Math. Commun. **19**(2014), 497–516

Balancing three matrices in control theory*

Nela Bosner[1,†]

[1] *Department of Mathematics, University of Zagreb, Bijenička cesta 30, HR-10 000 Zagreb, Croatia*

Received October 16, 2013; accepted February 11, 2014

Abstract. Several problems from control theory are presented which are sensitive to badly scaled matrices. We were specially concerned with the algorithms involving three matrices, thus we extended the Ward's balancing algorithm for two matrices. Numerical experiments confirmed that balancing three matrices can produce an accurate frequency response matrix for descriptor linear systems, it can also improve the solution of the pole assignment problem via state feedback and the determination of the controllable part of the system.

AMS subject classifications: 65F35, 65Y04

Key words: balancing three matrices, diagonal transformations, numerical stability, frequency response matrix, pole assignment problem, controllable part of the system

1. Introduction

When applied to matrices with a wide range in the magnitude of elements a numerically stable algorithm 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 [8]. As emphasized in the introduction of [10], 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 a 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 [20]. 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 [29], and by Lemonnier and Van Dooren in [16]. It is important to emphasize here that in most cases balancing will improve condition numbers, and the balanced problems will produce more accurate

*This work is supported by the Croatian Ministry of Science, Education and Sports grant 0372783–2750.
†Corresponding author. *Email address:* nela@math.hr (N. Bosner)

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 [30].

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 [9]. In the case of a positive definite matrix A, the condition number for eigenvalues is bounded by $n \cdot \min_{D \text{ is diagonal}} \kappa(DAD)$, and the condition number for singular values is bounded by $\sqrt{n} \cdot \min_{D \text{ is diagonal}} \kappa(AD)$ (see van der Sluis [27]), 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

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

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 $m \leq n$ (usually, it is $m \ll n$). E is often singular. As pointed out by Paige in [18], 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) in the sense of restricted state-space equivalence (i.e., they have the same transfer function). 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 it 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 system (1) with a general matrix E, the scaling has to include three matrices A, B and E.

In [4] and [5], an efficient algorithm for computing the frequency response matrix $\mathcal{G}(\sigma) = C(\sigma E - A)^{-1}B + D$ of system (1) is proposed for a large number of shifts σ. The first part of this algorithm comprises 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 [4]. 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 the order of magnitude. In floating point arithmetic the influence of the number with a small order of magnitude can be completely lost in the sum. Although the orthogonal transformation always produces a result with a small norm-wise relative error, in case of badly scaled matrices they can severely increase the element-wise error (for QR factorization see [15, Section 19.4]), or they can transform a nonsingular matrix into a singular one. For example, in [22], Powell and Reid 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 a desirable backward error, and in [7], Cox and Higham proved that sorting the rows by a 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. 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 [18] (see also [4]), an algorithm that reveals controllability of system (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 the exact matrices 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. This is a typical situation in practice, where the input data represent measured physical quantities. These quantities may be represented in different units, hence the element in matrices may have a wide range in magnitude. 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, 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.

It is important to emphasize here that balancing is always applied to the original input data before any other transformation. Suppose that the numerically singular matrix E presented in the previous example is obtained from a singular matrix E_0 by some non exact numerical algorithm. Then the reasoning of the rest of the example does not hold for this situation, because balancing is not going to be applied to E. The element $E_{3,3}$ is equal to ε instead of 0, and represents the error introduced by floating point arithmetic. Balancing in this case would only increase the error by the factor of 10^6. On the other hand, balancing performed on E_0 would result with a singular matrix $E_{bal,0}$, and the same numerical algorithm applied to $E_{bal,0}$ would again produce a numerically singular matrix.

Besides control theory, there are other examples involving three matrices of the 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, [24] and [25]).

1.1. Balancing two matrices

There are two different approaches to balancing two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$. In [16] , Lemonnier and Van Dooren developed a balancing algorithm

suitable for general eigenvalue computing. The main idea is to reduce the relative condition number of the computed eigenvalues by diagonal scaling D_lAD_r and D_lBD_r. They showed that this is achieved when $\|D_lAD_re_j\|_2^2 + \|D_lBD_re_j\|_2^2 = \|e_i^TD_lAD_r\|_2^2 + \|e_i^TD_lBD_r\|_2^2 = 1$ for all i,j, and their method is based on a coordinate descent method, which can have slow convergence. Ward [29] proposes a different balancing technique which applies to two matrices involved in the generalized eigenvalue problem, but this technique is applicable to other problems with two matrices as well. This technique produces two diagonal matrices D_l and D_r such that the range of elements in D_lAD_r and D_lBD_r is favorably small. The basic idea is forcing the exponents in exponential notation of all nonzero elements in D_lAD_r and D_lBD_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 $(D_lAD_r)_{ij} = 10^{l_i+r_j}a_{ij}$ and $(D_lBD_r)_{ij} = 10^{l_i+r_j}b_{ij}$. 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

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

is minimized using a generalized conjugate gradient method developed by Concus et al. [6]. 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 3. Since we are going to apply balancing to several different problems, and since the Lemonnier and Van Dooren's approach is specially tailored for the eigenvalue problem, the Ward's approach seems more suitable for our task. We will just mention here that in [2] Betcke generalized Lemonnier and Van Dooren's method to balancing an arbitrary number of matrices of the same dimensions involved in the polynomial eigenvalue problem.

In the Control and Systems Library SLICOT [26], 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 3.

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. We can optionally 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

$$\min_{l,r \in \mathbb{R}^n} \phi(l, r), \tag{2}$$

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

To find a solution of the minimization problem (2), we differentiate the function $\phi(l, r)$ and equalize its gradient with zero, as in [29]. 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

$$L = \begin{bmatrix} F_1 & G \\ G^T & F_2 \end{bmatrix}, \quad p = \begin{bmatrix} -c \\ -d \end{bmatrix}, \quad x = \begin{bmatrix} l \\ r \end{bmatrix}, \tag{3}$$

where

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

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

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

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

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

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

- $d = [d_j] \in \mathbb{R}^n$ 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 (3) reduces to

$$M = \begin{bmatrix} (2n+m)I_n & 2e_n e_n^T \\ 2e_n e_n^T & 2nI_n \end{bmatrix}, \tag{4}$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix and $e_n = [\,1 \ldots 1\,]^T \in \mathbb{R}^n$.

Minimization problem (2) is in fact a linear least squares problem, and (3) represent its system of normal equations. We know that the matrix of normal equations is positive semidefinite, and that the system is consistent (see [12], [3]). The system matrix L of system (3) is symmetric positive semidefinite or positive definite. The matrix M defined by (4) is symmetric positive definite, and its inverse is equal to

$$M^{-1} = \begin{bmatrix} \frac{1}{2n+m} I_n + \frac{2}{(2n+m)m} e_n e_n^T & -\frac{1}{nm} e_n e_n^T \\ -\frac{1}{nm} e_n e_n^T & \frac{1}{2n} I_n + \frac{1}{nm} e_n e_n^T \end{bmatrix}. \tag{5}$$

Now we can conclude 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 [14]). The conjugate gradient method is an iterative method specially suited for positive (semi-) definite matrices with a structure (such as L), where the matrix–vector product is efficiently implemented. For our purpose most convenient is the usage of a preconditioned conjugate gradient method with M as the preconditioner (see, for example, [13]), presented in Algorithm 1. 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 [13]). For $L = M$, the method converges in only 1 iteration.

Algorithm 1 (Preconditioned conjugate gradient method for the system $Lx = p$.)**.**
(1) $x_0 = 0$*;*
(2) $r_0 = p$*;*
(3) solve $Mz_0 = r_0$*;*

(4) $s_0 = z_0$;
(5) ***for*** $k = 1, 2, \ldots$
(6) $\alpha_{k-1} = \frac{z_{k-1}^T r_{k-1}}{s_{k-1}^T L s_{k-1}}$;
(7) $x_k = x_{k-1} + \alpha_{k-1} s_{k-1}$;
(8) $r_k = r_{k-1} - \alpha_{k-1} L s_{k-1}$;
(9) *solve* $M z_k = r_k$;
(10) $\beta_k = \frac{z_k^T r_k}{z_{k-1}^T r_{k-1}}$;
(11) $s_k = z_k + \beta_k s_{k-1}$;
(12) ***end***

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 unsatisfactory result for B, where resulting A and E are balanced much better than B. This is not an issue when balancing involves matrices of the same dimensions as in [29] and [16]. Therefore, we introduce 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 $\frac{n}{m}$. In this case only the definition of the matrix F_1 and the vector c in (3) are changed to

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

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

- $c = [c_i] \in \mathbb{R}^n$ 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 (3) reduces to

$$M = \begin{bmatrix} 3nI_n & 2e_n e_n^T \\ 2e_n e_n^T & 2nI_n \end{bmatrix}. \tag{6}$$

The matrix M is positive definite, and its inverse is equal to

$$M^{-1} = \begin{bmatrix} \frac{1}{3n} I_n + \frac{2}{3n^2} e_n e_n^T & -\frac{1}{n^2} e_n e_n^T \\ -\frac{1}{n^2} e_n e_n^T & \frac{1}{2n} I_n + \frac{1}{n^2} e_n e_n^T \end{bmatrix}. \tag{7}$$

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

$$\min_{l,r,q} \phi(l, r, q), \tag{8}$$

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

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

$$\begin{bmatrix} F_1 & G & K \\ G^T & F_2 & 0 \\ K^T & 0 & F_3 \end{bmatrix} \begin{bmatrix} l \\ r \\ q \end{bmatrix} = \begin{bmatrix} -c \\ -d \\ -f \end{bmatrix}, \tag{9}$$

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

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

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

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

- $f = [f_j] \in \mathbb{R}^m$ 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 (9) reduces to

$$M = \begin{bmatrix} (2n+m)I_n & 2e_ne_n^T & e_ne_m^T \\ 2e_ne_n^T & 2nI_n & 0 \\ e_me_n^T & 0 & nI_m \end{bmatrix}, \tag{10}$$

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

There is an alternative form of the minimization function ϕ in (8), where the part involving the matrix B has stronger weight, as in the previous subsection.

Proposition 1. *(i) The matrix $M \in \mathbb{R}^{(2n+m)\times(2n+m)}$ defined in (10) is symmetric positive semidefinite of rank $2n+m-1$, and its Moore–Penrose generalized inverse is*

$$M^\dagger = \begin{bmatrix} \frac{1}{2n+m} I_n - \frac{3}{2(2n+m)^2} e_n e_n^T & \frac{n-m}{2n(2n+m)^2} e_n e_n^T & \frac{3}{2(2n+m)^2} e_n e_m^T \\ \frac{n-m}{2n(2n+m)^2} e_n e_n^T & \frac{1}{2n} I_n - \frac{3}{2(2n+m)^2} e_n e_n^T & \frac{-5n-m}{2n(2n+m)^2} e_n e_m^T \\ \frac{3}{2(2n+m)^2} e_m e_n^T & \frac{-5n-m}{2n(2n+m)^2} e_m e_n^T & \frac{1}{n} I_m + \frac{-7n-2m}{2n(2n+m)^2} e_m e_m^T \end{bmatrix}. \tag{11}$$

(ii) In case when $L \neq M$, the following holds: $\mathrm{Ker}(L)^\perp \subset \mathrm{Ker}(M)^\perp$.

Proof. (i) 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.

(ii) 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 (ii) of Proposition 1 is important for line **(9)** of Algorithm 1. Since the system $Lx = p$ is consistent $r_k = p - Lx_k \in \mathrm{Im}(L) = \mathrm{Ker}(L)^\perp \subset \mathrm{Ker}(M)^\perp = \mathrm{Im}(M)$. Thus, the system $Mz_k = r_k$ 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

$$M = 2 \begin{bmatrix} nI_n & e_n e_n^T \\ e_n e_n^T & nI_n \end{bmatrix}$$

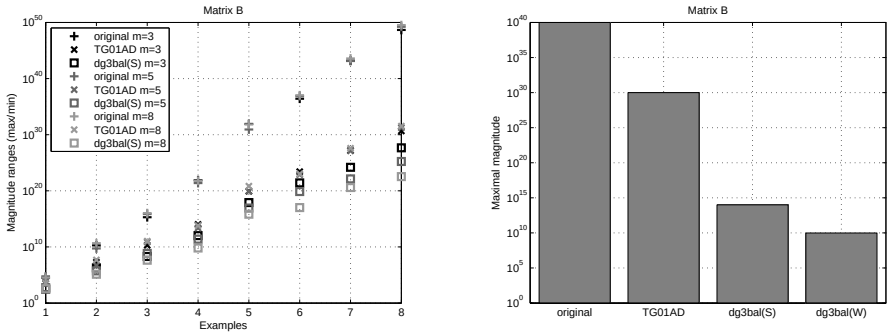
is the same as in `dggbal` (see [29]) and is singular. On the other hand, the system matrix L is as in the original version of our algorithm for $m = 1$ and can be non-singular. In some special cases it can happen that $0 \neq r_k \in \mathrm{Ker}(M)$, producing $z_k = 0$ and the algorithm will prematurely stop since $x_{k+1} = x_k$. Such an example is presented in Section 3.

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 $D_l B D_r$. In this case the balancing procedure reduces to the Ward's balancing algorithm described in [29], 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. 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. We denote by: S – the original variant, W – the weighted variant, R – the variant where B is balanced from both sides.

As a verification of functionality of the `dg3bal` routine, we performed a set of tests with random matrices. The matrices were generated by starting with well scaled random matrices, and then by scaling rows and columns of A and E, as well as rows of B with the same diagonal matrices D_{lr}. The variant S of the balancing algorithm produced all three balanced matrices with a narrow magnitude range, and balanced matrices A and E are very similar to the matrices produced by the LAPACK routine `dggbal`. In case when the columns of B are badly scaled, only the variant R of `dg3bal` algorithm was successful in balancing the matrix B. In case when the rows of B are scaled with a different diagonal matrix D_{lB} whose diagonal elements $D_{lB}(i,i) = D_{lr}(i,i)^3$ have a wider range in magnitude than D_{lr}, the variant W is slightly better for B than the other two variants. On the other hand, the balanced matrices A and E have more scattered elements in this case than the obtained balanced matrices for the variants S and R. Nevertheless, all three variants of our balancing algorithm balanced the matrix B better than the diagonal matrix obtained from `dggbal`. If we define range as proportion between the largest and the smallest element in magnitude, then the logarithm of range of the original matrix B



(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*

Figure 1: *dg3bal vs. TG01AD: reduction of the magnitude range and the maximal magnitude of elements in B*

in this case is reduced on average to 68.86% of its value by `dggbal`, to 57.80% of its value by the variant S, and to 44.27% of its value by the variant W.

3.1. Example 1: `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 1(a) 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`. Let us note here that `TG01AD` cannot balance the columns of B, thus we compare it only to the variant S of `dg3bal`. The variant R would produce a better result than `TG01AD` in case when B has badly scaled columns.

The second test round comprises 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 1(b) 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 system (1). These are the SLICOT routines `TG01HD` and `TG01HX`, and the new staircase reduction algorithm from [4]. 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. A detailed illustration of sensitivity of the staircase reduction is given in Example 5. Figure 1(b) 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. Unfortunately, TG01AD has no option to use the conjugate gradient method without preconditioning, or to change the preconditioner itself. Let the matrices A, E, and B be defined as follows:

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

and $\sqrt{\|A\|_F^2 + \|B\|_F^2} = 1.414214 \cdot 10^{10}$. 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 $\sqrt{\|A_b\|_F^2 + \|B_b\|_F^2} = 1.000001 \cdot 10^5$. The original and the balanced matrices are now used as input to the routine TG01HD. For system (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 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.

3.2. Example 2: 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 [4] was the first step in the algorithm, and σ ranged from $10^{-2}i$ up to $10^{2}i$.

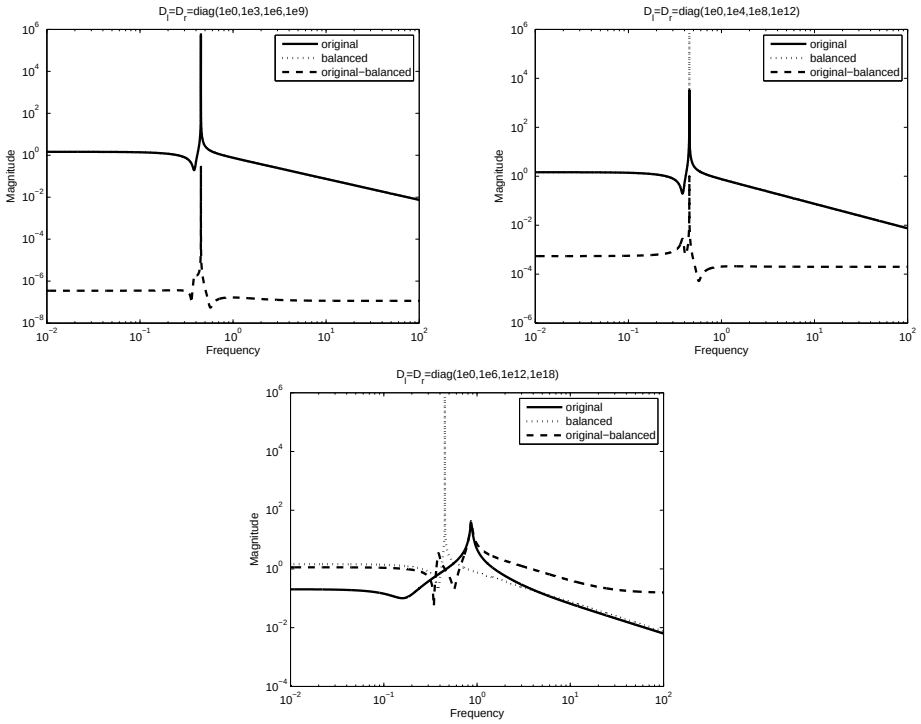


Figure 2: *Magnitude of the computed frequency response matrices, for the original and the balanced system:* $\mathbf{D_{l,0} = D_{r,0}} = diag(\mathbf{10^0, 10^3, 10^6, 10^9})$, $\mathbf{D_{l,0} = D_{r,0}} = diag(\mathbf{10^0, 10^4, 10^8, 10^{12}})$, *and* $\mathbf{D_{l,0} = D_{r,0}} = diag(\mathbf{10^0, 10^6, 10^{12}, 10^{18}})$

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 2.

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 were 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 result 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 to the balanced system. Our algorithm based on the 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.

3.3. Example 3: sensitivity of the staircase reduction

As mentioned earlier, the staircase reduction is a tool used to determine the controllable part of system (1), see, for example, [18] and [28]. 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

$$\begin{aligned} D_1 &= \operatorname{diag}(1, 10^2, 1, 1, 10^4, 1, 1, 10^6, 1, 1, 10^8, 1, 1, 10^9, 1),\\ D_2 &= \operatorname{diag}(1, 10^3, 1, 1, 10^6, 1, 1, 10^9, 1, 1, 10^{12}, 1, 1, 10^{14}, 1),\\ D_3 &= \operatorname{diag}(1, 10^3, 1, 1, 10^6, 1, 1, 10^9, 1, 1, 10^{12}, 1, 1, 10^{16}, 1). \end{aligned}$$

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

Table 1: *The results of `TG01HD` applied to four examples*

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 decreases. 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`. Our balancing routine returns the matrices A_0, B_0 and E_0 for all examples.

3.4. Example 4: 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) via state feedback. For details, see

[11]. Let us define the closed-loop system

$$E\dot{x} = (A - BK)x(t) + Bv(t), \tag{12}$$

where $v(t)$ is an external signal, and $K \in \mathbb{R}^{m\times n}$ is a feedback matrix. For the regular system (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 (12), 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 [17], and the implicit version of the algorithm for ordinary linear time invariant systems with multiple inputs is proposed by Patel and Misra [21]. 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) is defined for $A, E \in \mathbb{R}^{10\times 10}$ and $B \in \mathbb{R}^{10\times 3}$, where

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

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

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

The set of desired finite poles of the closed-loop system is taken to be $\Gamma = \{-1 \pm 2i, -8\pm12i, -3, -7, -15, -25, -30, -100\}$. The pole assignment algorithm produced the feedback matrix

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

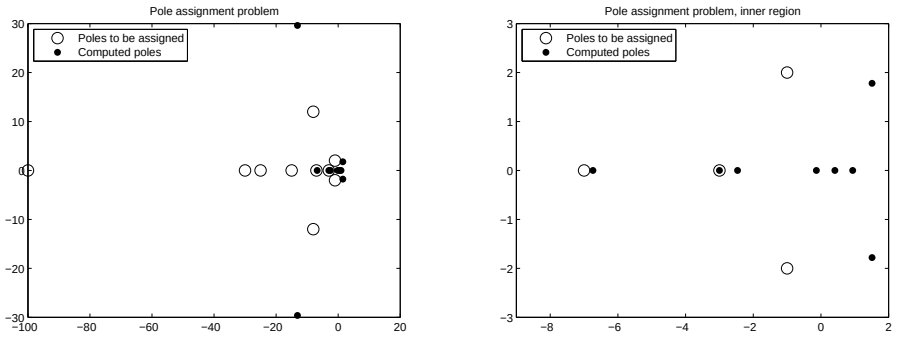


Figure 3: *Desired poles in* Γ *and the computed poles*

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 3. 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

$$K_b = \begin{bmatrix} 1.30{\cdot}10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.49{\cdot}10^{-1} & -6.66{\cdot}10^{0} & -1.00{\cdot}10^{0} & -1.28{\cdot}10^{-1} & -2.04{\cdot}10^{0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.91{\cdot}10^{8} & 2.93{\cdot}10^{8} & 7.45{\cdot}10^{8} & -4.3341{\cdot}10^{7} \end{bmatrix}.$$

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.

3.5. Conclusion of the test results

From the presented test results we can conclude the following. The variant S is most suitable for the control theory problems that include orthogonal transformations of the matrices A and E from both sides, and B only from the left, such as frequency response computing, staircase reduction and the pole assignment problem. If this variant does not balance B in a satisfactory way because the rows of B are worse scaled than the rows of A and E, then usage of the variant W is recommended. In addition, if B has badly scaled columns and the algorithms involving rank revealing are to be applied, such as staircase reduction and pole assignment problem, then the variant R should be used.

At the end, we would like to illustrate sensitivity issues of a badly scaled system demonstrated in this section by an example coming from a real application. The

descriptor system of the form (1) is derived as a power system model, and this example 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 part of the Rommes group in UF Sparse Matrix Collection [23]. 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 $\kappa_2(A) = 3.1321 \cdot 10^{26}$. The routine `dg3bal` balanced all three matrices successfully. 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.

4. Conclusion

In this paper, three versions of an algorithm for balancing three matrices simultaneously are proposed. Balancing is performed via diagonal transformations and the goal is to reduce the range of the 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 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. 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, D. C. Sorensen,

LAPACK Users' Guide, Third ed., SIAM, Philadelphia, 1999.

[2] T. Betcke, *Optimal scaling of generalized and polynomial eigenvalue problems*, SIAM J. Matrix Anal. Appl. **30**(2008), 1320—1338.

[3] Å. Björck, *Least squares methods*, in: *Handbook of Numerical Analysis, Volume I*, (P. G. Ciarlet et al., Eds.), North-Holland, Amsterdam, 1990, 465–652.

[4] N. Bosner, *Efficient algorithm for simultaneous reduction to the m-Hessenberg–triangular–triangular form*, BIT, 2014, DOI: 10.1007/s10543-014-0516-y

[5] N. Bosner, Z. Bujanović, Z. Drmač, *Efficient generalized Hessenberg form and applications*, ACM Trans. Math. Softw. **39**(2013), 19:1–19:19.

[6] P. Concus, G. H. Golub, 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*, Academic Press, New York, 1976, 309–332.

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

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

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

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

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

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

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

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

[15] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second ed., SIAM, Philadelphia, 2002.

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

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

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

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

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

[21] R. V. Patel, P. Misra, *Numerical algorithm for eigenvalue assignment by state feedback*, Proc. IEEE **72**(1984), 1755–1764.

[22] M. J. D. Powell, 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.

[23] *Rommes group*, `http://www.cise.ufl.edu/research/sparse/matrices/Rommes/`

[24] V. Simoncini, *Restarted full orthogonalization method for shifted linear systems*, BIT

43(2003), 459–466.
[25] V. SIMONCINI, 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), 1876–1898.
[26] *SLICOT*, `http://www.slicot.org`
[27] A. VAN DER SLUIS, *Condition numbers and equilibration of matrices*, Numer. Math. **14**(1969), 14–23.
[28] A. VARGA, *Computation of irreducible generalized state-space realizations*, Kybernetika **26**(1990), 89–106.
[29] R. C. WARD, *Balancing the generalized eigenvalue problem*, SIAM J. Sci. Stat. Comput. **2**(1981), 141–152.
[30] D. S. WATKINS, *A case where balancing is harmful*, Electron. Trans. Numer. Anal. **23**(2006), 1–4.