

**The numerically stable algorithm
for quadratic programs
with simple upper bounds**

by

GRAŻYNA KRYŃSKA

Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warszawa, Poland

In the paper [4] all the details of the PSUBQ algorithm are described, its convergence is examined and compared with other algorithms from the same class of methods of feasible directions, but (except for a short remark) numerical features of the proposed algorithm are not discussed. So now, these considerations are completed and four versions of the PSUBQ algorithm are presented making use of four numerically stable methods of matrix factorization.

1. Introduction

In the paper [4], the primal algorithm using conjugate directions for quadratic programming problems with simple upper bounds, called the PSUBQ algorithm, is described in all details. The algorithm belongs to the class of methods of feasible directions. At each iteration a feasible point (a successive approximation of the optimal solution) and some system of equations (for which this point is a solution) are found. A set of directions that are determined by rows of the inverse of the basis matrix connected with this point is also available. A feasible direction is chosen just among these directions. The updating scheme (2.8)–(2.9) [4] for rows of the inverse of the basis matrix is adequate for the theoretical considerations, but it does not assure numerical stability of the PSUBQ algorithm (see [8]). So now four versions of the PSUBQ algorithm are presented which use four numerically stable methods of matrix factorization: Gaussian elimination, the method of Bartels and Golub, the method of Tomlin and the method of Forrest and Tomlin. All these methods of matrix factorization must use partial or complete pivoting to ensure numerical stability.

2. Formulation of the problem; notations

The PSUBQ algorithm solves the problem:

$$\min \{f(x): x \in X\}. \quad (2.1)$$

where $f(x) = d^T x + \frac{1}{2} x^T D x$ is a convex quadratic objective function, $X = \{x \in E^n: Ax = b \wedge \beta \leq x \leq \alpha\}$ is a feasible set and $d, \beta, \alpha \in E^n$, $b \in E^m$, $A \in E^{m \times n}$, $D = D^T \in E^{n \times n}$. Notice that X is a compact polyhedron, and hence the finite optimal value of the objective function is reached in X .

The PSUBQ algorithm constructs a sequence $\{x^k\}$ of feasible points such that $f(x^{k+1}) \leq f(x^k)$. Let $P^k = (p_1^k, \dots, p_n^k)$ be the basis matrix associated with the feasible point x^k . Its columns p_i^k for $i = 1, \dots, n$ are associated with n linearly independent and active constraints at x^k . Let $(P^k)^{-1}$ denote the inverse of P^k and π_j^{kT} be its j -th row, for $j = 1, \dots, n$. Let $N = \{1, \dots, n\}$. We define the following sets of indices:

$$R_\beta^k = \{i \in N: (\exists j_i \in N) p_i^k = e_{j_i} \wedge x_{j_i}^k = \beta_{j_i}\},$$

$$R_\alpha^k = \{i \in N: (\exists j_i \in N) p_i^k = e_{j_i} \wedge x_{j_i}^k = \alpha_{j_i}\},$$

$$R_A^k = \{i \in N: p_i^k = A_i \wedge A_i^T x^k = b_i\} = \{1, \dots, m\}$$

and $IW^k = N \setminus (R_\beta^k \cup R_\alpha^k \cup R_A^k)$, where A_i^T denotes the i -th row of the matrix A .

Directional derivatives w_i^k of the objective function $f(x)$ at x^k in directions π_i^k for $i = 1, \dots, n$ are equal to $\pi_i^{kT} z^k$, where $z^k = d + D x^k$ denotes the value of the gradient of the objective function at x^k . Hence $w^k = (P^k)^{-1} z^k$ is a vector of directional derivatives of the objective function at x^k in directions π_i^k for $i = 1, \dots, n$.

In Section 2 [4] we show that if one of the following cases:

- (i) $(\exists_{r \in R_\beta^k}) w_r^k < 0$;
- (ii) $(\exists_{r \in R_\alpha^k}) w_r^k > 0$;
- (iii) $(\exists_{r \in IW^k}) w_r^k \neq 0$

occurs then x^k is not the optimal solution for the problem (2.1) and we can decrease the value of the objective function moving along the direction π_r^k to the new feasible point x^{k+1} such, that $f(x^{k+1}) < f(x^k)$.

Let us define sets of indices:

$$K_\beta^k = \{i \in R_\beta^k: w_i^k < 0\}, \quad K_\alpha^k = \{i \in R_\alpha^k: w_i^k > 0\},$$

$$K_I^k = \{i \in IW^k: w_i^k \neq 0\} \text{ and } K^k = K_\beta^k \cup K_\alpha^k \cup K_I^k.$$

Optimality criterion for the PSUBQ algorithm is formulated as follows (see Theorem 4.1 [4]):

THEOREM. If $K^k = \emptyset$, then x^k is an optimal solution for the problem (2.1).

We are going to calculate and compare costs of one iteration for the PSUBQ algorithm and its four numerically stable versions. Accordingly, let N_{add} , N_{mult} and N_{div} denote a number of additions, multiplications and divisions, respectively.

3. The PSUBQ algorithm

STEP 1. (see Section 3 [4]). Establish the initial feasible point x^0 using the SUB method [3]. Construct the basis matrix P^0 and its inverse $(P^0)^{-1}$. Compute $z^0 = d + Dx^0$. Define R_β^0 , R_α^0 and put $IW^0 = \emptyset$. Put $k = 0$.

STEP 2. Compute $w_i^k = \pi_i^{kT} z^k$ for all $i \geq m+1$.

STEP 3. If $K^k = \emptyset$ then x^k is the optimal solution, STOP. If $K^k \neq \emptyset$ then choose the feasible direction, i.e. an index $r \geq m+1$ according to the rule A or B, where:

Case I. If $K_1^k \neq \emptyset$ then r corresponds to

$$(A) \quad |w_r^k| = \max_{i \in K_1^k} |w_i^k|$$

Case II. If $K_1^k = \emptyset$ then r corresponds to

$$|w_r^k| = \max_{i \in K_1^k \cup K_2^k} |w_i^k|,$$

and

$$(B) \quad r \text{ corresponds to } |w_r^k| = \max_{i \in K^k} |w_i^k|.$$

Remove the column p_r^k from the basis matrix P^k .

STEP 4. Compute $q^k = D\pi_r^k$. Compute

$$\theta_0^k = \begin{cases} \text{sign } w_r^k & \text{if } q^k = 0 \\ \frac{w_r^k}{q^{kT} \pi_r^k} & \text{if } q^k \neq 0. \end{cases}$$

$\theta_0^k \pi_r^k$ and t_0^k (according to (2.4) [4]).

If $q^k \neq 0$ and $t_0^k > 1$, then $\theta^k = \theta_0^k$. Go to Step 5.

If $q^k \neq 0$ and $0 \leq t_0^k \leq 1$ or if $q^k = 0$, then $\theta^k = t_0^k \theta_0^k$. Go to step 6.

STEP 5. Put $p_r^{k+1} = q^k$. Go to Step 7.

STEP 6. Put $p_r^{k+1} = e_i$, where e_i is found from (2.5)–(2.6) [4].

STEP 7. Put $x^{k+1} = x^k - \theta^k \pi_r^k$, $z^{k+1} = z^k - \theta^k q^k$. Update π_i^k for all $i \geq m+1$ according to (2.8)–(2.9) [4]. i.e.:

$$\begin{cases} \pi_r^{k+1} = \frac{1}{(p_r^{k+1})^T \pi_r^k} \pi_r^k \\ \pi_i^{k+1} = \pi_i^k - \frac{(p_r^{k+1})^T \pi_i^k}{(p_r^{k+1})^T \pi_r^k} \pi_r^k & \text{for } i \neq r \end{cases}$$

Update R_β^k , R_α^k and IW^k according to (2.11), (2.12) and (2.13) [4], respectively. Increase k by 1. Go to Step 2.

REMARK 3.1. The updating scheme (2.8)–(2.9) [4], presented also at Step 7, requires performing $2n(n-m-1)+m$ additions, $2n(n-m)$ multiplications and 1 division.

4. A version of the PSUBQ algorithm using Gaussian elimination

Let us recall that solving systems of linear equations and inverting matrices by Gaussian elimination is based on the triangular decomposition of the square matrix (see [7]). As it was already mentioned, to assure numerical stability we use the strategy of partial or complete pivoting. The version of the PSUBQ algorithm with Gaussian elimination uses the upper-triangular matrix U^k and the inverse $(L^k)^{-1}$ of the lower-triangular matrix L^k such that $L^k U^k = P^k$ (instead of the matrices P^k and $(P^k)^{-1}$), and these are the matrices that must be updated. Notice that in consequence the search direction π_r^k is not known explicitly therefore it has to be computed in Step 3.

STEP 1. Establish the initial feasible point x^0 using the numerically stable modification (see [5]) of the SUB method [3]. Construct the basis matrix P^0 and obtain its $L^0 U^0$ factorization using Gaussian elimination, where lower-triangular matrix L^0 and upper-triangular matrix U^0 are such that $L^0 U^0 = P^0$ with its rows permuted. Find $(L^0)^{-1}$ by solving n systems of linear equations:

$$L^0 y_i = e_i \quad \text{for } i = 1, \dots, n.$$

Compute $z^0 = d + Dx^0$, define sets R_β^0 , R_α^0 and put $IW^0 = \emptyset$. Put $k = 0$.

STEP 2. Compute $\bar{x} = \overline{(L^k)^{-1}} z^k$, where $\overline{(L^k)^{-1}}$ is a submatrix of $(L^k)^{-1}$ formed by deleting its first m rows. Then solve the system of $n-m$ equations $\bar{U}^k \bar{w}^k = \bar{x}$, where \bar{U}^k is a submatrix of U^k formed by deleting its first m

rows and columns. The vector $\bar{w}^k = [w_{m+1}^k, \dots, w_n^k]^T$ is the solution of this system.

STEP 3. Define sets of indices: K_β^k , K_α^k , K_l^k and K^k . If $K^k = \emptyset$ then x^k is the optimal solution, STOP. If $K^k \neq \emptyset$ then choose an index $r \geq m+1$ according to the rule A or B and compute the new search direction $\pi_r^{kT} = [\pi_{r1}^k, \dots, \pi_{rn}^k]$ solving n systems of $n-r+1$ equations:

$$\tilde{U}^k \begin{bmatrix} \pi_{ri}^k \\ \vdots \\ \pi_{ni}^k \end{bmatrix} = (L^k)_{\tilde{i}}^{-1} \quad \text{for } i = 1, \dots, n.$$

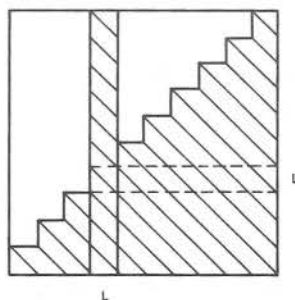
where \tilde{U}^k is a submatrix of U^k formed by deleting its first $r-1$ rows and columns, $(L^k)_{\tilde{i}}^{-1}$ is submatrix of $(L^k)^{-1}$ formed by deleting its first $r-1$ rows and $(L^k)_{\tilde{i}}^{-1}$ is the i -th column of $(L^k)_{\tilde{i}}^{-1}$.

STEP 4. Compute $q^k = D\pi_r^k$, θ_0^k (according to (4.3) [4]), $\theta_0^k \pi_r^k$ and t_0^k (according to (2.4) [4]). If $q^k \neq 0$ and $t_0^k > 1$ then $\theta^k = \theta_0^k$. Go to Step 5. If $q^k \neq 0$ and $0 \leq t_0^k \leq 1$ or if $q^k = 0$ then $\theta^k = t_0^k \theta_0^k$. Go to Step 6.

STEP 5. Put $p_r^{k+1} = q^k$. Go to Step 7.

STEP 6. Put $p_r^{k+1} = e_l$, where l is found by (2.5)–(2.6) [4].

STEP 7. Compute $Z_k = (L^k)^{-1} p_r^{k+1}$. Next, form the matrix $H^{k+1} = (U_1^k, \dots, U_{r-1}^k, Z_k, U_{r+1}^k, \dots, U_n^k) = (h_{ij})_{i,j=1,\dots,n}$, where U_i^k for $i = 1, \dots, n$ denotes the i -th column of $U^k = (L^k)^{-1} P^k$. Hence H^{k+1} is of the following form:



(4.1)

and its nonzero elements can appear only at the lined area.

STEP 8. Put $x^{k+1} = x^k - \theta^k \pi_r^k$ and $z^{k+1} = z^k - \theta^k q^k$. Using Gaussian elimination transform H^{k+1} to the upper-triangular matrix $U^{k+1} = \Gamma_{n-1}^k \phi_{n-1}^k \dots \Gamma_r^k \phi_r^k H^{k+1}$, where for $i = r, \dots, n-1$:

$$\Gamma_i^k = \begin{array}{|ccc|} \hline & i & \\ \hline 1 & & \\ \hline & 1 & \\ \hline & -\frac{h_{i+1,j}^{(i-1)}}{h_{pi}^{(i-1)}} & 1 \\ \hline & \vdots & \\ \hline & -\frac{h_{ii}^{(i-1)}}{h_{pi}^{(i-1)}} & \\ \hline & \vdots & \\ \hline & -\frac{h_{ni}^{(i-1)}}{h_{pi}^{(i-1)}} & \\ \hline & & \\ \hline \end{array} \quad (4.2)$$

is an elementary triangular matrix when $h_{pi}^{(i-1)}$ was chosen as the pivot, $(h_{ij}^{(i)})_{i,j=1,\dots,n} = \Gamma_i^k \varphi_l^k \dots \Gamma_r^k \varphi_r^k H^{k+1}$, $h_{ij}^{(i-1)} = h_{ij}$ for all i, j and φ_i^k is a permutation matrix exchanging the i -th row with the p -th one. Compute the new matrix $(L^{k+1})^{-1} = \Gamma_{n-1}^k \varphi_{n-1}^k \dots \Gamma_r^k \varphi_r^k (L^k)^{-1}$ (let us notice here that the matrix $(L^{k+1})^{-1}$ is not usually the lower-triangular one). Update R_β^k , R_α^k and IW^k according to (2.11), (2.12) and (2.13) of [4], respectively. Increase k by 1. Go to Step 2.

REMARK 4.1. To transform H^{k+1} to the upper-triangular matrix U^{k+1} using Gaussian elimination and to update $(L^k)^{-1}$ we must perform $N_{\text{add}} = N_{\text{mult}} = \frac{(n-r)(n-r+1)(5n-2r+1)}{6}$ and $N_{\text{div}} = \frac{(n-r+1)(n-r)}{2}$.

5. A version of the PSUBQ algorithm using the method of Bartels and Golub

We point out here the differences only between the version of the PSUBQ algorithm that uses the method of Bartels and Golub [1, 5] and the one described in Section 4.

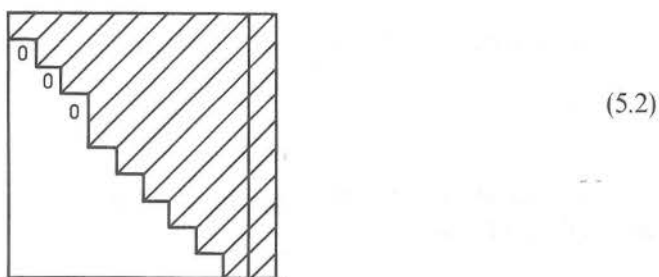
The first difference appears at the 5-th and 6-th steps, since the new column q^k or e_l enters the basis matrix not instead of the r -th column, but instead of the n -th one. So let

$$P_n^{k+1} = \begin{cases} q^k & \text{if } q^k \neq 0 \text{ and } t_0^k > 1 \\ e_l & \text{if } q^k \neq 0 \text{ and } 0 \leq t_0^k \leq 1 \text{ or if } q^k = 0. \end{cases} \quad (5.1)$$

The new basis matrix is then

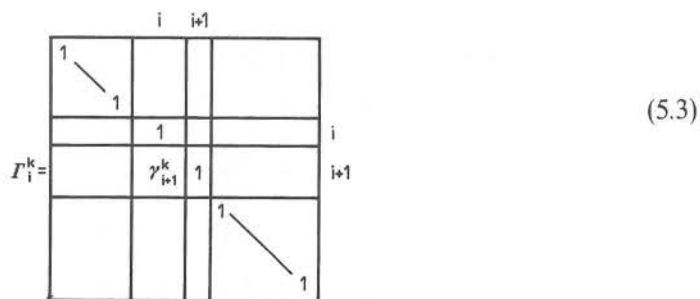
$$\mathcal{P}^{k+1} = (p_1^{k+1} = p_1^k, \dots, p_{r-1}^{k+1} = p_{r-1}^k, p_r^{k+1} = p_{r+1}^k, \dots, p_{n-1}^{k+1} = p_n^k, p_n^{k+1}).$$

At Step 7 we compute now $Z_k = (L^k)^{-1} p_n^{k+1}$ and instead of H^{k+1} we construct an upper Hessenberg matrix $\mathcal{H}^{k+1} = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, U_n^k, Z_k) = (h_{ij})_{i,j=1,\dots,n}$, which is of the form:



Then Step 8 is as follows:

STEP 8a. Put $x^{k+1} = x^k - \theta^k \pi_r^k$, $z^{k+1} = z^k - \theta^k q^k$. Using the method of Bartels and Golub transform \mathcal{H}^{k+1} to the upper-triangular matrix $U^{k+1} = \Gamma_{n-1}^k \varphi_{n-1}^k \dots \Gamma_r^k \varphi_r^k \mathcal{H}^{k+1}$, where for $i = r, \dots, n-1$: φ_i^k is either an identity matrix (if the pivot is a diagonal element) or is a permutation matrix exchanging the i -th row with the $(i+1)$ -th one (if the pivot is a sub-diagonal element) and



is an elementary triangular matrix such that

$$\gamma_{i+1,i}^k = \begin{cases} -\frac{h_{i+1,i}^{(i-1)}}{h_{ii}^{(i-1)}} & \text{if } |h_{ii}^{(i-1)}| \geq |h_{i+1,i}^{(i-1)}| \\ -\frac{h_{ii}^{(i-1)}}{h_{i+1,i}^{(i-1)}} & \text{if } |h_{ii}^{(i-1)}| < |h_{i+1,i}^{(i-1)}| \end{cases}$$

and $(h_{ij}^{(l)})_{i,j=1,\dots,n} = \Gamma_l^k \varphi_l^k \dots \Gamma_r^k \varphi_r^k \mathcal{H}^{k+1}$, $h_{ij}^{(r-1)} = h_{ij}$ for all i, j .

Compute the new matrix $(L^{k+1})^{-1} = \Gamma_{n-1}^k \varphi_{n-1}^k \dots \Gamma_r^k \varphi_r^k (L^k)^{-1}$ (notice that $(L^{k+1})^{-1}$ is not usually the lower-triangular matrix). Update R_β^k , R_α^k and IW^k according to (2.11), (2.12) and (2.13) of [4]. Increase k by 1. Go to Step 2.

REMARK 5.1. To transform the upper Hessenberg matrix \mathcal{H}^{k+1} to the upper-triangular matrix U^{k+1} using the method of Bartels and Golub and to update $(L^k)^{-1}$ we must perform $N_{\text{add}} = N_{\text{mult}} = \frac{(n-r)(3n-r+1)}{2}$ and $N_{\text{div}} = n-r$.

6. A version of the PSUBQ algorithm using the method of Tomlin

The third version of the PSUBQ algorithm uses the method of Tomlin [6, 5] with partial or complete pivoting. It coincides with the version using Gaussian elimination up to Step 8. Let it be as previously: $Z_k = (L^k)^{-1} p_r^{k+1}$ and $H^{k+1} = (U_1^k, \dots, U_{r-1}^k, Z_k, U_{r+1}^k, \dots, U_n^k) = (h_{ij})_{i,j=1,\dots,n}$, where U_i^k is the i -th column of U^k and H^{k+1} is of the form (4.1). Let us notice that $H^{k+1} = U^k (I - e_r e_r^T) + Z_k e_r^T$ and $Z_k = U^k y^k$, where $y^k = (P^k)^{-1} p_r^{k+1}$. Then $H^{k+1} = U^k (I - e_r e_r^T + y^k e_r^T) = U^k E_r^{-1}$, where $E_r^{-1} = I - e_r e_r^T + y^k e_r^T$ and E_r is an elementary matrix of Gauss-Jordan method defined by the vector y^k and the r -th pivot, i.e.:

$$E_r = \begin{array}{|ccc|} \hline 1 & -\frac{y_1^k}{y_r^k} & \\ & \frac{1}{y_r^k} & \\ & \vdots & \\ & -\frac{y_n^k}{y_r^k} & \\ \hline & & 1 \\ \hline \end{array} \quad (6.1)$$

We describe now how to transform the matrix H^{k+1} to an upper-triangular one by the method of Tomlin. First we must find the pivot among four elements of the matrix H^{k+1} : h_{rr} , $h_{r+1,r}$, h_{rn} , $h_{r+1,n}$. Depending on the pivot we choose one of four algorithms: I, II, III or IV. If h_{rr} was chosen as the pivot then the method of Tomlin with partial or complete pivoting is just the original method of Tomlin [6] and is carried out as Algorithm I:

STEP 1. Find $c^k = [0, \dots, 0, c_r, \dots, c_n]^T$ by solving the system:

$$c^{kT} U^k = e_r^T H^{k+1}. \tag{6.2}$$

Hence $c^{kT} = e_r^T H^{k+1} (U^k)^{-1}$.

STEP 2. Construct the matrix $C_k = I - e_r c^{kT}$ and obtain the matrix $H'^{(k+1)} = C_k H^{k+1} = (h'_{ij})_{i,j=1,\dots,n}$. Notice that $C_k H^{k+1} = H^{k+1} - e_r c^{kT} H^{k+1} = H^{k+1} - e_r e_r^T H^{k+1} (U^k)^{-1} U^k E_r^{-1} = H^{k+1} - e_r e_r^T H^{k+1} E_r^{-1}$. Hence $h'_{ij} = h_{ij}$ for all j and for all $i \neq r$, $h'_{rj} = 0$ for all $j \neq r$, $h'_{rr} = h_{rr} - \sum_{i=r}^n h_{ri} y_i^k = (C_k Z_k)_r = h_{rr} - \sum_{i=r}^n c_i h_{ir}$.

STEP 3. Denote the r -th column of $H'^{(k+1)}$ by $H'_r{}^{(k+1)}$. Construct the matrix $T^k = I + (H'_r{}^{(k+1)} - e_r) e_r^T$, i.e.

$$T^k = \begin{array}{|c|c|c|} \hline 1 & h'_{1r} & \\ \hline & \vdots & \\ \hline & h'_{rr} & \\ \hline & \vdots & 1 \\ \hline & h'_{nr} & \\ \hline \end{array}$$

Then $(T^k)^{-1}$ is of the form:

$$(T^k)^{-1} = \begin{array}{|c|c|c|} \hline 1 & \frac{h'_{1r}}{h'_{rr}} & \\ \hline & \frac{1}{h'_{rr}} & \\ \hline & \vdots & 1 \\ \hline & \frac{h'_{nr}}{h'_{rr}} & \\ \hline \end{array} \tag{6.3}$$

and $H''^{(k+1)} = (T^k)^{-1} H'^{(k+1)} = (T^k)^{-1} C_k H^{k+1} = (h''_{ij})_{i,j=1,\dots,n}$ is such an upper-triangular matrix that $h''_{rr} = 1$, $h''_{rj} = 0$ for $j \geq r+1$, $h''_{ir} = 0$ for $i \neq r$ and $h''_{ij} = h_{ij}$ for all $i \neq r, j \neq r$.

and for all $i \neq r+1$, $\bar{h}_{r+1,j} = 0$ for all $j \neq r$, $\bar{h}_{r+1,r} = -\delta_r y_r^k = h_{r+1,r} - \bar{c}_r h_{rr} - \sum_{i=r+2}^n \bar{c}_i h_{ir}$.

STEP 3. Permute columns and rows of \bar{H}^{k+1} to obtain the upper-triangular matrix $\varphi_{r+1,n} \bar{H}^{k+1} \varphi_{rn}^T$, where φ_{rn}^T is defined by (6.5) for $i=r$, $j=n$ and $\varphi_{r+1,n}$ is a permutation matrix defined by (6.4) for $i=r+1$, $j=n$.

If h_{rn} was chosen as the pivot then Algorithm III is realized:

STEP 1. Find $\bar{c}^k = [0, \dots, 0, \bar{c}_{r+1}, \dots, \bar{c}_n]^T$ by solving the system:

$$\bar{c}^{kT} U^k = u_r^k, \quad (6.7)$$

where $u_r^k = e_r U^k - U_{rr}^k e_r^T E_n = \left[0, \dots, 0, U_{r,r+1}^k, \dots, U_{r,n-1}^k, U_{rn}^k + \frac{U_{rr}^k y_r^k}{y_n^k} \right]$ and E_n is an elementary matrix of Gauss-Jordan method defined by the vector y^k and the n -th pivot, i.e.

$$E_n = \begin{array}{|c|c|} \hline \begin{array}{cc} 1 & \\ & 0 \\ & & \ddots \\ & & & 1 \end{array} & \begin{array}{c} -\frac{y_1}{y_n^k} \\ \vdots \\ -\frac{y_{n-1}}{y_n^k} \end{array} \\ \hline \begin{array}{c} 0 \end{array} & \begin{array}{c} \frac{1}{y_n^k} \end{array} \\ \hline \end{array} \quad (6.8)$$

Hence $\bar{c}^{kT} = u_r^k (U^k)^{-1} = e_r^T - U_{rr}^k e_r^T E_n (U^k)^{-1}$.

STEP 2. Construct the matrix $\bar{C}_k = I - e_r \bar{c}^{kT} = I - e_r e_r^T + U_{rr}^k e_r e_r^T E_n (U^k)^{-1}$ and obtain the matrix $\bar{H}^{k+1} = \bar{C}_k H^{k+1} = (\bar{h}_{ij})_{i,j=1,\dots,n}$. Notice that $\bar{C}_k H^{k+1} = (I - e_r e_r^T + U_{rr}^k e_r e_r^T E_n (U^k)^{-1}) H^{k+1} = (I - e_r e_r^T) H^{k+1} + U_{rr}^k e_r e_r^T E_n (U^k)^{-1} U^k E_r^{-1} = (I - e_r e_r^T) H^{k+1} + U_{rr}^k e_r e_r^T E_n E_r^{-1}$. Hence, $\bar{h}_{ij} = h_{ij}$ for all j and for all $i \neq r$, $\bar{h}_{r,j} = 0$ for all $j \neq n$, $\bar{h}_{rn} = -\frac{U_{rr}^k y_r^k}{y_n^k} = h_{rn} - \sum_{i=r+1}^n \bar{c}_i h_{in}$.

STEP 3. Permute rows of \bar{H}^{k+1} to obtain the upper-triangular matrix $\varphi_{rn} \bar{H}^{k+1}$, where φ_{rn} is the permutation matrix defined by (6.4) for $i=r$, $j=n$.

If $h_{r+1,n}$ was chosen as the pivot then Algorithm IV is performed:

STEP 1. Find $\bar{c}^k = [\bar{c}_1, \dots, \bar{c}_n]^T$ by solving the system:

$$\bar{c}^{kT} \bar{W}_k = v_{r+1}^k, \quad (6.9)$$

where $U_{r+1}^k = e_{r+1}^T U^k E_n^{-1} = [0, \dots, 0, U_{r+1,r+1}^k, \dots, U_{r+1,n-1}^k, (Z_k)_{r+1}]$. $\bar{W}_k = U^k E_n^{-1} (I - e_r e_r^T) + e_{r+1} e_r^T = (U_1^k, \dots, U_{r-1}^k, e_{r+1}, U_{r+1}^k, \dots, U_{n-1}^k, Z_k)$ and $E_n^{-1} = I - e_n e_n^T + y_n^k e_n^T$.

Notice that $\bar{c}_1 = \dots = \bar{c}_{r-1} = 0$, $\bar{c}_{r+1} = 0$ and $\bar{c}_r = \frac{U_{r+1,r+1}^k}{U_{r,r+1}^k}$ so far as $U_{r,r+1}^k \neq 0$. Moreover, the system (6.9) is equivalent to the system $\bar{c}^{kT} U^k E_n^{-1} = e_{r+1}^T U^k E_n^{-1} + \delta_r e_r^T$, where

$$\delta_r = \frac{U_{rr}^k U_{r+1,r+1}^k}{U_{r,r+1}^k}. \text{ Hence } \bar{c}^{kT} = e_{r+1}^T + \delta_r e_r^T E_n (U^k)^{-1}.$$

STEP 2. Construct the matrix $\bar{C}_k = I - e_{r+1} \bar{c}^{kT} = I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T E_n (U^k)^{-1}$ and obtain the matrix $\bar{H}^{k+1} = \bar{C}_k H^{k+1} = (\bar{h}_{ij})_{i,j=1,\dots,n}$. Notice that $\bar{C}_k H^{k+1} = (I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T E_n (U^k)^{-1}) H^{k+1} = (I - e_{r+1} e_{r+1}^T) H^{k+1} - \delta_r e_{r+1} e_r^T E_n (U^k)^{-1} U^k E_r^{-1} = (I - e_{r+1} e_{r+1}^T) H^{k+1} - \delta_r e_{r+1} e_r^T E_n E_r^{-1}$. Hence $\bar{h}_{ij} = h_{ij}$ for all j and for all $i \neq r+1$, $\bar{h}_{r+1,j} = 0$ for all $j \neq n$,

$$\bar{h}_{r+1,n} = \frac{\delta_r y_n^k}{y_n^k} = h_{r+1,n} - \bar{c}_r h_{rn} - \sum_{i=r+2}^n \bar{c}_i h_{in}.$$

STEP 3. Permute rows of \bar{H}^{k+1} to obtain the upper-triangular matrix $\varphi_{r+1,n} \bar{H}^{k+1}$, where $\varphi_{r+1,n}$ is the permutation matrix defined by (6.4) for $i = r+1, j = n$.

Let us present now the 8-th step of the version of the PSUBQ algorithm using the method of Tomlin described above.

STEP 8b. Put $x^{k+1} = x^k - \theta^k \pi_r^k$, $z^{k+1} = z^k - \theta^k q^k$. Using Algorithm I, II, III or IV for the method of Tomlin transform H^{k+1} to the upper-triangular matrix U^{k+1} and compute the new $(L^{k+1})^{-1}$:

- (I) if h_{rr} was chosen as the pivot then either $U^{k+1} = (T^k)^{-1} C_k H^{k+1}$ and $(L^{k+1})^{-1} = (T^k)^{-1} C_k (L^k)^{-1}$ (if Step 3 was realized) or $U^{k+1} = \varphi_{rn} C_k H^{k+1} \cdot \varphi_{rn}^T$ and $(L^{k+1})^{-1} = \varphi_{rn} C_k (L^k)^{-1}$ (if Step 3a was realized and as far as we set that $\mathcal{H}^{k+1} = H^{k+1} \varphi_{rn}^T = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, U_n^k, Z_k)$ and $\mathcal{P}^{k+1} = P^{k+1} \varphi_{rn}^T = L^{k+1} U^{k+1}$);
- (II) if $h_{r+1,r}$ was chosen as the pivot then $U^{k+1} = \varphi_{r+1,n} \bar{C}_k H^{k+1} \varphi_{rn}^T = \varphi_{r+1,n} \bar{C}_k \mathcal{H}^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{r+1,n} \bar{C}_k (L^k)^{-1}$ as far as we set that $\mathcal{H}^{k+1} = H^{k+1} \varphi_{rn}^T$ and $\mathcal{P}^{k+1} = P^{k+1} \varphi_{rn}^T = L^{k+1} U^{k+1}$;
- (III) if h_{rn} was chosen as the pivot then $U^{k+1} = \varphi_{rn} \bar{C}_k H^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{rn} \bar{C}_k (L^k)^{-1}$;
- (IV) if $h_{r+1,n}$ was chosen as the pivot then $U^{k+1} = \varphi_{r+1,n} \bar{C}_k H^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{r+1,n} \bar{C}_k (L^k)^{-1}$.

Update R_β^k , R_α^k and IW^k according to (2.11), (2.12) and (2.13) of [4], respectively. Increase k by 1. Go to Step 2.

REMARK 6.2. To transform H^{k+1} to the upper-triangular matrix U^{k+1} using the method of Tomlin with partial or complete pivoting and to update $(L^k)^{-1}$ we must perform $N_{\text{add}} = \frac{(n-r)(3n-r+1)}{2} + n^2 - n$, $N_{\text{mult}} = \frac{(n-r)(3n-r+1)}{2} + n^2$, $N_{\text{div}} = 2n-r$ for Algorithm I with Step 3, and $N_{\text{add}} = N_{\text{mult}} = \frac{(n-r)(3n-r+1)}{2}$, $N_{\text{div}} = n-r$ for the remaining algorithms.

7. A version of the PSUBQ algorithm using the method of Forrest and Tomlin

Similarly to the version of the PSUBQ algorithm which uses the method of Bartels and Golub we form two matrices: a basic matrix $\mathcal{P}^{k+1} = (p_1^{k+1} = p_1^k, \dots, p_{r-1}^{k+1} = p_{r-1}^k, p_r^{k+1} = p_{r+1}^k, \dots, p_{n-1}^{k+1} = p_n^k, p_n^{k+1})$, where p_n^{k+1} is defined by (5.1), and an upper Hessenberg matrix $\mathcal{H}^{k+1} = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, U_n^k Z_k) = (h_{ij})_{i,j=1,\dots,n}$ (which is of the form (5.2)), where $Z_k = (L^k)^{-1} p_n^{k+1}$. Let us notice that $\mathcal{H}^{k+1} = H^{k+1} \varphi_{rn}^T = (U^k (I - e_r e_r^T) + Z_k e_r^T) \varphi_{rn}^T$ where φ_{rn}^T is the permutation matrix defined by (6.5) for $i=r, j=n$, and $Z_k = U^k y^k$ where $y^k = (P^k)^{-1} p_n^{k+1}$. Then $\mathcal{H}^{k+1} = U^k E_r^{-1} \varphi_{rn}^T$, where E_r is an elementary matrix of Gauss-Jordan method defined by the vector y^k and the r -th pivot (see (6.1)), and $E_r^{-1} = I - e_r e_r^T + y^k e_r^T$.

We describe now how to transform an upper Hessenberg matrix \mathcal{H}^{k+1} to an upper-triangular one by the method of Forrest and Tomlin [2, 5]. To ensure numerical stability we use the strategy of partial or complete pivoting. So first we must find the pivot among four elements of \mathcal{H}^{k+1} : $h_{rn}, h_{r+1,n}, h_{r,n-1}, h_{r+1,n-1}$. Depending on the pivot we choose one of four algorithms: *A, B, C* or *D*. If h_{rn} was chosen as the pivot then the method of Forrest and Tomlin with partial or complete pivoting is just the original method of Forrest and Tomlin [2, 5] and is carried out as Algorithm A:

STEP 1. Find $c^k = [0, \dots, 0, c_{r+1}, \dots, c_n]^T$ by solving the system:

$$c^{kT} U^k = u_r^k, \quad (7.1)$$

where $u_r^k = e_r^T U^k - U_{rr}^k e_r^T = [0, \dots, 0, U_{r,r+1}^k, \dots, U_{rn}^k]$. Hence $c^{kT} = e_r^T - U_{rr}^k e_r^T (U^k)^{-1}$.

STEP 2. Construct the matrix $C_k = I - e_r c^{kT} = I - e_r e_r^T + U_{rr}^k e_r e_r^T (U^k)^{-1}$ and obtain the matrix $\mathcal{H}'^{(k+1)} = C_k \mathcal{H}^{k+1} = (h'_{ij})_{i,j=1,\dots,n}$. Notice that $C_k \mathcal{H}^{k+1} = (I - e_r e_r^T + U_{rr}^k e_r e_r^T (U^k)^{-1}) \mathcal{H}^{k+1} = (I - e_r e_r^T) \mathcal{H}^{k+1} + U_{rr}^k e_r e_r^T (U^k)^{-1} U^k E_r^{-1} \varphi_{rn}^T = (I - e_r e_r^T) \mathcal{H}^{k+1} + U_{rr}^k e_r e_r^T E_r^{-1} \varphi_{rn}^T$. Hence $h'_{ij} = h_{ij}$ for all $i \neq r$

and for all j , $h'_{rj} = 0$ for all $j \neq n$, $h'_{rn} = U_{rr}^k y_r^k = h_{rn} - \sum_{i=r+1}^n c_i h_{in}$.

STEP 3. Permute rows of $\mathcal{H}'^{(k+1)}$ to obtain the upper-triangular matrix $\varphi_{rn} \mathcal{H}'^{(k+1)}$, where φ_{rn} is the permutation matrix defined by (6.4) for $i = r, j = n$.

If $h_{r+1,n}$ was chosen as the pivot then Algorithm B is performed.

STEP 1. Find $\bar{c}^k = [\bar{c}_1, \dots, \bar{c}_n]^T$ by solving the system:

$$\bar{c}^{kT} W^k = e_{r+1}^T U^k, \quad (7.2)$$

where $W^k = U^k (I - e_r e_r^T) + e_{r+1} e_r^T = (U_1^k, \dots, U_{r-1}^k, e_{r+1}, U_{r+1}^k, \dots, U_n^k)$. Notice that $\bar{c}_1 = \dots = \bar{c}_{r-1} = 0$, $\bar{c}_{r+1} = 0$ and $\bar{c}_r = \frac{U_{r+1,r+1}^k}{U_{r,r+1}^k}$ so far as $U_{r,r+1}^k \neq 0$.

Moreover, the system (7.2) is equivalent to the system $\bar{c}^{kT} U^k = e_{r+1}^T U^k + \delta_r e_r^T$, where $\delta_r = \frac{U_{rr}^k U_{r+1,r+1}^k}{U_{r,r+1}^k}$. Hence $\bar{c}^{kT} = e_{r+1}^T + \delta_r e_r^T (U^k)^{-1}$.

STEP 2. Construct the matrix $\bar{C}_k = I - e_{r+1} \bar{c}^{kT} = I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T (U^k)^{-1}$ and obtain the matrix $\bar{\mathcal{H}}^{k+1} = \bar{C}_k \mathcal{H}^{k+1} = (\bar{h}_{ij})_{i,j=1,\dots,n}$. Notice that $\bar{C}_k \mathcal{H}^{k+1} = (I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T (U^k)^{-1}) \mathcal{H}^{k+1} = (I - e_{r+1} e_{r+1}^T) \mathcal{H}^{k+1} - \delta_r e_{r+1} e_r^T (U^k)^{-1} U^k E_r^{-1} \varphi_{rn}^T = (I - e_{r+1} e_{r+1}^T) \mathcal{H}^{k+1} - \delta_r e_{r+1} e_r^T E_r^{-1} \varphi_{rn}^T$. Hence $\bar{h}_{ij} = h_{ij}$ for all $i \neq r+1$ and for all j , $\bar{h}_{r+1,j} = 0$ for all $j \neq n$, $\bar{h}_{r+1,n} = -\delta_r y_r^k = h_{r+1,n} - \bar{c}_r h_{rn} - \sum_{i=r+2}^n \bar{c}_i h_{in}$.

STEP 3. Permute rows of $\bar{\mathcal{H}}^{k+1}$ to obtain the upper-triangular matrix $\varphi_{r+1,n} \bar{\mathcal{H}}^{k+1}$, where $\varphi_{r+1,n}$ is the permutation matrix defined by (6.4) for $i = r+1, j = n$.

If $h_{r,n-1}$ was chosen as the pivot then Algorithm C is carried out:

STEP 1. Find $\bar{c}^k = [0, \dots, 0, \bar{c}_{r+1}, \dots, \bar{c}_n]^T$ by solving the system:

$$\bar{c}^{kT} U^k = \bar{u}_r^k, \quad (7.3)$$

where $\bar{u}_r^k = e_r^T U^k - U_{rr}^k e_r^T E_n = \left[0, \dots, 0, U_{r,r+1}^k, \dots, U_{rn}^k, U_{rn}^k + \frac{U_{rr}^k y_r^k}{y_n^k} \right]$ and E_n is an elementary matrix of the form (6.8).

Hence, $\bar{c}^{kT} = e_r^T - U_{rr}^k e_r^T E_n (U^k)^{-1}$.

STEP 2. Construct the matrix $\bar{C}_k = I - e_r \bar{c}^{kT} = I - e_r e_r^T + U_{rr}^k e_r e_r^T E_n (U^k)^{-1}$ and obtain the matrix $\bar{\mathcal{H}}^{k+1} = \bar{C}_k \mathcal{H}^{k+1} = (\bar{h}_{ij})_{i,j=1,\dots,n}$. Notice that $\bar{C}_k \mathcal{H}^{k+1} = (I - e_r e_r^T + U_{rr}^k e_r e_r^T E_n (U^k)^{-1}) \mathcal{H}^{k+1} = (I - e_r e_r^T) \mathcal{H}^{k+1} + U_{rr}^k e_r e_r^T E_n (U^k)^{-1} U^k E_r^{-1} \varphi_{rn}^T = (I - e_r e_r^T) \mathcal{H}^{k+1} + U_{rr}^k e_r e_r^T E_n E_r^{-1} \varphi_{rn}^T$. Hence $\bar{h}_{ij} = h_{ij}$ for all $i \neq r$ and for all j , $\bar{h}_{r,j} = 0$ for all $j \neq n-1$, $\bar{h}_{r,n-1} = \frac{U_{rr}^k y_r^k}{y_n^k} = h_{r,n-1} - \sum_{i=r+1}^n \bar{c}_i h_{i,n-1}$.

STEP 3. Permute rows and columns of $\bar{\mathcal{H}}^{k+1}$ to obtain the upper-triangular matrix $\varphi_{rn} \bar{\mathcal{H}}^{k+1} \varphi_{n-1,n}^T$, where φ_{rn} is the permutation matrix defined by (6.4) for $i=r, j=n$, and $\varphi_{n-1,n}^T$ is the permutation matrix exchanging the $(n-1)$ -th column with the n -th one.

If $h_{r+1,n-1}$ was chosen as the pivot the Algorithm *D* is carried out:

STEP 1. Find $\bar{c}^k = [\bar{c}_1, \dots, \bar{c}_n]^T$ by solving the system:

$$\bar{c}^{kT} \bar{W}^k = v_{r+1}^k, \quad (7.4)$$

where $v_{r+1}^k = e_{r+1}^T U^k E_n^{-1} = [0, \dots, 0, U_{r+1,r+1}^k, \dots, U_{r+1,n-1}^k, (Z_k)_{r+1}]$ and $\bar{W}^k = U^k E_n^{-1} (I - e_r e_r^T) + e_{r+1} e_r^T = (U_1^k, \dots, U_{r-1}^k, e_{r+1}, U_{r+1}^k, \dots, U_{n-1}^k, Z_k)$. Notice that $\bar{c}_1 = \dots = \bar{c}_{r-1} = 0, \bar{c}_{r+1} = 0$ and $\bar{c}_r = \frac{U_{r+1,r+1}^k}{U_{r,r+1}^k}$, so far as $U_{r,r+1}^k \neq 0$. Moreover, the system (7.4) is equivalent to the system $\bar{c}^{kT} U^k E_n^{-1} = e_{r+1}^T U^k E_n^{-1} + \delta_r e_r^T$, where $\delta_r = \frac{U_{rr}^k U_{r+1,r+1}^k}{U_{r,r+1}^k}$. Hence $\bar{c}^{kT} = e_{r+1}^T + \delta_r e_r^T$.

$L_n(L^k)^{-1}$.

STEP 2. Construct the matrix $\bar{C}_k = I - e_{r+1} \bar{c}^{kT} = I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T \cdot E_n (U^k)^{-1}$ and obtain the matrix $\bar{\mathcal{H}}^{k+1} = \bar{C}_k \bar{\mathcal{H}}^{k+1} = (\bar{h}_{ij})_{i,j=1,\dots,n}$. Notice that $\bar{C}_k \bar{\mathcal{H}}^{k+1} = (I - e_{r+1} e_{r+1}^T - \delta_r e_{r+1} e_r^T E_n (U^k)^{-1}) \bar{\mathcal{H}}^{k+1} = (I - e_{r+1} e_{r+1}^T) \cdot \bar{\mathcal{H}}^{k+1} - \delta_r e_{r+1} e_r^T E_n (U^k)^{-1} U^k E_n^{-1} \varphi_{rn}^T = (I - e_{r+1} e_{r+1}^T) \bar{\mathcal{H}}^{k+1} - \delta_r e_{r+1} e_r^T \cdot E_n E_n^{-1} \varphi_{rn}^T$. Hence $\bar{h}_{ij} = h_{ij}$ for all $i \neq r+1$ and for all $j, \bar{h}_{r+1,j} = 0$ for all $j \neq n-1, \bar{h}_{r+1,n-1} = \frac{\delta_r y_n^k}{y_n^k} = h_{r+1,n-1} - \bar{c}_r h_{r,n-1} - \sum_{i=r+2}^n \bar{c}_i h_{i,n-1}$.

STEP 3. Permute rows and columns of $\bar{\mathcal{H}}^{k+1}$ to obtain the upper-triangular matrix $\varphi_{r+1,n} \bar{\mathcal{H}}^{k+1} \varphi_{n-1,n}^T$, where $\varphi_{r+1,n}$ is the permutation matrix defined by (6.4) for $i=r+1, j=n$ and $\varphi_{n-1,n}^T$ is the permutation matrix exchanging the $(n-1)$ -th column with the n -th one.

Let us present now the 8-th step of the version of the PSUBQ algorithm using the method of Forest and Tomlin described above.

STEP 8c. Put $x^{k+1} = x^k - \theta^k \pi_r^k, z^{k+1} = z^k - \theta^k q^k$. Using Algorithm *A, B, C* or *D* for the method of Forrest and Tomlin transform $\bar{\mathcal{H}}^{k+1}$ to the upper-triangular matrix U^{k+1} and compute the new $(L^{k+1})^{-1}$:

- (A) if h_{rn} was chosen as the pivot then $U^{k+1} = \varphi_{rn} C_k \bar{\mathcal{H}}^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{rn} C_k (L^k)^{-1}$ since $\bar{\mathcal{H}}^{k+1} = (L^k)^{-1} \bar{\mathcal{P}}^{k+1}$;
- (B) if $h_{r+1,n}$ was chosen as the pivot then $U^{k+1} = \varphi_{r+1,n} \bar{C}_k \bar{\mathcal{H}}^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{r+1,n} \bar{C}_k (L^k)^{-1}$;
- (C) if $h_{r,n-1}$ was chosen as the pivot then $U^{k+1} = \varphi_{rn} \bar{C}_k \bar{\mathcal{H}}^{k+1} \varphi_{n-1,n}^T$ and $(L^{k+1})^{-1} = \varphi_{rn} \bar{C}_k (L^k)^{-1}$ as far as we set that $\bar{\mathcal{H}}^{k+1} = \bar{\mathcal{H}}^{k+1} \varphi_{n-1,n}^T = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, Z_k, U_n^k)$ and $\bar{\mathcal{P}}^{k+1} = \bar{\mathcal{P}}^{k+1} \varphi_{n-1,n}^T = L^{k+1} U^{k+1}$;

(D) if $h_{r+1,n-1}$ was chosen as the pivot then $U^{k+1} = \varphi_{r+1,n} \bar{C}_k \mathcal{H}^{k+1} \cdot \varphi_{n-1,n}^T$ and $(L^{k+1})^{-1} = \varphi_{r+1,n} \bar{C}_k (L^k)^{-1}$ as far as we set that $\mathcal{H}^{k+1} = \mathcal{H}^{k+1} \varphi_{n-1,n}^T = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, U_{n-1}^k, Z_k, U_n^k)$ and $\tilde{\mathcal{P}}^{k+1} = \mathcal{P}^{k+1} \varphi_{n-1,n}^T = L^{k+1} U^{k+1}$.

Update R_β^k , R_α^k and IW^k according to (2.11), (2.12) and (2.13) of [4], respectively. Increase k by 1. Go to Step 2.

REMARK 7.1. To transform the upper Hessenberg matrix \mathcal{H}^{k+1} to the upper-triangular matrix U^{k+1} using Algorithm A, B, C or D for the method of Forrest and Tomlin with partial or complete pivoting and to update $(L^k)^{-1}$ we must perform $N_{\text{add}} = N_{\text{mult}} = \frac{(n-r)(3n-r+1)}{2}$ and $N_{\text{div}} = n-r$.

8. A version of the PSUBQ algorithm using the modified methods of triangular factorization

We describe now how to take advantage of the structure of Z_k to modify two of methods of triangular factorization: the method of Bartels and Golub and the method of Forrest and Tomlin. Let $H^{k+1} = (L^k)^{-1} P^{k+1} = (U_1^k, \dots, U_{r-1}^k, Z_k, U_{r+1}^k, \dots, U_n^k) = (h_{ij})_{i,j=1,\dots,n}$ where $Z_k = (L^k)^{-1} p_r^{k+1}$ is the r -th column of H^{k+1} . Let $h_{l_k r}$ be the last nonzero element of Z_k . Let us notice that $l_k \geq r$. (If $l_k < r$ then $h_{rr} = 0$, $U^{k+1} = H^{k+1}$ and $\det(U^{k+1}) = 0$. But $\det P^{k+1} \neq 0$ and $P^{k+1} = L^{k+1} U^{k+1}$, a contradiction). If $l_k = r$ then $U^{k+1} = H^{k+1}$. So let us assume that $l_k > r$. Then H^{k+1} is of the following form:

(8.1)

The differences between the versions of the PSUBQ algorithm using the modified methods of triangular factorization and the versions using the methods described earlier do not appear up to the 5-th and the 6-th steps. Namely, the new column q^k or e_l is introduced into the basis matrix not instead of the n -th column but instead of the l_k -th one.

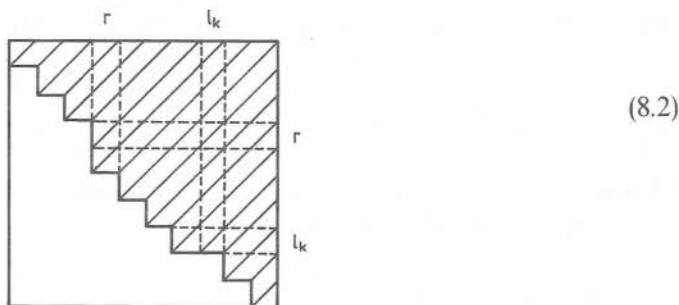
Let

$$\tilde{p}_{l_k}^{k+1} = \begin{cases} q^k & \text{if } q^k \neq 0 \text{ and } t_0^k > 1 \\ e_l & \text{if } q^k \neq 0 \text{ and } 0 \leq t_0^k \leq 1 \end{cases} \text{ of } q^k = 0.$$

and the new basis matrix is

$$\begin{aligned} P^{k+1} &= (\tilde{p}_1^{k+1} = p_1^k, \dots, \tilde{p}_{r-1}^{k+1} = p_{r-1}^k, \tilde{p}_r^{k+1} = p_{r+1}^k, \dots, \tilde{p}_{l_k-1}^{k+1} = \\ &= p_{l_k}^k, \tilde{p}_{l_k}^{k+1}, \tilde{p}_{l_k+1}^{k+1} = p_{l_k+1}^k, \dots, \tilde{p}_n^{k+1} = p_n^k). \end{aligned}$$

Let us notice that if $l_k = n$ then there is no reason to modify methods of triangular factorization. At Step 7 we compute $\tilde{Z}_k = (L^k)^{-1} \tilde{p}_{l_k}^{k+1}$ instead of Z_k and construct an upper Hessenberg matrix $\tilde{H}^{k+1} = (U_1^k, \dots, U_{r-1}^k, U_{r+1}^k, \dots, U_{l_k}^k, \tilde{Z}_k, U_{l_k+1}^k, \dots, U_n^k) = (\tilde{h}_{ij})_{i,j=1,\dots,n}$ which is of the form:



Let us notice, that $\tilde{H}^{k+1} = H^{k+1} \varphi_{rl_k}^T$, where $\varphi_{rl_k}^T$ is a permutation matrix defined by (6.5) for $i = r, j = l_k$. The way of realizing Step 8 of the PSUBQ algorithm depends on the method of triangular factorization. If we use the modified method of Bartels and Golub then we transform the upper Hessenberg matrix \tilde{H}^{k+1} to the upper-triangular matrix $U^{k+1} = \Gamma_{l_k-1}^k \varphi_{l_k-1}^k \dots \Gamma_r^k \varphi_r^k \tilde{H}^{k+1}$ and compute $(L^{k+1})^{-1} = \Gamma_{l_k-1}^k \varphi_{l_k-1}^k \dots \Gamma_r^k \varphi_r^k (L^k)^{-1}$ where Γ_i^k for $i = r, \dots, l_k-1$ are the elementary triangular matrices defined by (5.3) and φ_i^k for $i = r, \dots, l_k-1$ are either identity matrices or permutation matrices exchanging the i -th row with the $(i+1)$ -th one.

REMARK 8.1. To transform the upper Hessenberg matrix \tilde{H}^{k+1} to the upper-triangular matrix U^{k+1} using the modified method of Bartels and Golub and to update $(L^k)^{-1}$ we must perform $N_{\text{add}} = N_{\text{mult}} = \frac{(l_k-r)(4n-l_k-r+1)}{2}$

and $N_{\text{div}} = l_k - r$.

To transform the upper Hessenberg matrix \tilde{H}^{k+1} to U^{k+1} we can also use one of two modifications of the method of Forrest and Tomlin with partial or complete pivoting. For both of them, first of all we must

find the pivot among four elements of \tilde{H}^{k+1} : \tilde{h}_{rl_k} , \tilde{h}_{r+1,l_k} , \tilde{h}_{r,l_k-1} , \tilde{h}_{r+1,l_k-1} instead of elements of \mathcal{H}^{k+1} : h_{rn} , $h_{r+1,n}$, $h_{r,n-1}$, $h_{r+1,n-1}$. Let $\hat{H}^{k+1} = \tilde{H}^{k+1} (I - \sum_{j=l_k+1}^n e_j e_j^T) + \sum_{j=l_k+1}^n e_j e_j^T$ and $\hat{U}^k = U^k (I - \sum_{j=l_k+1}^n e_j e_j^T) + \sum_{j=l_k+1}^n e_j e_j^T$ be the matrices \tilde{H}^{k+1} and U^k with their last $n-l_k$ columns equal to e_{l_k+1}, \dots, e_n instead of $\tilde{H}_{l_k+1}, \dots, \tilde{H}_n$ and $U_{l_k+1}^k, \dots, U_n^k$, respectively, where \tilde{H}_j denotes the j -th column of \tilde{H}^{k+1} . Notice that, $\hat{H}^{k+1} = \hat{U}^k E_r^{-1} \varphi_{rl_k}^T$ and $\tilde{H}^{k+1} = \hat{H}^{k+1} (I - \sum_{j=l_k+1}^n e_j e_j^T) + \sum_{j=l_k+1}^n \tilde{H}_j e_j^T$. Hence $\tilde{h}_{ij} = \hat{h}_{ij}$ for all i and for all $j \leq l_k$.

Let us suppose that \tilde{h}_{rl_k} was chosen as the pivot. Then, the first modification of the method of Forrest and Tomlin is performed as Algorithm A':

STEP 1. Find $\hat{c}^k = [\hat{c}_1, \dots, \hat{c}_{l_k-r}]^T$ by solving l_k-r equations:

$$\hat{c}^{kT} \hat{U}^k = \hat{u}_r^k. \quad (8.3)$$

where $\hat{u}_r^k = [U_{r,r+1}^k, \dots, U_{r,l_k}^k]$, $\hat{U}^k = (\hat{U}_{ij}^k)_{i,j=1,\dots,l_k-r}$ and $\hat{U}_{ij}^k = U_{r+i,r+j}^k$ for $i, j = 1, \dots, l_k-r$.

Hence, $\hat{c}^k = [0, \dots, 0, c_{r+1}, \dots, c_{l_k}, 0, \dots, 0]^T$ such that $c_i = \hat{c}_{i-r}$ for $i = r+1, \dots, l_k$, is the solution for the system:

$$\hat{c}^{kT} \hat{U}^k = u_r^k, \quad (8.4)$$

where $u_r^k = e_r^T \hat{U}^k - U_{rr}^k e_r^T = [0, \dots, 0, U_{r,r+1}^k, \dots, U_{r,l_k}^k, 0, \dots, 0]$.

STEP 2. Construct the matrix $C_k = I - e_r c^{kT} = I - e_r e_r^T + U_{rr}^k e_r e_r^T (\hat{U}^k)^{-1}$ and obtain $\hat{H}'^{(k+1)} = C_k \hat{H}^{k+1} = (\hat{h}'_{ij})_{i,j=1,\dots,n}$. Notice that $C_k \hat{H}^{k+1} = (I - e_r e_r^T) \hat{H}^{k+1} + U_{rr}^k e_r e_r^T (\hat{U}^k)^{-1} \hat{H}^{k+1} = (I - e_r e_r^T) \hat{H}^{k+1} + U_{rr}^k e_r e_r^T (\hat{U}^k)^{-1} \cdot \hat{U}^k E_r^{-1} \varphi_{rl_k}^T = (I - e_r e_r^T) \hat{H}^{k+1} + U_{rr}^k e_r e_r^T E_r^{-1} \varphi_{rl_k}^T$. Hence, $\hat{h}'_{ij} = h_{ij}$ for all $i \neq r$ and for all j , $\hat{h}'_{rj} = 0$ for all $j \neq l_k$, $\hat{h}'_{rl_k} = U_{rr}^k y_r^k = \hat{h}_{rl_k} - \sum_{i=r+1}^{l_k} c_i \hat{h}_{il_k}$. Now,

$$\begin{aligned} \tilde{H}'^{(k+1)} &= C_k \tilde{H}^{k+1} = C_k \hat{H}^{k+1} (I - \sum_{j=l_k+1}^n e_j e_j^T) + C_k \sum_{j=l_k+1}^n \tilde{H}_j e_j^T = \\ &= \hat{H}'^{(k+1)} (I - \sum_{j=l_k+1}^n e_j e_j^T) + (I - e_r c^{kT}) \sum_{j=l_k+1}^n \tilde{H}_j e_j^T. \end{aligned}$$

Hence,

$$\tilde{h}'_{ij} = \hat{h}'_{ij} = \hat{h}_{ij} = \tilde{h}_{ij} \text{ for all } j \leq l_k \text{ and for all } i \neq r.$$

$$\tilde{h}'_{ij} = \tilde{h}_{ij} \text{ for } j > l_k \text{ and for all } i \neq r,$$

$$\tilde{h}'_{rj} = \tilde{h}'_{rj} = 0 \text{ for all } j < l_k,$$

$$\tilde{h}'_{rl_k} = \hat{h}'_{rl_k} = \hat{h}_{rl_k} - \sum_{i=r+1}^{l_k} c_i \tilde{h}_{il_k} = \tilde{h}_{rl_k} - \sum_{i=r+1}^{l_k} c_i \tilde{h}_{il_k},$$

$$\tilde{h}'_{rj} = \tilde{h}_{rj} - \sum_{i=r+1}^{l_k} c_i \tilde{h}_{ij} \text{ for all } j > l_k.$$

STEP 3. Permute rows of $\tilde{H}'^{(k+1)}$ to obtain an upper-triangular matrix $\varphi_{rl_k} \tilde{H}'^{(k+1)}$, where φ_{rl_k} is a permutation matrix defined by (6.4) for $i = r$, $j = l_k$.

At Step 8 of the PSUBQ algorithm we obtain then $U^{k+1} = \varphi_{rl_k} C_k \tilde{H}^{k+1}$ and $(L^{k+1})^{-1} = \varphi_{rl_k} C_k (L^k)^{-1}$.

Analogically we can modify Algorithm B, C or D in the case when \tilde{h}_{r+1, l_k} , \tilde{h}_{r, l_k-1} or \tilde{h}_{r+1, l_k-1} was chosen as the pivot to obtain Algorithm B', C' or D', respectively.

REMARK 8.2. To transform the upper Hessenberg matrix \tilde{H}^{k+1} to the upper-triangular matrix U^{k+1} using the first modification of the method of Forrest and Tomlin and to update $(L^k)^{-1}$ we must perform

$$N_{\text{add}} = N_{\text{mult}} = \frac{(l_k - r)(4n - l_k - r + 1)}{2}, \quad N_{\text{div}} = l_k - r.$$

The second modification is just the method of Forrest and Tomlin with partial or complete pivoting used for triangular factorization of $\tilde{H}^{k+1} = H^{k+1} \varphi_{rl_k}^T$ instead of $\mathcal{H}^{k+1} = H^{k+1} \varphi_{rn}^T$, where $\varphi_{rl_k}^T$ and φ_{rn}^T are permutation matrices defined by (6.5). The only difference is that everywhere in the method of Forrest and Tomlin where the index n appears we replaced it by the index l_k , i.e. we deal with matrices φ_{rl_k} , $\varphi_{rl_k}^T$, φ_{r+1, l_k} , φ_{l_k-1, l_k}^T , E_{l_k} and $E_{l_k}^{-1}$ instead of matrices φ_{rn} , φ_{rn}^T , $\varphi_{r+1, n}$, $\varphi_{n-1, n}^T$, E_n and E_n^{-1} , respectively.

REMARK 8.3. Let us notice, that to transform \tilde{H}^{k+1} to U^{k+1} by the second modification of the method of Forrest and Tomlin with partial or complete pivoting we must perform the same number of elementary arithmetic operations as it was required in transforming \mathcal{H}^{k+1} to U^{k+1} by the original method of Forrest and Tomlin with partial or complete pivoting (see Remark 7.1), i.e. $N_{\text{add}} = N_{\text{mult}} = \frac{(n-r)(3n-r+1)}{2}$ and $N_{\text{div}} = n-r$. The only advantage is a decrease of the number of required permutations at Step 3 of the algorithms A, B, C and D.

9. Conclusion

The versions of the PSUBQ algorithm presented in Sections 4–8 require performing additional operations. First, at Step 1, we must obtain once the triangular decomposition of the initial basis matrix P^0 (which requires performing $\frac{n(n-1)(n-2)}{2}$ multiplications, the same number of

additions, and $\frac{(n-1)(n-2)}{2}$ divisions) and obtain $(L^0)^{-1}$ by solving n systems of n linear equations (which requires performing $\frac{n^2(n-1)}{2}$ multiplications, the same number of additions, and n^2 divisions). Hence, we must additionally perform

$$N_{\text{add}} = N_{\text{mult}} = n(n-1)^2 \quad \text{and} \quad N_{\text{div}} = \frac{3}{2}n(n-1) + 1.$$

- Many times (at each iteration of the PSUBQ algorithm) we must
- at Step 2, compute \bar{w}^k by solving a system of $n-m$ linear equations, which requires $\frac{(n-m-1)(n-m)}{2}$ multiplications and the same number of additions, and $n-m$ divisions;
 - at Step 3, compute π_r^{kT} by solving n systems of $n-r+1$ linear equations, which requires $\frac{n(n-r+1)(n-r)}{2}$ multiplications and the same number of additions, and $n(n-r+1)$ divisions;
 - at Step 7, compute $Z_k = (L^k)^{-1} p_r^{k+1}$ or $Z_k = (L^k)^{-1} p_n^{k+1}$ or $\tilde{Z}_k = (L^k)^{-1} \tilde{p}_{i_k}^{k+1}$, which requires $n(n-1)$ additions and n^2 multiplications;
 - at Step 7 and 8 form and reduce H^{k+1} or \mathcal{H}^{k+1} or \tilde{H}^{k+1} to triangular form U^{k+1} by one of methods of triangular factorization described in Sections 4–8, and next update $(L^k)^{-1}$.

The Table 9.1 contains comparison of costs for different methods of matrix factorization calculated in Sections 4–8.

Comparing costs (in the sense of numbers of elementary arithmetic operations and required permutations of rows and/or columns) of the presented methods of matrix factorization, we can order them along increasing costs in the following sequence:

1. the first modification of the method of Forrest and Tomlin with pivoting (if $l_k < n$).
2. the modified method of Bartels and Golub (if $l_k < n$),
3. the second modification of the method of Forrest and Tomlin with pivoting (if $l_k < n$),
4. the method of Bartels and Golub,
5. the method of Forrest and Tomlin with pivoting,
6. the method of Tomlin with pivoting (unless we use Algorithm I with Step 3),
7. the method of Tomlin with pivoting (if we use Algorithm I with Step 3),
8. Gaussian elimination.

Table 9.1

	Gaussian elimination	The method of Bartels and Golub	The modified method of Bartels and Golub	The method of Tomlin	The method of Forrest and Tomlin	The modified method of Forrest and Tomlin	
						The 1 st modification	The 2 nd modification
N_{add}	$(n-r)(n-r+1)(5n-2r+1)$	$(n-r)(3n-r+1)$	$(l_k-r)(4n-l_k-r+1)$	for Algorithm I with Step 3 $\frac{(n-r)(3n-r+1)}{2} + n^2 - n$	$(n-r)(3n-r+1)$	$(l_k-r)(4n-l_k-r+1)$	$(n-r)(3n-r+1)$
	6	2	2	for the remaining Algorithms $\frac{(n-r)(3n-r+1)}{2}$	2	2	2
N_{mult}	—”—	—”—	—”—	for Algorithm I with Step 3 $\frac{(n-r)(3n-r+1)}{2} + n^2$	—”—	—”—	—”—
				for the remaining Algorithms $\frac{(n-r)(3n-r+1)}{2}$			
N_{div}	$\frac{(n-r+1)(n-r)}{2}$	$n-r$	l_k-r	for Algorithm I with Step 3 $2n-r$	$n-r$	l_k-r	$n-r$
				for the remaining Algorithms $n-r$			

Let us notice that the 1-st and the 2-nd method from the above sequence require the same number of elementary arithmetic operations. They require $\frac{(n-l_k)(3n-l_k+1)}{2}$ additions, the same number of multiplications and $n-l_k$ divisions less than the 3-rd one.

Next, a number of elementary arithmetic operations for the 3-rd, 4-th, 5-th and 6-th method is the same. They vary only on numbers of permutations of rows and/or columns. The 6-th method requires n^2-n additions, n^2 multiplications and n divisions less than the 7-th one and it is also cheaper than the 8-th one since it requires $\frac{(n-r)(n-r-1)(5n-2r+2)}{6}$ additions, the same number of multiplications and $\frac{(n-r)(n-r-1)}{2}$ divisions less than Gaussian elimination. Moreover, costs for the 7-th and the 8-th methods are comparable.

References

- [1] BARTELS R. H., GOLUB G. The simplex method of linear programming using LU decomposition. *Communications ACM*, **12** (1969), 5.
- [2] FORREST J. H., TOMLIN J. A. Updated triangular factors of the basis to maintain sparsity in the product form Simplex Method. *Mathematical Programming* 2, 1972, p. 263-278.
- [3] HADLEY H. *Linear Programming*. Reading, Addison-Wesley, 1962.
- [4] KRYŃSKA G. The Primal Algorithm Using Conjugate Directions for Quadratic Programming Problems with Simple Upper Bounds. *Optimization*, **18** (1987), 4.
- [5] TOLLA P. Contribution à l'amélioration des logiciels de programmation mathématique en variables réelles. Doctoral dissertation, Paris 1983.
- [6] TOMLIN J. A. On pricing and Backward transformation in linear programming. *Mathematical Programming* 6, 1974, p. 42-47.
- [7] WILKINSON J. H. *Rounding errors in algebraic processes*. London 1963.
- [8] WILKINSON J. H. *The algebraic eigenvalue problem*. Clarendon Press, Oxford 1965.

Received, September 1986

Numerycznie stabilna implementacja algorytmu PSUBQ

W pracy [4] został opisany algorytm PSUBQ kierunków sprzężonych, zbadana została jego zbieżność a także porównano go z innymi algorytmami z tej samej klasy metod kierunków dopuszczalnych. W niniejszej pracy uzupełniono rozważania dotyczące numerycznych własności algorytmu PSUBQ i przedstawiono cztery różne numerycznie stabilne metody triangularyzacji macierzy.

Численно устойчивое применение алгоритма PSUBQ

В работе [4] описаны алгоритм PSUBQ сопряженных направлений. Исследована его сходимость, а также дается сравнение его с другими алгоритмами из этого же класса методов допустимых направлений. В данной работе дополнительно рассмотрены понятия, касающиеся численных свойств алгоритма PSUBQ и представлены четыре разные, численно устойчивые методы триангуляции матриц.

