BALANCING ALGORITHM FOR THREE MATRICES

NELA BOSNER*

Abstract. Three versions of an algorithm for balancing three matrices simultaneously are proposed. The balancing is performed by premultiplication and postmultiplication with positive definite diagonal matrices, in order to reduce the magnitude range of all elements in the involved matrices. Some numerically stable algorithms, when applied to matrices with a wide range in the magnitude of elements, can produce results with a large error. As an application we present several problems from control theory. A reduction to the m-Hessenberg–triangular–triangular form is efficiently used for computing the frequency response $\mathcal{G}(\sigma) = C(\sigma E - A)^{-1}B + D$ of a descriptor system. The reduction algorithm can produce inaccurate results for badly scaled matrices. Numerical experiments confirmed that balancing matrices A, B and E before the m-Hessenberg–triangular–triangular reduction can produce an accurate frequency response matrix. Balancing three matrices can also improve the solution of the pole assignment problem for descriptor linear systems via state feedback, and the determination of the controllable part of the system. Other applications are: the quadratic eigenvalue problem $\lambda^2 Ax + \lambda Ex + Bx = 0$, solution of the algebraic linear system $(\sigma^2 A + \sigma B + C)x = b$ for several values of the parameter σ, and any other problems involving three matrices of same dimensions. These problems are not covered by the paper, but they can be balanced with a simplified version of the proposed algorithm.

Key words. balancing three matrices, diagonal transformations, numerical stability

AMS subject classifications. 65F35, 65Y04

1. Introduction. A numerically stable algorithm, when applied to matrices with a wide range in the magnitude of elements, can nevertheless produce a result with a large error. The standard attempt to reduce the magnitude range of elements of a matrix A is to scale its rows and columns, by multiplication with positive definite diagonal matrices. Such a technique is used prior to solving a linear system $Ax = b$ in [7]. As emphasized in the introduction of [9], in the absence of any other information, a satisfactory scaling is one in which the absolute errors in the elements are all about the same size. This choice of scaling strategy makes the condition number meaningful. In case when only rounding errors are introduced, the error in an element is proportional to its size and the scaling forces all elements of A to be about the same size. This process is called balancing. Furthermore, the balanced system can produce a more accurate result. The same problem is observed when solving the standard eigenvalue problem $Ax = \lambda x$, where inaccuracies in eigenvalues and eigenvectors are reduced by a diagonal similarity transformation, introduced by Parlett and Reinsch in [18]. This similarity transformation equilibrates the column and row norms, and reduces the norm of A. In the generalized eigenvalue problem $Ax = \lambda Bx$, balancing is enforced on both matrices A and B, as proposed by Ward in [27], and by Lemonnier and Van Dooren in [14]. It is important to emphasize here that balancing will in most cases improve condition numbers, and the balanced problems will produce more accurate results than the original ones. Nevertheless, there are examples where balancing does not improve the condition of the problem, or even worse, it can produce more ill-conditioned problems as illustrated by Watkins [28].

On the other hand, some algorithms are almost invariant under scaling in respect of numerical stability. Such algorithms are, for example, Jacobi algorithms for solving symmetric eigenvalue and singular value problems, see [8]. In the case

*Department of Mathematics, University of Zagreb, Croatia, (nela.bosner@math.hr), the author is supported by the Croatian MZOS grant 0372783–2750.

of a positive definite matrix A the condition number for eigenvalues is bounded by $n \cdot \min_{D\text{diagonal}} \kappa(DAD)$, and the condition number for singular values is bounded by $\sqrt{n} \cdot \min_{D\text{diagonal}} \kappa(AD)$ (see van der Sluis [25]), showing that the forward errors are invariant under appropriate scalings.

In this paper we propose efficient algorithms for balancing three matrices, and we study how the scaling issues affect computational tasks in computational control. Specially, we are interested in numerical problems related to descriptor systems, which involve three matrices of different dimensions. A descriptor system has the following form

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1.1}$$

where $A, E \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $m \ll n$. As pointed out by Paige in [16], scaling represents a coordinate transformation. When $\tilde{x} = D_2^{-1}x$, $\tilde{u} = D_3^{-1}u$, and $\tilde{y} = D_4 y$, for positive definite diagonal matrices D_1, D_2, D_3 and D_4, then the system

$$\begin{aligned} D_1 E D_2 \dot{\tilde{x}}(t) &= D_1 A D_2 \tilde{x}(t) + D_1 B D_3 \tilde{u}(t) \\ \tilde{y}(t) &= D_4 C D_2 \tilde{x}(t) + D_4 D D_3 u(t), \end{aligned}$$

is equivalent to (1.1). Paige distinguishes two reasons for scaling in control theory. The first one is to choose coordinate transformations and units (this corresponds to diagonal scalings) so that the mathematical problem accurately reflects the sensitivity of the physical problem. The second reason is to minimize the effect of rounding errors on the computed solution, and is less important. Scaling for numerical stability must not alter physical sensitivity. As an example of scaling in control theory Paige refers to measuring how far a system with $E = I$, where I is the identity matrix, is from an uncontrollable one. His proposed measure is pessimistic if bounds on model uncertainties are dominated by the uncertainties in just a few elements. In this case, a good scaling is similar to the one for the generalized eigenvalue problem: scale so that the uncertainties in elements of A and B are all of the same order of magnitude. In order to determine controllability of the system (1.1) with a general matrix E, the scaling has to include three matrices A, B and E.

In [3] and [4] an efficient algorithm for computing the frequency response matrix $\mathcal{G}(\sigma) = C(\sigma E - A)^{-1}B + D$ of the system (1.1) is proposed, for large number of shifts σ. The first part of this algorithm comprises of the m-Hessenberg–triangular–triangular reduction of matrices A, B and E performed by a sequence of orthogonal transformations, and an efficient version of this algorithm is introduced in [3]. A similar reduction is used for solving the pole assignment problem in descriptor linear systems via state feedback. This problem is very sensitive to input data.

The algorithms of our interest, which are related to the descriptor systems, are based on orthogonal transformations. Even orthogonal transformations applied to badly scaled matrices can result in adding two numbers with a large difference in order of magnitude. In floating point arithmetic the influence of the number with the small order of magnitude can be completely lost in the sum. Although the orthogonal transformation always produces a result with a small relative error, in case of badly scaled matrices they can severely increase the condition number. For example, Powell and Reid in [20] observed that for the least squares problems, in which the rows of the coefficient matrix vary widely in norm, Householder QR factorization has unsatisfactory backward stability properties. They showed that row and column pivoting give

desirable backward error, and Cox and Higham in [6] proved that sorting the rows by descending ∞-norm at the start gives the same result.

In order to avoid this sort of numerical instability in problems involving three matrices, it is advisable to balance all three matrices. The balancing is performed by diagonal transformations which reduce the difference in orders of magnitude of all elements in these matrices. We will illustrate the need for balancing with an example. In [16] (see also [3]) an algorithm that reveals controllability of the system (1.1) with nonsingular E is proposed. The original system is transformed to an equivalent system with a suitable form, and for $m = 1$ this form is equivalent to the m-Hessenberg–triangular–triangular form. The algorithm for the m-Hessenberg–triangular–triangular reduction first reduces E to the upper triangular form. Then it annihilates one by one element below the diagonal of B with the Givens rotations applied from the left, while simultaneously maintaining the triangular form of E by applying the Givens rotations from the right. When B is done the algorithm switches to A, annihilating elements below the m-th subdiagonal. Suppose that E and B are already upper triangular. Let A, B and E be defined as follows

$$A = \begin{bmatrix} \frac{5}{4} & -\frac{1}{2} & 2 \\ 1 & \frac{3}{4} & -\frac{1}{3} \\ 1 & -\frac{1}{4} & \frac{1}{30} \end{bmatrix}, \qquad B = \begin{bmatrix} \frac{3}{2} \\ 0 \\ 0 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & \epsilon \end{bmatrix},$$

where ϵ is a small number. Nevertheless, E is a nonsingular matrix. Now, we want to annihilate the element on position $(3,1)$ of A by the Givens rotation G_l. We obtain

$$G_l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad A_1 = G_l A = \begin{bmatrix} \frac{5}{4} & -\frac{1}{2} & 2 \\ \sqrt{2} & \frac{1}{2\sqrt{2}} & -\frac{9}{30\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{11}{30\sqrt{2}} \end{bmatrix},$$

$$E_1 = G_l E = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1+\epsilon}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{-1+\epsilon}{\sqrt{2}} \end{bmatrix}.$$

When the above computation is performed in finite precision arithmetic with the roundoff error u, and when $\epsilon \leq u/2$, then the computed matrix $\hat{E}_1$ and its elements are of the form

$$\begin{aligned} \mathrm{fl}\left(\tfrac{1}{\sqrt{2}} + \tfrac{\epsilon}{\sqrt{2}}\right) &= \mathrm{fl}\left(\tfrac{1}{\sqrt{2}}\right), \\ \mathrm{fl}\left(-\tfrac{1}{\sqrt{2}} + \tfrac{\epsilon}{\sqrt{2}}\right) &= -\mathrm{fl}\left(\tfrac{1}{\sqrt{2}}\right), \end{aligned} \qquad \hat{E}_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \mathrm{fl}\left(\frac{1}{\sqrt{2}}\right) & \mathrm{fl}\left(\frac{1}{\sqrt{2}}\right) \\ 0 & -\mathrm{fl}\left(\frac{1}{\sqrt{2}}\right) & -\mathrm{fl}\left(\frac{1}{\sqrt{2}}\right) \end{bmatrix},$$

and $\hat{E}_1$ is singular. In this example, matrix E is badly scaled which caused adding one large and one small number, and thus loosing nonsingularity of the orthogonally transformed matrix E. Actually, the condition number of E_1 increased to infinity. If our task was to solve the pole assignment problem, this result would reduce the number of finite poles that can be assigned. On the other hand, if the balancing algorithm presented in the next section is applied to the matrices A, B, and E, it produces matrices $A_{bal} = D_l A D_r$, $B_{bal} = D_l B$, and $E_{bal} = D_l E D_r$, with $D_l = \mathrm{diag}(1, 1, 10^4)$

and $D_r = \mathrm{diag}(0.1, 1, 100)$. Elements which determine the Givens rotation $G_{bal,1}$ that annihilates $A_{bal}(3,1)$ are $(0.1, 1000)$, and the updated matrix $\hat{E}_{bal,1} = \mathrm{fl}(G_{bal,1}E_{bal})$ has the form

$$\hat{E}_{bal,1} = \begin{bmatrix} 1.000000000000000\cdot 10^{-1} & 1.000000000000000\cdot 10^{0} & 1.000000000000000\cdot 10^{2} \\ 0 & 9.999999950000001\cdot 10^{-5} & 1.000000000551115\cdot 10^{-2} \\ 0 & -9.999999950000000\cdot 10^{-1} & -9.999999950000000\cdot 10^{1} \end{bmatrix}.$$

$\hat{E}_{bal,1}$ is now both exactly and numerically nonsingular.

Besides the control theory, there are other examples involving three matrices of same dimensions. Such an example is the quadratic eigenvalue problem $\lambda^2 Ax + \lambda Ex + Bx = 0$, for $A, B, E \in \mathbb{R}^{n\times n}$. A similar example comes from the structural dynamics engineering problem, where direct frequency analysis leads to the solution of the algebraic linear system $(\sigma^2 A + \sigma B + C)x = b$ for several values of the frequency-related parameter σ (see, for example, [22] and [23]).

1.1. Balancing two matrices. Ward [27] proposes a balancing technique which applies to two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ involved in the generalized eigenvalue problem. This technique produces two diagonal matrices D_l and D_r such that the range of elements in $D_l A D_r$ and $D_l B D_r$ is optimally small. The basic idea is forcing the exponents in exponential notation of all nonzero elements in $D_l A D_r$ and $D_l B D_r$ to be as close to zero as possible. The diagonal matrices are defined as $D_l = \mathrm{diag}(10^{l_1}, \ldots, 10^{l_n})$ and $D_r = \mathrm{diag}(10^{r_1}, \ldots, 10^{r_n}) \in \mathbb{R}^{n\times n}$. Besides 10, another radix can be used for exponential representation. For example, multiplication with powers of 2 introduces no rounding errors. The elements of the scaled matrices are of the form $(D_l A D_r)_{ij} = 10^{l_i + r_j} a_{ij}$ and $(D_l B D_r)_{ij} = 10^{l_i + r_j} b_{ij}$. The magnitude of an element is represented as the logarithm of its absolute value. In order to reduce the range of magnitude for elements, the objective

$$\min_{l_i, r_j} \left[\sum_{\substack{i,j=1 \\ a_{ij}\neq 0}}^{n} (l_i + r_i + \log_{10}|a_{ij}|)^2 + \sum_{\substack{i,j=1 \\ b_{ij}\neq 0}}^{n} (l_i + r_i + \log_{10}|b_{ij}|)^2 \right].$$

is minimized, using a generalized conjugate gradient method developed by Concus et al. [5]. This algorithm is implemented in the LAPACK [1] routine `dggbal`.

2. Balancing three matrices. Our main goal is to indicate a need of balancing three matrices in particular problems, and we will illustrate its benefits in Section 4.

In the Control and Systems Library SLICOT [24] there already exists the routine `TG01AD` which can balance three matrices, but it has several drawbacks. It is only a slight modification of the Ward's algorithm. The minimization function takes into account only the element with the largest magnitude in each row of the matrix B. On the other hand, the generalized conjugate gradient method in this routine uses the same preconditioner as in the case of two matrices. The preconditioner in `dggbal` is a singular matrix, while the system of normal equations in case of three matrices can be nonsingular. Thus, the conjugate gradient method can stop before reaching the minimum of the minimization function. The second drawback of `TG01AD` is that it can result with unsatisfactory scaling of the matrix B, leaving it with a large norm. This is particulary inconvenient for algorithms that include rank revealing, such as the staircase reduction. Both drawbacks are illustrated by examples and presented in Section 4.

Besides stressing the benefits of the balancing algorithms, our intention is to offer a proper and better algorithm for balancing three matrices, which will correct both drawbacks of the routine `TG01AD` and provide more functionality.

We will extend Ward's approach to balancing three matrices which will change the minimization problem, but the minimization algorithm will remain the same. Our balancing algorithm will produce diagonal matrices D_l and D_r, such that the range of magnitude orders of all elements in the matrices D_lAD_r, D_lED_r, and D_lB is optimally small. Computation of the frequency response matrix $\mathcal{G}(\sigma)$ is invariant under such diagonal transformations. Optionally we can balance the matrix B from the right introducing the third diagonal matrix D_B.

After introducing abbreviations $l = (l_1, \ldots, l_n)$ and $r = (r_1, \ldots, r_n)$, our problem of balancing matrices $A = [a_{ij}]$, $B = [b_{ij}]$, and $E = [e_{ij}]$ is equivalent to the minimization problem

$$\min_{l,r \in \mathbb{R}^n} \phi(l, r), \tag{2.1}$$

$$\phi(l,r) = \sum_{i=1}^{n} \left[\sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|a_{ij}|)^2 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|e_{ij}|)^2 + \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} (l_i + \log_{10}|b_{ij}|)^2 \right].$$

To find a solution of the minimization problem (2.1), we differentiate the function $\phi(l, r)$ and equalize its gradient with zero, as in [27]. After obtaining minimizing l_{min} and r_{min}, if integers in l and r are required, they can be retrieved by the rounding operation. Further, $\nabla\phi(l, r) = 0$ results with a linear system $Lx = p$ which has the following form

$$\begin{bmatrix} F_1 & G \\ G^T & F_2 \end{bmatrix} \begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} -c \\ -d \end{bmatrix}, \qquad L = \begin{bmatrix} F_1 & G \\ G^T & F_2 \end{bmatrix},\ p = \begin{bmatrix} -c \\ -d \end{bmatrix}, \tag{2.2}$$

where

1. $F_1 \in \mathbb{R}^{n\times n}$ is a diagonal matrix $F_1 = \mathrm{diag}(n_{r_1}, \ldots, n_{r_n})$ and

$$n_{r_i} = \sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} 1 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} 1 + \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} 1$$

is the total number of nonzero elements in the i-th rows of A, B, and E,

2. $F_2 \in \mathbb{R}^{n\times n}$ is a diagonal matrix $F_2 = \mathrm{diag}(n_{c_1}, \ldots, n_{c_n})$ and

$$n_{c_j} = \sum_{\substack{i=1 \\ a_{ij}\neq 0}}^{n} 1 + \sum_{\substack{i=1 \\ e_{ij}\neq 0}}^{n} 1$$

is the total number of nonzero elements in the j-th columns of A and E,

3. $G = [g_{ij}] \in \mathbb{R}^{n\times n}$ is the sum of the incidence matrices of A and E, i. e.

$$g_{ij} = \left\{ \begin{array}{ll} 1, & \text{if } a_{ij} \neq 0 \\ 0, & \text{if } a_{ij} = 0 \end{array} \right\} + \left\{ \begin{array}{ll} 1, & \text{if } e_{ij} \neq 0 \\ 0, & \text{if } e_{ij} = 0 \end{array} \right\},$$

4. $c = [c_i] \in \mathbb{R}^n$ has elements

$$c_i = \sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} \log_{10}|a_{ij}| + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} \log_{10}|e_{ij}| + \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} \log_{10}|b_{ij}|,$$

5. $d = [d_j] \in \mathbb{R}^n$ has elements

$$d_j = \sum_{\substack{i=1 \\ a_{ij} \neq 0}}^{n} \log_{10} |a_{ij}| + \sum_{\substack{i=1 \\ e_{ij} \neq 0}}^{n} \log_{10} |e_{ij}|.$$

In the special case when the matrices A, B, and E contain only nonzero elements the system matrix in (2.2) reduces to

$$M = \begin{bmatrix} (2n+m)I & 2ee^T \\ 2ee^T & 2nI \end{bmatrix}, \tag{2.3}$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix and $e = [\ 1 \quad \dots \quad 1\ ]^T \in \mathbb{R}^n$.

The minimization problem (2.1) is in fact a linear least squares problem, and (2.2) represent its system of normal equations. We know that the matrix of normal equations is positive semidefinite, and that the system is consistent (see [11], [2]). Nevertheless, we found the structure of this system interesting. We explored its properties directly from the definition (2.2) and stated them in the following proposition. Since we did not encounter such an approach in literature on balancing techniques, we present the proof of the proposition with some details.

PROPOSITION 2.1.

(i) *The system matrix L of the system (2.2) is symmetric positive semidefinite or positive definite.*

(ii) *If L is singular positive semidefinite, then the system (2.2) is consistent.*

(iii) *The matrix M defined with (2.3) is symmetric positive definite, and its inverse is equal to*

$$M^{-1} = \begin{bmatrix} \frac{1}{2n+m} I + \frac{2}{(2n+m)m} ee^T & -\frac{1}{nm} ee^T \\ -\frac{1}{nm} ee^T & \frac{1}{2n} I + \frac{1}{nm} ee^T \end{bmatrix}. \tag{2.4}$$

Proof.

(i) The proof of positive semidefinitness of L can be obtained by mathematical induction on the total number of nonzero elements in A, B, and E. If $A = E = 0_{n \times n}$ and $B = 0_{n \times m}$ are zero matrices, then $L = 0_{2n \times 2n}$ which can be regarded as a degenerate positive semidefinite matrix.

Let us assume that for A, B, and E, which have altogether N nonzero elements, the matrix L is positive semidefinite. Let us observe what happens when we add one more nonzero element to one of these three matrices. If this extra element is $a_{i_0 j_0}$ or $e_{i_0 j_0}$, then the elements of L with indices (i_0, i_0), $(i_0, n+j_0)$, $(n+j_0, i_0)$, and $(n+j_0, n+j_0)$ are incremented by 1. In case when this extra element is $b_{i_0 j_0}$, then only the element of L with indices (i_0, i_0) is incremented by 1. The new system matrix in (2.2) is now of the form

$$L + H, \quad H = uu^T,$$

where $u = \xi_{i_0} + \xi_{n+j_0}$ for the first case, and $u = \xi_{i_0}$ for the second case, with ξ_i being the i-th unit vector. In both cases the matrix H is symmetric positive semidefinite (the spectrum of H is $\sigma(H) = \{0, 2\}$ or $\sigma(H) = \{0, 1\}$). If we order eigenvalues of the matrices L, H, and $L + H$ in the ascending order, then by the Weyl monotonicity theorem [17, Section 10-3] we have

$$\lambda_j(L + H) \geq \lambda_j(L) + \lambda_1(H) \geq 0, \quad j = 1, \dots, 2n,$$

and the matrix $L + H$ is positive semidefinite, too.
In case when there is no zero elements in A, B, and E, then L is equal to M from (2.3). The spectrum of M is $\sigma(M) = \{\frac{1}{2}((4n+m)-\sqrt{16n^2+m^2}), 2n, 2n+m, \frac{1}{2}((4n+m)+\sqrt{16n^2+m^2})\}$, hence M is positive definite.

(ii) Here we use mathematical induction on the total number of nonzero elements in A, B, and E, one more time. For $A = E = 0_{n\times n}$ and $B = 0_{n\times m}$, $p = 0$ and the system (2.2) is consistent.
Again, we assume that for A, B, and E, which have altogether N nonzero elements, the system (2.2) is consistent. Hence, there exists $x \in \mathbb{R}^{2n}$ such that $p = Lx$. When we add one more nonzero element to A, B, or E, then the new righthand side of the system (2.2) is $p+\alpha u$, where $\alpha = -log_{10}|a_{i_0j_0}|$ or $\alpha = -log_{10}|e_{i_0j_0}|$ with $u = \xi_{i_0}+\xi_{n+j_0}$ if the extra element is $a_{i_0j_0}$ or $e_{i_0j_0}$, and $\alpha = -log_{10}|b_{i_0j_0}|$ with $u = \xi_{i_0}$ if the extra element is $b_{i_0j_0}$. The new form of the system (2.2) is

$$(L + H)y = p + \alpha u, \qquad H = uu^T, \tag{2.5}$$

and the question now arises whether such $y \in \mathbb{R}^{2n}$ exists. If $L+H$ is a positive definite matrix then we have no problem in solving the system in (2.5). If $L + H$ remains positive semidefinite, then the problem divides into two cases.
The first case is when $u \notin \mathrm{Im}(L)$. In this situation we can write $u = u_1 + u_2$, where $u_1 \perp u_2$, $u_1 \in \mathrm{Im}(L)$, and $u_2 \in \mathrm{Ker}(L)$. It can be easily shown that

$$y = x + \frac{\alpha - u^T x}{u_2^T u_2} u_2$$

satisfies (2.5). In the second case $u \in \mathrm{Im}(L)$, hence there exists $v \in \mathbb{R}^{2n}$ such that $u = Lv$. Again, it can be easily shown that

$$y = x + \frac{\alpha - u^T x}{1 + v^T L v} v + z_0, \quad z_0 \in \mathrm{Ker}(L) \text{ arbitrary}$$

satisfies (2.5).

(iii) Direct multiplication proves that $MM^{-1} = M^{-1}M = I$ is satisfied.

□

Now we know that there is a solution to the equation $\nabla\phi(l, r) = 0$, and the Hessian $\mathrm{Hess}(\phi) = 2L$ is positive semidefinite, so this solution is indeed the global minimum of the function ϕ. Since we are dealing here with a symmetric positive semidefinite consistent system, we can apply the conjugate gradient method for obtaining its solution (see [13]). The conjugate gradient method is an iterative method specially suited for positive (semi-) definite matrices with a structure (such as the L), where the matrix–vector product is efficiently implemented. On the other hand, it is much more convenient to work with the matrix M, and for that we can employ the generalized conjugate gradient method from [5], as in the function `dggbal`. The generalized conjugate gradient method is in fact a preconditioned conjugate gradient method with the preconditioner equal to M. The solution of a system with the matrix M can be found directly, since we know the explicit expression for M^{-1}, and the product $M^{-1}v$ with a vector v requires only $O(n)$ operations. The convergence rate of this method depends on the number of different eigenvalues of $M^{-1}L$ (see [12]). For $L = M$ the method converges in only 1 iteration.

2.1. Increasing the weight of B. Since the matrix B has usually less elements than A and E, its influence on the minimization process is weaker. This can give an

ALGORITHM 1: Generalized conjugate gradient method for the system $Lx=p$.

Input: symmetric positive semidefinite $L \in \mathbb{R}^{2n\times 2n}$, preconditioner M, $p \in \mathrm{Im}(L)$
Output: solution $x \in \mathbb{R}^{2n}$ of the system $Lx = p$

$x_0 = s_0 = 0$;
repeat
  solve $Mz_k = p - Lx_k$;
  if $k = 0$ **then**
    $\beta_0 = 0$;
  else
    $\beta_k = \frac{z_k^T M z_k}{z_{k-1}^T M z_{k-1}}$;
  end
  $s_k = z_k + \beta_k s_{k-1}$;
  $\alpha_k = \frac{z_k^T M z_k}{s_k^T L s_k}$;
  $x_{k+1} = x_k + \alpha_k s_k$;
until *convergence*;

unsatisfactory result for B, where resulting A and E are much better balanced than B. This is not an issue when balancing involves matrices of the same dimensions as in [27] and [14]. Therefore, we are introducing here the weighted minimization function. To increase its influence, we can multiply the part in ϕ involving the matrix B with stronger weight. The matrices A and E have n^2 elements, and B only mn, thus the logical choice for weight is $\frac{n}{m}$. We introduce a new variant of ϕ as

$$\phi(l,r) = \sum_{i=1}^{n} \left[\sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|a_{ij}|)^2 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|e_{ij}|)^2 + \frac{n}{m} \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} (l_i + \log_{10}|b_{ij}|)^2 \right], \tag{2.6}$$

which changes the definition of the matrix F_1 and the vector c in (2.2) to

1. $F_1 \in \mathbb{R}^{n\times n}$ is a diagonal matrix $F_1 = \mathrm{diag}(n_{r_1}, \ldots, n_{r_n})$ and

$$n_{r_i} = \sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} 1 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} 1 + \frac{n}{m} \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} 1,$$

2. $c = [c_i] \in \mathbb{R}^n$ has elements

$$c_i = \sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} \log_{10}|a_{ij}| + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} \log_{10}|e_{ij}| + \frac{n}{m} \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} \log_{10}|b_{ij}|.$$

In case when the matrices A, B, and E contain only nonzero elements the system matrix in (2.2) reduces to

$$M = \begin{bmatrix} 3nI & 2ee^T \\ 2ee^T & 2nI \end{bmatrix}. \tag{2.7}$$

The (i) and (ii) part of Proposition 2.1 hold for weighted ϕ, too, but M^{-1} has a simpler form.

PROPOSITION 2.2.

(i) *The system matrix L of the system (2.2) obtained from (2.6) is symmetric positive semidefinite or positive definite.*

(ii) *If L is singular positive semidefinite, then this system is consistent.*

(iii) *The matrix M defined with (2.7) is symmetric positive definite, and its inverse is equal to*

$$M^{-1} = \begin{bmatrix} \frac{1}{3n}I + \frac{2}{3n^2}ee^T & -\frac{1}{n^2}ee^T \\ -\frac{1}{n^2}ee^T & \frac{1}{2n}I + \frac{1}{n^2}ee^T \end{bmatrix}. \tag{2.8}$$

Proof. The proof is the same as in Proposition 2.1. □

2.2. Balancing B from both sides. This variant of the balancing algorithm produces diagonal matrices D_l, D_r and D_B such that the range of magnitude orders of all elements in the matrices D_lAD_r, D_lED_r, and D_lBD_B is optimally small.

Let us denote $D_B = \mathrm{diag}(10^{q_1}, \ldots, 10^{q_m}) \in \mathbb{R}^{m\times m}$ and $q = (q_1, \ldots, q_m)$. The minimization problem is a bit more complicated than before:

$$\min_{l,r,q} \phi(l, r, q), \tag{2.9}$$

$$\phi(l,r,q) = \sum_{i=1}^{n} \left[\sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|a_{ij}|)^2 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} (l_i + r_j + \log_{10}|e_{ij}|)^2 + \sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} (l_i + q_j + \log_{10}|b_{ij}|)^2 \right].$$

Equalizing $\nabla\phi(l, r, q) = 0$ produces a linear system $Lx = p$ with the following form

$$\begin{bmatrix} F_1 & G & K \\ G^T & F_2 & 0 \\ K^T & 0 & F_3 \end{bmatrix} \begin{bmatrix} l \\ r \\ q \end{bmatrix} = \begin{bmatrix} -c \\ -d \\ -f \end{bmatrix}, \tag{2.10}$$

where F_1, F_2, G, c and d have the same form as in the original version of the balancing, and

1. $F_3 \in \mathbb{R}^{m\times m}$ is a diagonal matrix $F_3 = \mathrm{diag}(n_{cb_1}, \ldots, n_{cb_m})$ and

$$n_{cb_j} = \sum_{\substack{i=1 \\ b_{ij}\neq 0}}^{n} 1$$

is the total number of nonzero elements in the j-th column of B,

2. $K = [k_{ij}] \in \mathbb{R}^{n\times m}$ is the incidence matrix of B, i. e.

$$k_{ij} = \left\{ \begin{array}{ll} 1, & \text{if } b_{ij} \neq 0 \\ 0, & \text{if } b_{ij} = 0 \end{array} \right\},$$

3. $f = [f_j] \in \mathbb{R}^m$ has elements

$$f_j = \sum_{\substack{i=1 \\ b_{ij}\neq 0}}^{n} \log_{10}|b_{ij}|.$$

In the special case, when the matrices A, B, and E contain only nonzero elements, the system matrix in (2.10) reduces to

$$M = \begin{bmatrix} (2n+m)I_n & 2e_n e_n^T & e_n e_m^T \\ 2e_n e_n^T & 2nI_n & 0 \\ e_m e_n^T & 0 & nI_m \end{bmatrix}, \tag{2.11}$$

where $I_n \in \mathbb{R}^{n\times n}$ and $I_m \in \mathbb{R}^{m\times m}$ are the identity matrices, $e_n = [\ 1 \quad \dots \quad 1\ ]^T \in \mathbb{R}^n$, and $e_m = [\ 1 \quad \dots \quad 1\ ]^T \in \mathbb{R}^m$.

There is an alternative form of the minimization function ϕ in (2.9), where the part involving the matrix B has stronger weight, as in the previous subsection. Properties of its normal equations are similar to the properties of the system (2.10), which are given in the following proposition.

PROPOSITION 2.3.

(i) *The system matrix L of the system (2.10) is symmetric positive semidefinite and the system is consistent.*

(ii) *The matrix $M \in \mathbb{R}^{(2n+m)\times(2n+m)}$ defined with (2.11) is symmetric positive semidefinite of rank $2n+m-1$, and its Moore–Penrose generalized inverse is*

$$M^\dagger = \begin{bmatrix} \frac{1}{2n+m}I_n - \frac{3}{2(2n+m)^2}e_n e_n^T & \frac{n-m}{2n(2n+m)^2}e_n e_n^T & \frac{3}{2(2n+m)^2}e_n e_m^T \\ \frac{n-m}{2n(2n+m)^2}e_n e_n^T & \frac{1}{2n}I_n - \frac{3}{2(2n+m)^2}e_n e_n^T & \frac{-5n-m}{2n(2n+m)^2}e_n e_m^T \\ \frac{3}{2(2n+m)^2}e_m e_n^T & \frac{-5n-m}{2n(2n+m)^2}e_m e_n^T & \frac{1}{n}I_m + \frac{-7n-2m}{2n(2n+m)^2}e_m e_m^T \end{bmatrix}. \tag{2.12}$$

(iii) *In case when $L \neq M$, the following holds: $Ker(L)^\perp \subset Ker(M)^\perp$.*

Proof.

(i) The proof of this part of Proposition 2.3 is similar to the proof of the parts (i) and (ii) of Proposition 2.1.

(ii) A direct verification of the Moore–Penrose conditions proves the statement about $M^\dagger$. It can be easily verified that the vector

$$u_0 = \frac{1}{\sqrt{2n+m}} \begin{bmatrix} e_n \\ -e_n \\ -e_m \end{bmatrix}$$

is a unit eigenvector of M corresponding to the eigenvalue $\lambda_0 = 0$. Further, the following equation holds

$$MM^\dagger = I - u_0 u_0^T,$$

thus $\mathrm{Ker}(M) = \mathrm{span}\{u_0\}$ and $\mathrm{Im}(M) = \mathrm{Ker}(M)^\perp$ since M is symmetric.

(iii) Let us compute Lu_0:

$$Lu_0 = \frac{1}{\sqrt{2n+m}} \begin{bmatrix} F_1 & G & K \\ G^T & F_2 & 0 \\ K^T & 0 & F_3 \end{bmatrix} \begin{bmatrix} e_n \\ -e_n \\ -e_m \end{bmatrix} = \frac{1}{\sqrt{2n+m}} \begin{bmatrix} F_1 e_n - G e_n - K e_m \\ G^T e_n - F_2 e_n \\ K^T e_n - F_3 e_m \end{bmatrix},$$

whose components are

$$(Lu_0)(i) = \frac{1}{\sqrt{2n+m}} \left(n_{r_i} - \left(\sum_{\substack{j=1 \\ a_{ij}\neq 0}}^{n} 1 + \sum_{\substack{j=1 \\ e_{ij}\neq 0}}^{n} 1 \right) - \left(\sum_{\substack{j=1 \\ b_{ij}\neq 0}}^{m} 1 \right) \right) = 0,$$
$$i = 1, \ldots, n$$
$$(Lu_0)(n+j) = \frac{1}{\sqrt{2n+m}} \left(\sum_{\substack{i=1 \\ a_{ij}\neq 0}}^{n} 1 + \sum_{\substack{i=1 \\ e_{ij}\neq 0}}^{n} 1 - n_{c_j} \right) = 0, \quad j = 1, \ldots, n$$
$$(Lu_0)(2n+j) = \frac{1}{\sqrt{2n+m}} \left(\sum_{\substack{i=1 \\ b_{ij}\neq 0}}^{n} 1 - n_{cb_j} \right) = 0, \quad j = 1, \ldots, m$$

Hence, $u_0 \in \mathrm{Ker}(L)$ and $\mathrm{Ker}(M) \subset \mathrm{Ker}(L)$. The statement follows immediately by taking orthogonal complements.

□

Part (iii) of the Proposition 2.3 is important for line 3 of Algorithm 1. Since the system $Lx = p$ is consistent $p - Lx_k \in \mathrm{Im}(L) = \mathrm{Ker}(L)^\perp \subset \mathrm{Ker}(M)^\perp = \mathrm{Im}(M)$. Thus, the system $Mz_k = p - Lx_k$ is consistent and there exists its solution z_k. In the SLICOT routine `TG01AD` this might not be the case. The preconditioner used in this routine

$$M = 2 \begin{bmatrix} nI & ee^T \\ ee^T & nI \end{bmatrix}$$

is the same as in `dggbal` (see [27]) and is singular. On the other hand, the system matrix L is as in the original version of our algorithm for $m = 1$ and can be nonsingular. In some special cases it can happen that $0 \neq p - Lx_k \in \mathrm{Ker}(M)$, producing $z_k = 0$ and the algorithm will prematurely stop since $x_{k+1} = x_k$. Such an example is presented in Section 4.

In the cases of the quadratic eigenvalue problem $\lambda^2 Ax + \lambda Ex + Bx = 0$ and the algebraic linear system $(\sigma^2 A + \sigma B + C)x = b$, all three matrices are of the same size $A, B, E \in \mathbb{R}^{n\times n}$, and they all have to be balanced from both sides with the same diagonal matrices: $D_l A D_r$, $D_l E D_r$ and $D_l B D_r$. This case reduces to the Ward's balancing algorithm, described in [27], except that the elements of F_1, F_2, G, c and d equally include elements of all three matrices, and the matrix M is equal to the corresponding matrix in `dggbal` multiplied by factor of 3/2.

3. Details of the algorithm. There are several details in Algorithm 1 which can be improved in order to produce a more efficient software implementation. First, the right hand side of the system in line 3 is residual $r_k = p - Lx_k$, which can be computed by a recurrence induced by the recurrent computing of x_k. Since $x_0 = 0$, it follows that $r_0 = p$, and for $k \geq 1$, $r_{k+1} = r_k - \alpha_k L s_k$ (see [5]). Second, the matrices L, M, and M^{-1} or $M^\dagger$ are not stored, they are only used in products with vectors such as $z_k = M^{-1} r_k$ or $z_k = M^\dagger r_k$, $\gamma_k = z_k^T M z_k$, and $t_k = L s_k$. Since we proposed three variants of the balancing algorithm in this paper, each of these variants has its own form of these products. Thus, let us denote by: S – the original variant, W – the weighted variant, R – the variant where B is balanced from both sides. The products are computed explicitly for each variant, as in the LAPACK routine `dggbal`.

variant S

$$
\begin{aligned}
r_k &= \begin{bmatrix} r_{k,1} \\ r_{k,2} \end{bmatrix}, \quad r_{k,1} = r_k(1:n),\ r_{k,2} = r_k(n+1:2n), \\
z_k &= M^{-1} r_k \\
&= \begin{bmatrix} \frac{1}{2n+m} r_{k,1} + \left(\frac{2}{(2n+m)m} e_n^T r_{k,1} - \frac{1}{nm} e_n^T r_{k,2} \right) e_n \\ \frac{1}{2n} r_{k,2} + \left(-\frac{1}{nm} e_n^T r_{k,1} + \frac{1}{nm} e_n^T r_{k,2} \right) e_n \end{bmatrix}, \\
\gamma_k &= \frac{1}{2n+m} r_{k,1}^T r_{k,1} + \frac{1}{2n} r_{k,2}^T r_{k,2} - \frac{1}{(2n+m)n} (e_n^T r_{k,1})^2 + \frac{1}{nm} (e_n^T r_{k,1} - e_n^T r_{k,2})^2;
\end{aligned}
$$

variant W

$$
\begin{aligned}
r_k &= \begin{bmatrix} r_{k,1} \\ r_{k,2} \end{bmatrix}, \quad r_{k,1} = r_k(1:n),\ r_{k,2} = r_k(n+1:2n), \\
z_k &= M^{-1} r_k \\
&= \begin{bmatrix} \frac{1}{3n} r_{k,1} + \frac{1}{n^2} \left(\frac{2}{3} e_n^T r_{k,1} - e_n^T r_{k,2} \right) e_n \\ \frac{1}{2n} r_{k,2} + \frac{1}{n^2} \left(-e_n^T r_{k,1} + e_n^T r_{k,2} \right) e_n \end{bmatrix}, \\
\gamma_k &= \frac{1}{3n} r_{k,1}^T r_{k,1} + \frac{1}{2n} r_{k,2}^T r_{k,2} - \frac{1}{3n^2} (e_n^T r_{k,1})^2 + \frac{1}{n^2} (e_n^T r_{k,1} - e_n^T r_{k,2})^2;
\end{aligned}
$$

variant R

$$
\begin{aligned}
r_k &= \begin{bmatrix} r_{k,1} \\ r_{k,2} \\ r_{k,3} \end{bmatrix}, \quad r_{k,1} = r_k(1:n),\ r_{k,2} = r_k(n+1:2n),\ r_{k,3} = r_k(2n+1:2n+m), \\
z_k &= M^{\dagger} r_k \\
&= \begin{bmatrix} \frac{1}{2n+m} r_{k,1} + \left(-\frac{3}{2(2n+m)^2} e_n^T r_{k,1} + \frac{n-m}{2n(2n+m)^2} e_n^T r_{k,2} + \frac{3}{2(2n+m)^2} e_m^T r_{k,3} \right) e_n \\ \frac{1}{2n} r_{k,2} + \left(\frac{n-m}{2n(2n+m)^2} e_n^T r_{k,1} - \frac{3}{2(2n+m)^2} e_n^T r_{k,2} + \frac{-5n-m}{2n(2n+m)^2} e_m^T r_{k,3} \right) e_n \\ \frac{1}{n} r_{k,3} + \left(\frac{3}{2(2n+m)^2} e_n^T r_{k,1} + \frac{-5n-m}{2n(2n+m)^2} e_n^T r_{k,2} + \frac{-7n-2m}{2n(2n+m)^2} e_m^T r_{k,3} \right) e_m \end{bmatrix}, \\
\gamma_k &= \frac{1}{2n+m} r_{k,1}^T r_{k,1} + \frac{1}{2n} r_{k,2}^T r_{k,2} + \frac{1}{n} r_{k,3}^T r_{k,3} - \frac{3}{2(2n+m)^2} (e_n^T r_{k,2} + e_m^T r_{k,3} - \\
&\quad - e_n^T r_{k,1})^2 - \frac{1}{n(2n+m)} ((e_m^T r_{k,3})^2 + e_n^T r_{k,1} e_n^T r_{k,2} + e_n^T r_{k,2} e_m^T r_{k,3}).
\end{aligned}
$$

The vector t_k is computed as a sum of expressions $H_{ij} s_k$, where for each nonzero element of A, E or B with indices (i, j), H_{ij} is a matrix as defined in the proof of Proposition 2.1.

The algorithm requires extra memory storage for storing vectors r_k, s_k and t_k. The dimension of the workspace used for storing these auxiliary variables is $6n$ in case of the variants S and W, or $6n + 3m$ in case of the variant R.

4. Numerical tests. The tests were executed on the Intel® Core™ Duo CPU under Ubuntu Linux 10.04 (lucid). They were programmed in Fortran, compiled with Intel® Fortran Compiler 12.0.2, and all variables were in double precision or double complex precision. The balancing algorithms for three matrices are implemented in the routine `dg3bal`, whose code is available from the author.

4.1. Example 1: random scaled matrices A, E and B. The matrices were generated by starting with well scaled random matrices $A, E \in \mathbb{R}^{8\times 8}$ and $B \in \mathbb{R}^{8\times 3}$, and then by scaling rows and columns of A and E, and rows of B with the same diagonal matrices D_{lr}. We chose altogether 8 different matrices D_{lr} with a growing range in the magnitude of its diagonal elements: the first matrix $D_{lr} = \mathrm{diag}(1, 10, 1, 10, 1, 10, 1, 10, 1, 10)$ has the smallest range, and the last $D_{lr} = \mathrm{diag}(10^{-8}, 10^{-4}, 10^{0}, 10^{4}, 10^{8}, 10^{-8}, 10^{-4}, 10^{0})$ has the largest range. Besides that, the columns of B were scaled by $\mathrm{diag}(10^4, 10^8, 10^{12})$ in all examples. Hence, the rows of all three matrices and columns of A and E are gradually scaled, but the columns of B are badly scaled. We generated altogether 80 examples, with 10 different starting sets of random matrices for each choice of D_{lr}. The obtained results are illustrated in Figure 4.1. In all following figures the results for the matrix E are omitted since they are almost the same as the results for A.

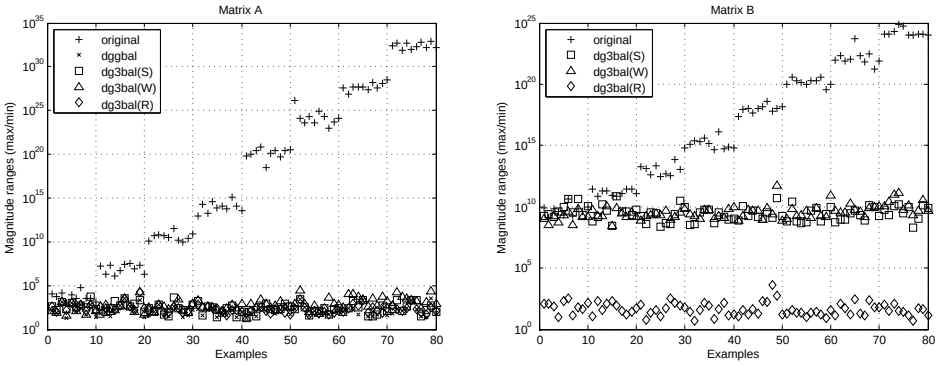


Fig. 4.1: Random scaled matrices: magnitude ranges of elements for the original matrices and the balanced matrices A and B.

The variant S of the balancing algorithm produced matrices $D_l A D_r$ and $D_l E D_r$ which are very similar to the matrices produced by the LAPACK routine `dggbal`. The rows of $D_l B$ are also well scaled as well, but the columns remain badly scaled. Thus, the range between the minimal and the maximal magnitude of elements in $D_l B$ is not as small as the ranges for $D_l A D_r$ and $D_l E D_r$. We can not expect more from balancing the matrix B only from the left. Weighted balancing for the matrix B does not have much sense for this example since rows of A, E and B are scaled with the same diagonal matrix. Thus, the inverses of the generating diagonal matrices balance the rows of all three matrices well. In the case of the variant R of the balancing algorithm, all three matrices are well balanced. The order of magnitude of the elements of $D_l A D_r$, $D_l E D_r$, and $D_l B D_B$ remained in a narrow range.

4.2. Example 2: B with heavily scaled rows. The matrices A and E are generated in the same way as ones in Example 1. While the rows of A and E are scaled with the same matrices D_{lr}, the rows of B are scaled with different diagonal matrices D_{lB} whose diagonal elements have wider range in magnitude: $D_{lB}(i,i) = D_{lr}(i,i)^3$. The obtained results are illustrated in Figure 4.2. The variant W should have a stronger influence on this example.

As we can see, the variants S and R produced similar results, since the columns of B are not scaled. Both variants were not completely successful in balancing the

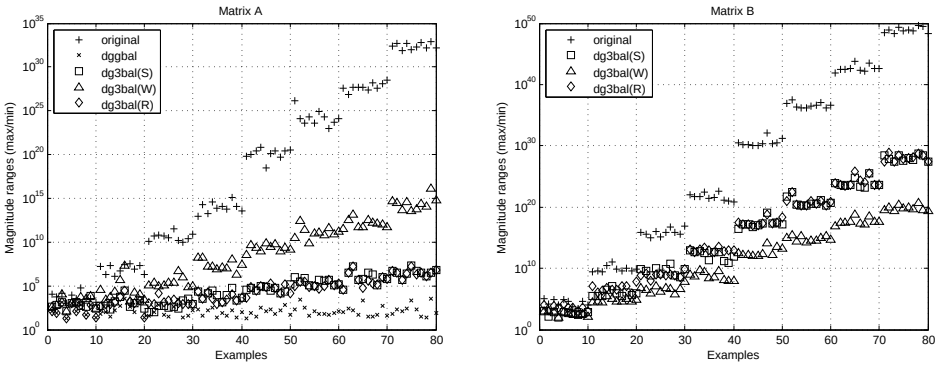


Fig. 4.2: B with heavily scaled rows: magnitude ranges of elements for the original matrices and the balanced matrices A and B.

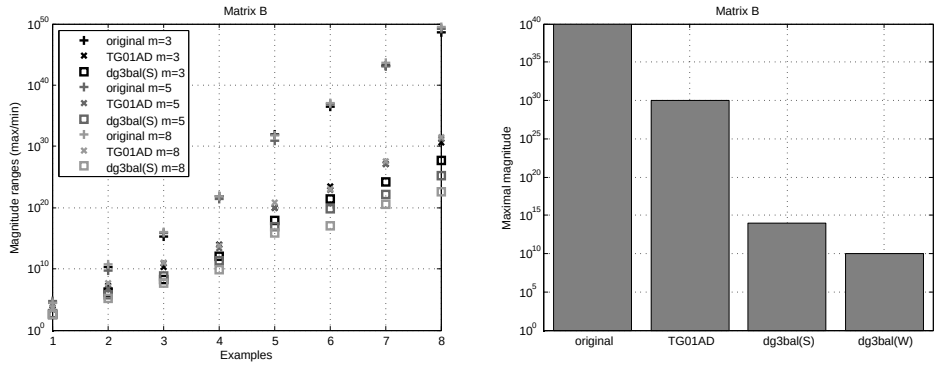
matrix B. The variant W is slightly better for B, while in this case the balanced matrices A and E have more scattered elements than the obtained balanced matrices for the variants S and R.

4.3. Example 3: `dg3bal` vs. `TG01AD`. Here we show the superiority of our balancing routine `dg3bal` over the SLICOT routine `TG01AD`. In the first test rounds we generated matrices A, B, and E as in Example 2, but with one example for each choice of diagonal matrices. The only difference is that we changed the number of columns of B, taking $m = 3, 5, 8$. The results presented in Figure 4.3a show that `dg3bal` balances elements of B better than `TG01AD`, specially for larger m when B has larger influence in the minimization function of `dg3bal`.

The second test round comprises of one set of matrices A, B, and E, where $n = 1000$ and $m = 10$. A and E are generated by scaling random matrices with the matrix $D_{lr} = \mathrm{diag}(10, 10^2, 10^3, \ldots, 10^{10}, 10, 10^2, 10^3, \ldots, 10^{10})$ from both sides, and B is generated by scaling a random matrix with $D_{lB} = \mathrm{diag}(10^4, 10^8, 10^{12}, \ldots, 10^{40}, 10^4, 10^8, 10^{12}, \ldots, 10^{40})$ from the left. In this case Figure 4.3b displays the maximal magnitude of elements in B, which maximally influences $\|B\|_F$. The magnitude of the norm is extremely important for the rank revealing algorithms, deployed in the staircase reduction routines, which are used to determine the controllable part of the system (1.1). These are the SLICOT routines `TG01HD` and `TG01HX`, and the new staircase reduction algorithm from [3]. The standard tolerance for rank determination is $n^2 u \sqrt{\|A\|_F^2 + \|B\|_F^2}$, where u is the unit roundoff error. When the norms of A and B are large, the numerical rank is usually smaller than the exact rank. In extreme cases it turns out to be equal to zero for nontrivial submatrices. Detailed illustration of sensitivity of the staircase reduction is given in Example 5. Figure 4.3b shows that the S and W variants of `dg3bal` are more successful in norm reduction of B than `TG01AD`. The maximal magnitude of elements in A and E are of order: 1 for `TG01AD`, 10 for variant S of `dg3bal`, and 10^6 for variant W of `dg3bal`.

The last test round in this example is concerned with the problem of the singular preconditioner in the routine `TG01AD`. The matrices A, E, and B are defined as follows:

$$A = \begin{bmatrix} 10^{-2} & 0 & 10^{-4} \\ 0 & 10^{-4} & 10^{4} \\ 10^{-2} & 0 & 10^{-4} \end{bmatrix}, \quad E = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 10^{10} \\ 10^{4} \\ 10^{10} \end{bmatrix},$$



(a) Magnitude ranges of elements for the original matrix and the balanced matrix B.

(b) Maximal magnitude of elements for the original matrix and the balanced matrix B.

Fig. 4.3: `dg3bal` vs. `TG01AD`: reduction of the magnitude range and maximal magnitude of elements in B.

and $\sqrt{\|A\|_F^2 + \|B\|_2} = 1.414214 \cdot 10^{10}$. In this case `TG01AD` generates the same matrix L and the vector p as the variant S of `dg3bal`, but M is the same as in `dggbal`:

$$L = \begin{bmatrix} 5 & 0 & 0 & 2 & 0 & 2 \\ 0 & 5 & 0 & 0 & 2 & 2 \\ 0 & 0 & 5 & 2 & 0 & 2 \\ 2 & 0 & 2 & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 2 & 2 & 0 & 0 & 6 \end{bmatrix}, \quad p = \begin{bmatrix} -4 \\ -4 \\ -4 \\ 4 \\ 4 \\ 4 \end{bmatrix}, \quad M = \begin{bmatrix} 6 & 0 & 0 & 2 & 2 & 2 \\ 0 & 6 & 0 & 2 & 2 & 2 \\ 0 & 0 & 6 & 2 & 2 & 2 \\ 2 & 2 & 2 & 6 & 0 & 0 \\ 2 & 2 & 2 & 0 & 6 & 0 \\ 2 & 2 & 2 & 0 & 0 & 6 \end{bmatrix}.$$

It is easy to check that $Mp = M^\dagger p = 0$. The first step of the conjugate gradient method is to compute $z_0 = M^\dagger p$. Since $z_0 = 0$, it implies $s_0 = 0$, $\alpha_0 = 0$, and $x_1 = x_0$ which will satisfy the stopping criterion and the routine stops with unchanged data. The problem arises due to $p \in \mathrm{Ker}(M)$. On the other hand the S variant of `dg3bal` produces

$$A_b = \begin{bmatrix} 10^{-1} & 0 & 10^{-3} \\ 0 & 10^{-2} & 10^{5} \\ 10^{-1} & 0 & 10^{-3} \end{bmatrix}, \quad E_b = \begin{bmatrix} 10 & 0 & 10 \\ 0 & 100 & 10 \\ 10 & 0 & 10 \end{bmatrix}, \quad B_b = \begin{bmatrix} 10^{2} \\ 10^{-4} \\ 10^{2} \end{bmatrix},$$

where $\sqrt{\|A_b\|_F^2 + \|B_b\|_2} = 1.000001 \cdot 10^5$. The original and the balanced matrices are now used as input to the routine `TG01HD`. For the system (1.1), defined by the original matrices A, B, and E, the routine returns that the system has 1 finite controllable and 2 finite uncontrollable poles (poles are generalized eigenvalues of the pencil $A - \lambda E$). For the balanced system it turns out to have 2 finite controllable and 1 infinite uncontrollable pole. Thus, since `TG01AD` stopped prematurely it does not change the original matrices, and `TG01HD` gives the wrong answer about the controllable poles. For the balanced system returned by `dg3bal`, `TG01HD` returns the correct answer. Since E is obviously singular, there has to be an infinite pole.

4.4. Example 4: sensitivity of frequency response computation to scaling. In this example we demonstrate how badly scaled matrices can produce an inaccurate frequency response matrix. We started with the descriptor system (A_0,B_0,C_0,

D_0,E_0), where $A_0, E_0 \in \mathbb{R}^{4\times4}$, $B_0 \in \mathbb{R}^{4\times1}$, $C_0 \in \mathbb{R}^{1\times4}$, $D_0 = 0$, and whose pole is placed near $0.4518i$. Then, we produced badly scaled matrices $E = D_{l,0}E_0D_{r,0}$, $A = D_{l,0}A_0D_{r,0}$, $B = D_{l,0}B_0$, $C = C_0D_{r,0}$, and $D = D_0$ where $D_{l,0}$ and $D_{r,0}$ are badly scaled diagonal matrices. We observed three different choices of these diagonal matrices. For each choice of the diagonal matrices we computed frequency response matrices for the original and the balanced system, where the m-Hessenberg–triangular–triangular reduction of A, B, and E from [3] was the first step in the algorithm, and σ was ranging from $10^{-2}i$ up to $10^{2}i$.

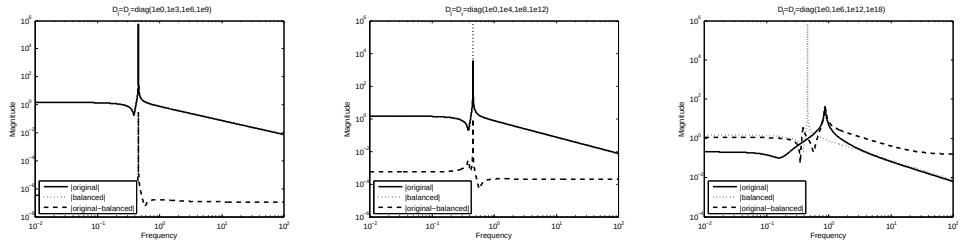


Fig. 4.4: Magnitude of the computed frequency response matrices, for the original and the balanced system: $\mathbf{D_{l,0} = D_{r,0}} = \mathrm{diag}(\mathbf{10^0}, \mathbf{10^3}, \mathbf{10^6}, \mathbf{10^9})$, $\mathbf{D_{l,0} = D_{r,0}} = \mathrm{diag}(\mathbf{10^0}, \mathbf{10^4}, \mathbf{10^8}, \mathbf{10^{12}})$, and $\mathbf{D_{l,0} = D_{r,0}} = \mathrm{diag}(\mathbf{10^0}, \mathbf{10^6}, \mathbf{10^{12}}, \mathbf{10^{18}})$.

We compared the computed frequency response matrices of the original system and the balanced system obtained by the variant S of the balancing algorithm. The obtained results are illustrated in Figure 4.4.

As we can see, as the matrices $D_{l,0}$ and $D_{r,0}$ are gradually becoming worse scaled, the produced frequency response matrix is becoming more inaccurate. For the first choice we obtained 6–7 accurate leading digits when compared with the result for the balanced system, while for the second choice there are 3–4 accurate digits. Specially, for the third choice, the magnitudes of the errors were of the same order as the magnitudes of the results, or larger, producing a results with no accurate digit for the original system.

Let us emphasize here that SLICOT has only the routine `TB05AD` which computes the frequency response matrix for a system with $E = I$. In that case balancing can be applied via the LAPACK routine `dgebal`. `dgebal` balances only the matrix A by a diagonal similarity transformation, since `TB05AD` reduces only the matrix A to the Hessenberg form. In MATLAB the frequency response matrix can be computed for a general descriptor system by the Hessenberg–triangular reduction of matrices A and E, implemented in the routine `freqresp`. The MATLAB routine seems to balance only the matrices A and E. The output of `freqresp` for the given example is indistinguishable from the result of our routine applied on the balanced system. Our algorithm based on m-Hessenberg–triangular–triangular reduction of A, B and E is more efficient than the algorithm based only on the Hessenberg–triangular reduction, and this is the reason why we need a balancing algorithm for three matrices.

4.5. Example 5: sensitivity of the staircase reduction. As mentioned earlier, the staircase reduction is a tool used to determine the controllable part of the system (1.1), see for example [16] and [26]. This problem is numerically very sensitive, since it relies on rank revealing algorithms. We started again with random matrices

$A_0, E_0 \in \mathbb{R}^{15\times 15}$ and $B_0 \in \mathbb{R}^{15\times 3}$, and generated three sets of badly scaled matrices $A_i = D_i A_0 D_i$, $E_i = D_i E_0 D_i$, $B_i = D_i B_0$, for $i = 1, 2, 3$, such that

$$D_1 = \mathrm{diag}(1, 10^2, 1, 1, 10^4, 1, 1, 10^6, 1, 1, 10^8, 1, 1, 10^9, 1)$$
$$D_2 = \mathrm{diag}(1, 10^3, 1, 1, 10^6, 1, 1, 10^9, 1, 1, 10^{12}, 1, 1, 10^{14}, 1)$$
$$D_3 = \mathrm{diag}(1, 10^3, 1, 1, 10^6, 1, 1, 10^9, 1, 1, 10^{12}, 1, 1, 10^{16}, 1)$$

We applied the SLICOT routine `TG01HD` to all four examples, and obtained four different results, presented in the following table.

example	# of contr. poles	# of fin. uncontr. poles	# of infin. uncontr. poles
0	15	0	0
1	3	9	1
2	1	14	0
3	0	15	0

The result for A_0, E_0, and B_0 is correct, while all the others are incorrect due to large norms of A_i and E_i. As the norms of these two matrices grow, the dimension of the controllable part is decreasing. Specially in case of the last example, `TG01HD` exited immediately without finding the controllable part.

The SLICOT routine `TG01HD` recommends balancing the system by the routine `TG01AD`.

4.6. Example 6: sensitivity of the pole assignment problem to scaling. We are interested in another problem from control theory: the pole assignment problem for descriptor linear systems of form (1.1) via state feedback. For details, see [10]. Let us define the closed-loop system

$$E\dot{x} = (A - BK)x(t) + Bv(t), \tag{4.1}$$

where $v(t)$ is an external signal, and $K \in \mathbb{R}^{m\times n}$ is a feedback matrix. For the regular system (1.1) with $n_1 = \deg\det(A - \lambda E)$, and for the set of complex numbers $\Gamma = \{\lambda_1, \lambda_2, \ldots, \lambda_{n_1}\}$ closed under complex conjugation, the problem is to find a state feedback controller in the form $u(t) = Kx(t) + v(t)$ such that Γ is the set of finite poles of the closed-loop system (4.1), or alternatively, the elements of Γ are the eigenvalues of the pencil $(A - BK) - \lambda E$. Reliable methods for solving this problem are the so-called Hessenberg methods based on explicit or implicit QZ-like techniques. An explicit shift method for single input systems is proposed by Miminis and Paige [15], and the implicit version of the algorithm for ordinary linear time invariant systems with multiple inputs is proposed by Patel and Misra [19]. We applied the QZ version of the Patel and Misra algorithm on a controllable descriptor system with a non-singular matrix E. This algorithm is based on a reduction, similar to the m-Hessenberg–triangular–reduction, which reveals controllability of the system.

Our example of the descriptor system (1.1) is defined for $A, E \in \mathbb{R}^{10\times 10}$ and $B \in \mathbb{R}^{10\times 3}$, where

$$A = \begin{bmatrix}
8.15\cdot10^{-8} & 0 & 0 & 7.06\cdot10^{-3} & 0 & 0 & 7.51\cdot10^{-5} & 8.41\cdot10^{-8} & 0 & 7.59\cdot10^{-2} \\
0 & 9.71\cdot10^{-8} & 0 & 0 & 3.82\cdot10^{-1} & 0 & 2.55\cdot10^{2} & 0 & 8.31\cdot10^{-7} & 0 \\
0 & 9.57\cdot10^{-5} & 8.49\cdot10^{2} & 0 & 0 & 6.55\cdot10^{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.63\cdot10^{-1} & 0 & 2.44\cdot10^{-1} & 0 & 7.79\cdot10^{6} \\
0 & 0 & 0 & 9.71\cdot10^{3} & 1.87\cdot10^{-1} & 1.19\cdot10^{-1} & 0 & 0 & 9.17\cdot10^{-7} & 0 \\
0 & 0 & 0 & 0 & 4.90\cdot10^{-1} & 0 & 0 & 0 & 2.86\cdot10^{-7} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.69\cdot10^{6} \\
0 & 0 & 0 & 0 & 0 & 0 & 1.39\cdot10^{-2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6.16\cdot10^{14} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.68\cdot10^{-7} & 3.37\cdot10^{6}
\end{bmatrix},$$

$$E = \begin{bmatrix}
4.31\cdot10^{-8} & 0 & 4.17\cdot10^{-8} & 0 & 2.35\cdot10^{-8} & 0 & 0 & 6.44\cdot10^{-8} & 2.08\cdot10^{-14} & 0 \\
0 & 6.22\cdot10^{-8} & 0 & 3.90\cdot10^{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 9.03\cdot10^{2} & 0 & 0 & 0 & 4.87\cdot10^{5} & 0 & 0 & 4.30\cdot10^{9} \\
0 & 0 & 0 & 4.04\cdot10^{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4.30\cdot10^{-2} & 6.87\cdot10^{-1} & 0 & 3.51\cdot10^{-1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.84\cdot10^{-1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 5.09\cdot10^{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.50\cdot10^{-5} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.28\cdot10^{8} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.09\cdot10^{6}
\end{bmatrix},$$

$$B^T = \begin{bmatrix}
1.62\cdot10^{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4.51\cdot10^{1} & 0 & 0 & 9.13\cdot10^{-1} & 0 & 8.26\cdot10^{5} & 0 & 0 & 0 & 0 \\
0 & 9.62\cdot10^{-1} & 0 & 0 & 8.17\cdot10^{-1} & 0 & 0 & 0 & 2.60\cdot10^{13} & 8.00\cdot10^{-1}
\end{bmatrix}.$$

The set of desired finite poles of the closed-loop system is taken to be $\Gamma = \{-1 \pm 2i, -8 \pm 12i, -3, -7, -15, -25, -30, -100\}$. The pole assignment algorithm produced the feedback matrix

$$K = \begin{bmatrix}
1.30\cdot10^{-8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -4.74\cdot10^{-13} & 1.07\cdot10^{-15} & -2.31\cdot10^{-19} & 7.53\cdot10^{-7} & -4.72\cdot10^{-8} & 0 & 0 & 0 & 0 \\
0 & -1.54\cdot10^{-17} & -4.41\cdot10^{-10} & 3.33\cdot10^{-16} & 1.84\cdot10^{-11} & 2.93\cdot10^{-10} & 0 & 0 & 6.13\cdot10^{-5} & 1.05\cdot10^{-16}
\end{bmatrix},$$

and the computed eigenvalues of the pencil $(A - BK) - \lambda E$ are $-3.0000, 8.6823 \pm 12.732i, -6.7362, -0.13415, 0.40521, 0.94064, -2.4631, 1.5121 \pm 1.7799i$, see Figure 4.5.

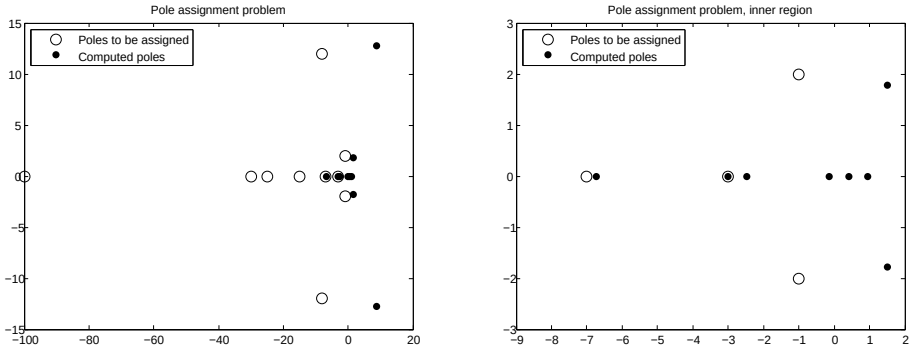


Fig. 4.5: Desired poles in Γ and the computed poles.

It has to be mentioned here that the matrices $A - BK$ and E were balanced prior to eigenvalue computation, and as we can see, only one eigenvalue was assigned correctly.

On the other hand, when we applied the balancing algorithm on the matrices A, B and E, obtaining $D_l = \mathrm{diag}(10^3, 10^{-2}, 10^{-6}, 10^{-2}, 10^{-2}, 10^{-3}, 10^{-3}, 10^{1}, 10^{-16}, 10^{-2})$ and $D_r = \mathrm{diag}(10^4, 10^{10}, 10^4, 10^{-2}, 10^3, 10^3, 10^1, 10^3, 10^9, 10^{-4})$, and the balanced matrices $A_b = D_l A D_r$, $B_b = D_l B$, $E_b = D_l E D_r$, the pole assignment algorithm applied to the balanced matrices produced the feedback matrix

$$K_b = \begin{bmatrix} 1.30\cdot10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.49\cdot10^{-1} & -6.66\cdot10^{0} & -1.00\cdot10^{0} & -1.28\cdot10^{-1} & -2.04\cdot10^{0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.91\cdot10^{8} & 2.93\cdot10^{8} & 7.45\cdot10^{8} & -4.3341\cdot10^{7} \end{bmatrix}.$$

In this case, all computed eigenvalues of the pencil $A_b - B_b K_b - \lambda E_b$ have at least 8 correct digits. We also observed that, in our experiments, where the columns of B were badly scaled the system became numerically uncontrollable, reducing the number of eigenvalues that can be assigned. Thus, the variant R of the balancing algorithm is applicable and useful for this problem.

4.7. Example 7: An example from a real application. In the last subsection we will illustrate sensitivity issues of a badly scaled system coming from a real application. The descriptor system of the form (1.1) is derived as a power system model, and this example it is based on the Brazilian interconnection power systems (BIPS) model relating to a 1998 heavy load condition. Matrices of the system can be found as example `bips98_606`, which is a part of Rommes group in UF Sparse Matrix Collection [21]. The dimensions are: $n = 7135$ and $m = p = 4$. The matrix A is very badly scaled, with the smallest element equal to $1.4341 \cdot 10^{-17}$ and the largest equal to 10^{20} on several entries, causing a large condition number $\kappa_2(A) = 3.1321 \cdot 10^{26}$. B and E are well scaled. First we apply our balancing routine `dg3bal` to this system, and then we are going to compare the results of the frequency response computation and staircase reduction of the original and the balanced systems. The routine `dg3bal` balanced the matrix A successfully. The obtained balanced matrix A_b has the smallest element equal to $1.1374 \cdot 10^{-6}$ and the largest equal to $1.0009 \cdot 10^{4}$, while the condition number is reduced to $\kappa_2(A_b) = 2.4964$. The balanced matrices B_b and E_b remained well scaled.

We computed frequency response matrices for shifts σ ranging from 10^{-2} to 10^4. The entries of the frequency response matrices for the original system differ from the corresponding elements of the frequency response matrices for the balanced system after 2 to 8 leading digits. This result is not as bad as it can be expected from the ill-conditioned matrix A, and can be explained by the sparsity pattern of B. On the other hand, since A has a large norm the real effect of the balancing is observed when trying to determine the controllable part of the original system. When applied to the original system, the routine `TG01HD` returned without finding the controllable part, while when applied to the balanced system, the routine returned the controllable part of dimension 616. Clearly, we would not be able to solve this problem for this particular example without balancing.

5. Conclusion. In this paper three versions of the algorithm for balancing three matrices simultaneously are proposed. The balancing is performed via diagonal transformations and the goal is to reduce the range of order of magnitude for all elements of the involved matrices. We illustrated its application with the reduction to the m-Hessenberg–triangular–triangular form of three matrices A, B and E, which is used for efficient computation of the frequency response matrix $\mathcal{G}(\sigma) = C(\sigma E - A)^{-1}B + D$ in case of a descriptor system, with the pole assignment problem via state feedback,

and with finding the controllable part of the system. The reduction algorithm can produce a very inaccurate result for badly scaled matrices. The basic variant balances rows and columns of A and E, and only rows of B, since computing $\mathcal{G}(\sigma)$ is invariant under such transformations. The weighted variant of the balancing algorithm gives more weight to the balancing of elements of B, since the basic algorithm can produce well balanced A and E and poorly balanced B. The third variant offers a possibility of balancing columns of B as well. Numerical experiments confirmed that balancing matrices A, B and E before the m-Hessenberg–triangular–triangular reduction produces an accurate frequency response matrix, as well as accurate pole assignment via state feedback. In case when the controllable part of the system is sought, the answer might not be obtained when the system is badly scaled. The balancing is very important for this kind of problem.

Acknowledgments. The author is indebted to Zlatko Drmač for reading the paper and giving many valuable comments.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. C. Sorensen, *LAPACK Users' Guide*, Third ed., SIAM, Philadelphia, 1999.

[2] Å. Björck, *Least squares methods*, in Handbook of Numerical Analysis, Volume I, P. G. Ciarlet et al., eds., North-Holland, Amsterdam, 1990, pp. 465–652.

[3] N. Bosner, *Efficient algorithm for simultaneous reduction to the m-Hessenberg–triangular–triangular form*, Tech. rep., University of Zagreb, 2011.

[4] N. Bosner, Z. Bujanović, and Z. Drmač, *Efficient generalized Hessenberg form and applications*, ACM Trans. Math. Softw., 39 (2013).

[5] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Comput., Proc. Symp. Lemont 1975, 1976, pp. 309–332.

[6] A. J. Cox and N. J. Higham, *Stability of Householder QR factorization for weighted least squares problems*, In: D.F. Griffiths, D.J. Higham, G.A. Watson (Eds.), Numerical Analysis 1997, Proceedings of the 17th Dundee Biennial Conference, Pitman Research Notes in Mathematics, vol. 380, Addison-Wesley, Longman, Harlow, Essex, UK, 1998, pp. 57-73.

[7] A. R. Curtis and J. K. Reid, *On the automatic scaling of matrices for Gaussian elimination*, J. Inst. Math. Appl., 10 (1972), pp. 118–124.

[8] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[9] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK User's Guide*, SIAM, Philadelphia, 1979.

[10] G.-R. Duan, *Analysis and Design of Descriptor Linear Systems*, Springer, New York, 2010.

[11] G. H. Golub and C. F. van Loan, *Matrix Computations*, Third ed., M. D. Johns Hopkins University Press, Baltimore, 1996.

[12] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997, Chapter 3.

[13] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.

[14] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 253–263.

[15] G. S. Miminis and C. C. Paige, *An algorithm for pole assignment of time invariant linear systems*, Int. J. Control, 35 (1982), pp. 341–354.

[16] C. C. Paige, *Properties of numerical algorithms related to computing controllability*, IEEE Trans. Autom. Control, 26 (1981), pp. 130–138.

[17] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall Series in Computational Mathematics, Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[18] B. N. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.

[19] R. V. Patel and P. Misra, *Numerical algorithm for eigenvalue assignment by state feedback*, Proc. IEEE, 72 (1984), pp. 1755-1764.
[20] M. J. D. Powell and J. K. Reid, *On applying Householder transformations to linear least squares problems*, Tech. report T.P. 322, Mathematics Branch, Theoretical Physics Division, Atomic Energy Research Establishment, Harwell, UK, February 1968.
[21] Rommes group, `http://www.cise.ufl.edu/research/sparse/matrices/Rommes/`
[22] V. Simoncini, *Restarted full orthogonalization method for shifted linear systems*, BIT, 43 (2003), pp. 459–466.
[23] V. Simoncini and F. Perotti, *On the numerical solution of* $(\lambda^2 A + \lambda B + C)x = b$ *and application to structural dynamics*, SIAM J. Scientific Comput., 23 (2002), pp. 1876-1898.
[24] SLICOT, `http://www.slicot.org`
[25] A. van der Sluis, *Condition numbers and equilibration of matrices*, Numer. Math., 14 (1969), pp. 14–23.
[26] A. Varga, *Computation of irreducible generalized state-space realizations*, Kybernetika, 26 (1990), pp. 89–106.
[27] R. C. Ward, *Balancing the generalized eigenvalue problem*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 141–152.
[28] D. S. Watkins, *A case where balancing is harmful*, ETNA, Electron. Trans. Numer. Anal., 23 (2006), pp. 1–4.