical multiplicative costs neglect terms linear in the dimensions and this approach may lead to inaccurate measures, especially in the case of small values of the parameters.

10 Numerical experiments

The experiments, conducted on a Intel Core Duo @ 3 GHz, 2GB RAM, have two aims: (1) to measure the errors affecting the computed solution, in order to analyze the stability properties of the algorithms, and (2) to compare the effective costs, in order to test if their behaviour agrees with the theoretical costs given in Section 9.

10.1 Stability tests

Different matrices of the form (2) have been considered:

1. entries generated by specific rules enforcing a decay behaviour of the diagonals,
2. positive entries randomly generated,
3. entries randomly generated, with mixed sign.

The left hand-side vector $\boldsymbol{b}$ has been computed from a randomly generated solution, to which the computed solution is compared.

For each bandwidth ratio $\rho = 2$, $\rho = 4$ and $\rho = 8$ the data depend on the following parameters:

1. the upper bandwidths m,
2. the number N of blocks,
3. the diagonal dominance factor $\delta = |a_0| / \sum_{i \neq 0} |a_i|$ which varies in the range $[0.2, 1.2]$. For $\delta > 1$ matrix A has point diagonal dominance (both by rows and columns).

For $\delta > 1$, when A has point diagonal dominance, most of the matrices of the first and second class satisfy block condition (55) in 1-norm, while for the matrices of the third class the condition is nearly never satisfied. For $\delta < 1$ some matrices of the first and second class still satisfy condition (55).

The conditioning of A depends heavily on δ. On average for the matrices of the first and second class, the growth of the condition number for increasing size is not significant when $\delta > 0.8$ and is slow when $\delta \in [0.4, 0.8]$. When $\delta < 0.4$ the matrix may become badly conditioned. The situation is much worse for the matrices of the third class, where the growth with increasing size of the condition number is slow for $\delta > 1$, but can be very high for $\delta < 0.8$. In the latter case the matrix may be so badly conditioned that no method (even the best software Gaussian elimination) can give an acceptable solution.

The stability tests have been performed on all the four methods and, for comparison, on a simple Gaussian elimination without any pivoting strategy. The latter test is particularly significant, because none of the four methods, which are intrinsically recursive, can implement a pivoting strategy.

The results of the tests depend on the diagonal dominance properties.

1. For $\delta > 1$ all the methods are stable and produce a solution affected by an error of the same magnitude of the machine precision ϵ of the arithmetic which is used.

2. For $\delta \leq 1$ we expect the error of the solution to increase with the condition number $\kappa(A)$ and with the parameter s, which measures the block diagonal dominance, defined in (55). We observe the following error behaviours corresponding to the three classes of problems.

2.a The tests on matrices of the first and second class not having point diagonal dominance show that the methods continue to be stable in regions $\delta \in [\overline{\delta}, 1]$, for suitable $\overline{\delta} < 1$, and that this happens also when there is not a block diagonally dominance. It is important to remark that the value of $\overline{\delta}$ for each problem is the same for all the method, included Gaussian elimination without pivoting, showing that the methods enjoy the same stability properties. In most cases the interval $[\overline{\delta}\,,\,1]$ stretches to $[0.6\,,\,1]$ and in some cases even to $[0.4\,,\,1]$.

2.b The tests on matrices of the third class not having point diagonal dominance show that all the methods produce solutions affected by errors much greater than the machine precision ϵ, suggesting instability, but we must note that in general these matrices have condition number $\kappa(A) > 1/\epsilon$.

10.2 Comparison of the effective costs

The effective cost of a method is measured in terms of the total number ω of operations performed, which reasonably represents the running time under the assumption that multiplications and additions consume roughly the same amount of CPU time. Due to the results of Section 9, method `ge` has not been considered in this part of the experimentation. Figures 4, 5 and 6 show the cost ω of `cr` (line with the $+$ mark), of `mcr` (line with the $*$ mark) and of `st` (line with the $\bullet$ mark) for $N = 2^8$.

These figures are to be compared with Figures 1-3, taking into account that ω measures, in addition to the higher order terms in multiplicative operations (which are considered in the theoretical costs) also the lower order terms in multiplicative operations and the additive operations (neglected in the theoretical costs). We point out that ω does not count the cost of the inversion of matrix B_0, initial step common to all the algorithms.

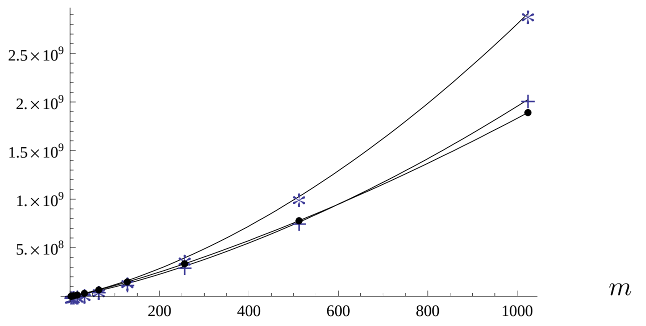


Figure 4: Graphs of the effective cost ω of `cr` (line with the $+$ mark), of `mcr` (line with the $*$ mark) and of `st` (line with the $\bullet$ mark) for $\rho = 2$ and $N = 2^8$.

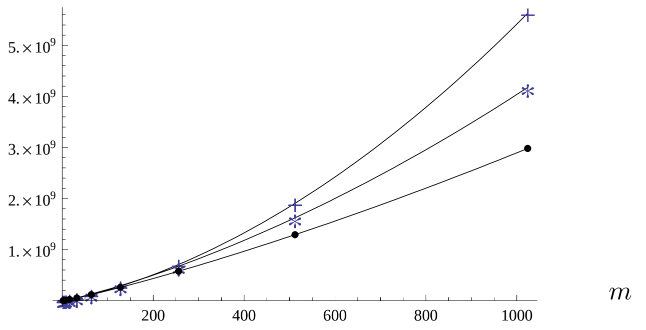


Figure 5: Graphs of the effective cost ω of `cr` (line with the $+$ mark), of `mcr` (line with the $*$ mark) and of `st` (line with the $\bullet$ mark) for $\rho = 4$ and $N = 2^8$.

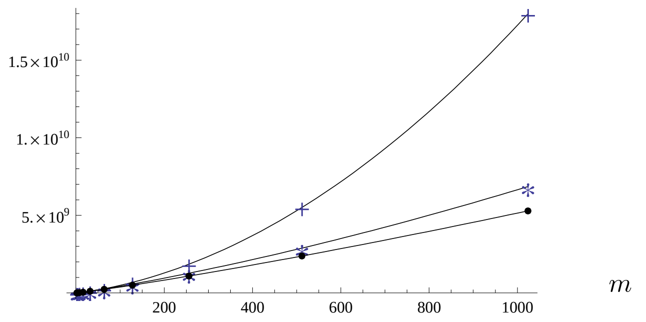


Figure 6: Graphs of the effective cost ω of `cr` (line with the $+$ mark), of `mcr` (line with the $*$ mark) and of `st` (line with the $\bullet$ mark) for $\rho = 8$ and $N = 2^8$.

It is evident that considering the effective cost ω does not alter significantly the ranking of the methods derived from the theoretical estimates for large m. By a more accurate analysis of Figure 4-6 we list the ranking of the methods for small values of m and $N = 2^8$.

For $\rho = 2$ we have

$$\begin{aligned} &\omega_{\texttt{cr}} < \omega_{\texttt{st}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad m \leq 2^8, \\ &\omega_{\texttt{st}} < \omega_{\texttt{cr}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad m \geq 2^9. \end{aligned}$$

For $\rho = 4$ we have

$$\begin{aligned} &\omega_{\texttt{cr}} < \omega_{\texttt{st}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad m \leq 2^5, \\ &\omega_{\texttt{st}} < \omega_{\texttt{cr}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad 2^6 \leq m \leq 2^7, \\ &\omega_{\texttt{st}} < \omega_{\texttt{mcr}} < \omega_{\texttt{cr}}, \quad \text{for} \quad m \geq 2^8. \end{aligned}$$

For $\rho = 8$ we have

$$\begin{aligned} &\omega_{\texttt{cr}} < \omega_{\texttt{st}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad m \leq 2^2, \\ &\omega_{\texttt{st}} < \omega_{\texttt{cr}} < \omega_{\texttt{mcr}}, \quad \text{for} \quad 2^3 \leq m \leq 2^5, \\ &\omega_{\texttt{st}} < \omega_{\texttt{mcr}} < \omega_{\texttt{cr}}, \quad \text{for} \quad m \geq 2^6. \end{aligned}$$

For $\rho > 8$ the methods keep the same ranking as for $\rho = 8$, but with a shift toward left of the intersection points. For asymptotically large values of N, when the cost of the substitution phase dominates, methods `cr` and `mcr` have the same behaviour and `st` has a slightly better behaviour only for $\rho \geq 8$.

11 Conclusions

The three methods `cr`, `mcr` and `st` (having excluded `ge` for its higher cost) have shown to be effective from the point of view of both the computational costs and the stability properties. Their running times confirm the rankings predicted from the theoretical cost measures, pointing out the efficacy of the employment of SMW formula in the case of unbalanced bandwidths.

The costs we have compared depend on the particular implementation of the algorithms. Other implementations may give different results, and the ranking may be altered for finite values of the parameters. But substantially the ranking depends on two issues:

1. the cost of the substitution phase can be considered equivalent for small values of ρ and favorable to `st` for high values of ρ,

2. the cost of the recursive phase depends mainly on the sizes of the matrices to be inverted and the ranking advantages `st` which inverts m sized matrices, followed by `mcr` which inverts $2m$ sized matrices, followed by `cr` which inverts n sized matrices. A higher number of matrix by vector products may alter the ranking only for small sizes.

References

[1] M. Arioli, H. Munthe-Kaas, L. Valdettaro, Componentwise error analysis for FFT's with applications to fast Helmholtz solvers, *Numer. Algorithms*, 12, (1996) 65-88.

[2] A. D. Bini, B. Meini, Inverting Block Toeplitz Matrices in Block Hessenberg Form by means of Displacement Operators: Application to Queueing Problems, *Linear Algebra Appl.*, 272, (1998) 1-16.

[3] A. D. Bini, B. Meini, Effective methods for solving banded Toeplitz systems, *SIAM J. Matrix Anal. Appl.*, 20, (1999) 700-719.

[4] A. D. Bini, V. Pan, Matrix and Polynomial Computations, Vol.1: Fundamental Algorithms, Birkhauser, Boston, 1994.

[5] R. R. Bitmead, B. D. O. Anderson, Asymptotically fast solution of Toeplitz and related systems of linear equations, *Linear Algebra Appl.*, 34 (1980) 103-116.

[6] A.W. Bojanczyk, R.P. Brent, F.R. de Hoog, D.R. Sweet, On the stability of the Bareiss and related Toeplitz factorization algorithms, *SIAM J. Matrix Anal. Appl.*, 16 (1999), 40-57.

[7] T.F. Chan, P.C. Hansen, A look-ahead Levinson algorithm for general Toeplitz systems, *IEEE Trans. Signal Process.*, 40, (1992), 1079-1090.

[8] R.H. Chan, M.K. Ng, Conjugate gradient methods for Toeplitz systems, *SIAM Rev.*, 38 (1996), 427-482.

[9] R.H. Chan, M.K. Ng, Scientific applications of iterative Toeplitz solvers, *Calcolo*, 33 (1996), 249-267.

[10] G. Cybenko, The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations, *SIAM J. Sci. Stat. Comput.* 1, (1980) 303-319.

[11] P. Favati, G. Lotti, O. Menchi, *Solving banded Toeplitz systems*, Technical Report IIT TR-04/2005.

[12] I. Gohberg, T. Kailath, I. Koltracht, Efficient Solution of Linear Systems of Equations with Recursive Structure, *Linear Algebra Appl.*, 80, (1986) 81-113.

[13] G.H. Golub, C.F. Van Loan, Matrix Computation, The Johns Hopkins University Press, Baltimore, Ma, 1996.

[14] N.J. Higham, Accuracy and stabilty of numerical algorithms, SIAM, 1961.

[15] T. Kailath, A. H. Sayed, Displacement structure: theory and applications, *SIAM Rev.*, 37, (1995) 297-386.

[16] T. Kailath, A. Viera, M. Morf, Inverse of Toeplitz operators, innovations and orthogonal polynomials, *SIAM Rev.*, 20, (1978) 106-119.

[17] G. Lotti, A note on the solution of not balanced banded Toeplitz systems, *Numer. Linear Algebra Appl.*, 14 (2007) 645-657.

[18] U. von Matt, G.W. Stewart, Rounding errors in solving block Hessenberg systems, *Math. Comp.*, 65, (1996) 115-135.

[19] B.R. Musicus, Levinson and fast Cholesky algorithms for Toeplitz and almost Toeplitz matrices, Research Laboratory of Electronics Tec. Rep. 538, MIT, Cambridge, MA.

[20] S. Serra Capizzano, Toeplitz preconditioner constructed from linear approximation processes, *SIAM J. Matrix Anal. Appl.*, 20 (1999), 446-465.

[21] H.V. Sorensen, D.L. Jones, M.T. Heideman, C.S. Burrus, Real-Valued Fast Fourier Transform Algorithms, *IEEE Trans. Acoust. Speech, Signal Process.*, ASSP-35, (1987), 849-863.

[22] G.W. Stewart, On the solution of block Hessenberg systems, *Numer. Linear Algebra Appl.*, 2, 1995 287-296.

[23] W.F. Trench, An Algorithm for the Inversion of Finite Toeplitz Matrices, *SIAM J. Appl. Math.* 12, (1964) 515-522.

[24] P. Yalamov, V. Pavlov, Stability of the bock cyclic reduction, *Linear Algebra Appl.*, 249, (1996) 341-358.

[25] E.L. Yip, A note on the stability of solving a rank-p modification of a linear system by the Sherman-Morrison-Woodbury formula, *SIAM J. Sci. Stat. Comput.* 7, (19860) 507-513.