Sherman-Morrison-Woodbury formula for Sylvester and $T$-Sylvester equation with applications

Ivana Kuzmanović, Ninoslav Truhar
Sherman-Morrison-Woodbury formula for Sylvester and T-Sylvester equation with applications

Ivana Kuzmanović * Ninoslav Truhar †

Abstract. In this paper we present the Sherman-Morrison-Woodbury-type formula for the solution of the Sylvester equation of the form

\[(A_0 + U_1V_1)X + X(B_0 + U_2V_2) = E,\]

as well as for the solution of the T-Sylvester equation of the form

\[(A_0 + U_1V_1)X + X^T(B_0 + U_2V_2) = E,\]

where \(U_1, U_2, V_1, V_2\) are low-rank matrices. Although the matrix version of this formula for the Sylvester equation has been used in several different applications (but not for the case of a T-Sylvester equation), we present a novel approach using a proper operator representation. This novel approach allows us to derive a matrix version of the Sherman-Morrison-Woodbury-type formula for the Sylvester equation, as well as for the T-Sylvester equation which seems to be new. We also present algorithms for the efficient calculation of the solution of Sylvester and T-Sylvester equations by using these formulas and illustrate their application in several examples.

Keywords: Sylvester equation, T-Sylvester equation, Sherman-Morrison-Woodbury formula

1 Introduction

In a general case, an \(m \times n\) Sylvester equation takes the form

\[AX + XB = C,\]  

where \(A, B,\) and \(C\) are \(m \times m, n \times n,\) and \(m \times n\) matrices, respectively, and the unknown matrix \(X\) is \(m \times n.\) A Lyapunov equation is a special case with \(m = n, B = A^*,\) and \(C = C^*,\) where the star superscript denotes complex conjugation and transposition.

On the other hand, an \(n \times n\) matrix equation

\[AX + X^TB = C,\]  

*Department of Mathematics, J.J. Strossmayer University of Osijek, Trg Lj. Gaja 6, 31000 Osijek, Croatia, ikuzmano@mathos.hr, corresponding author
†Department of Mathematics, J.J. Strossmayer University of Osijek, Trg Lj. Gaja 6, 31000 Osijek, Croatia, ntruhar@mathos.hr
where $A, B, C$ are $n \times n$, $m \times n$ and $n \times n$ real matrices, respectively, while the unknown matrix $X$ is an $m \times n$ real matrix, is usually called a Sylvester equation for $T$-congruence or in short $T$-Sylvester equation.

Sylvester equation (1) appears frequently in many areas of applied mathematics, both theoretically and practically. We refer the reader to an elegant survey by Bhatia and Rosenthal [1] and references therein for a history of the equation and many interesting and important theoretical results. For example, Sylvester equations play vital roles in matrix eigen-decompositions [2, 3], control theory [4, 5, 3], model reduction [6, 7, 8, 9], numerical solution of matrix differential Riccati equations [10] and algebraic Riccati equation [4], image processing [11], and many more.

$T$-Sylvester matrix equation (2) plays an important role in system theory, such as eigenstructure assignment, observer design, control of systems with input constraints, and fault detection. For a better insight into applications and basic properties of $T$-Sylvester matrix equations we refer the reader to two papers [12] and [13] and references therein.

As it has been known, there is a vast literature about methods for solving Sylvester equation (1) and somewhat smaller but still extensive literature about methods for solving $T$-Sylvester equation (2). However, the aim of this paper is not a discussion about new or improvement of some existing methods for solving Sylvester and $T$-Sylvester equations, but to present a Sherman-Morrison-Woodbury-type formula for Sylvester and $T$-Sylvester equations for the case when matrices $A$ and $B$ from (1) and (2) are updated with small rank updates.

The well known Sherman-Morrison-Woodbury formula reads: if $A_0$ is nonsingular, then $A_0 + UV$ is nonsingular if and only if $I + VA_0^{-1}U$ is nonsingular and then it holds

$$(A_0 + UV)^{-1} = A_0^{-1} - A_0^{-1}U (I + VA_0^{-1}U)^{-1} VA_0^{-1},$$

where $A_0$ is an $m \times m$ matrix, and $U, V^T$ are $m \times r$ real matrices. As we have mentioned above, we will generalize this formula to the case when $A_0$ is the matrix representation of Sylvester and $T$-Sylvester operator, respectively, as well as for the operator equations themselves. We refer the reader to a nice survey by W. W. Hager [14] and references therein for the history of the Sherman-Morrison-Woodbury formula and many interesting results.

The main contribution of this paper are the Sherman-Morrison-Woodbury formulas for the solution of the Sylvester and $T$-Sylvester equation in operator form, as well as for their matrix versions. That is, we consider the Sylvester equation of the form

$$(A_0 + U_1V_1)X + X(B_0 + U_2V_2) = E,$$

and the $T$-Sylvester equation of the form

$$(A_0 + U_1V_1)X + X^T(B_0 + U_2V_2) = E,$$

where $U_1, U_2, V_1, V_2$ are low-rank matrices of proper dimensions. Although a matrix version of the Sherman-Morrison-Woodbury formula for the Sylvester equation has been used in several different applications (but not for the case of the $T$-Sylvester equation), we present a novel approach using a proper operator representation.
The rest of the paper is organized as follows. In Section 2, we present the Sherman-Morrison-Woodbury formulas for the Sylvester equation. The corresponding algorithm for solving the Sylvester equation using the Sherman-Morrison-Woodbury formula is given in Section 2.1. Some applications of the Sherman-Morrison-Woodbury formula are presented in Section 2.2. In Section 3, we present Sherman-Morrison-Woodbury formula for the \( T \)-Sylvester equation. The corresponding algorithm with a numerical example is given in Section 3.1.

Throughout this paper we will use the following notations. The symbol \( I_n \) denotes an \( n \times n \) identity matrix and the subscript will be omitted when the dimension is clear from the context. \( A \otimes B \) denotes the Kronecker product of \( A \) and \( B \), while vec \( (A) \) denotes the vectorization of the matrix \( A \) formed by stacking the columns of \( A \) into a single column vector. If \( x \) is a vector such that \( x = [x_1, \ldots, x_n]^T \), then \( \text{x}(i : i + k) = [x_i, x_{i+1}, \ldots, x_{i+k}]^T \).

We will also use the standard Matlab notation for matrix columns and rows, that is \( A(:, i) \) and \( A(i,:) \) denote the \( i \)-th column and the \( i \)-th row of the matrix \( A \), respectively.

## 2 Sherman-Morrison-Woodbury formula for Sylvester equation

We consider the following Sylvester equation

\[
(A_0 + U_1 V_1)X + X(B_0 + U_2 V_2) = E, \tag{4}
\]

where \( A_0 \) is an \( m \times m \) matrix, \( U_1, V_1^T \) are \( m \times r_1 \) matrices, \( B_0 \) is an \( n \times n \) matrix, \( U_2, V_2^T \) are \( n \times r_2 \) real matrices and the unknown matrix \( X \) as well as the right-hand side matrix \( E \) are \( m \times n \) real matrices.

Our main result will be an operator version of the Sherman-Morrison-Woodbury formula for Sylvester equation (4). For that purpose let

\[
\mathcal{L}_0(X) = A_0 X + X B_0, \tag{5}
\]
\[
\mathcal{L}_{UV}(X) = U_1 V_1 X + X U_2 V_2 \tag{6}
\]

be standard Sylvester operators. It can be shown straightforwardly that Sylvester equation (4) is equivalent to

\[
(\mathcal{L}_0 + \mathcal{L}_{UV})(X) = E. \tag{7}
\]

Thus, if \( \mathcal{L}_0(X) \) from (5) has a known inverse or if it can be inverted easily and if \( \mathcal{L}_{UV} \) from (6) has a small rank, then it is natural to look for the Sherman-Morrison-Woodbury-type formula for Sylvester operator equation (7).

Before we present the Sherman-Morrison-Woodbury formula, we will introduce some auxiliary notation.

Let \( \mathcal{U}: \mathbb{R}^{r_1 \times n} \times \mathbb{R}^{m \times r_2} \rightarrow \mathbb{R}^{m \times n} \) be a bilinear operator defined by

\[
\mathcal{U}(X_1, X_2) = U_1 X_1 + X_2 V_2, \tag{8}
\]
and let \( V: \mathbb{R}^{m \times n} \to \mathbb{R}^{r_1 \times n} \times \mathbb{R}^{m \times r_2} \) be a linear operator defined by
\[
V(Y) = (V_1 Y, Y U_2),
\]
where \( U_1, U_2, V_1 \) and \( V_2 \) are matrices defined in (4). The key observation here is that Sylvester operator \( L_{UV}(X) \) from (6) can be decomposed as
\[
L_{UV}(X) = (UV)(X).
\]
(10)

Now, using (10), the Sylvester equation (in operator form) can be written as
\[
(L_0 + UV)(X) = E.
\]

If \( L_0 \) and \( L_0 + UV \) are regular operators, one can apply the Sherman-Morrison-Woodbury formula to the previous equation. After that, we obtain the following so-called Sherman-Morrison-Woodbury formula for the Sylvester equation:\(^1\)
\[
X = (L_0^{-1} - L_0^{-1} U (I + VL_0^{-1} U)^{-1} VL_0^{-1})(E),
\]
(11)

where \( I \) is an identity operator.

Since the main aim of this paper is construction of the Sherman-Morrison-Woodbury formula for the Sylvester equation (and the \( T \)-Sylvester equation), we will skip a discussion about possible instability problems and regularity of \( L_0 \) and \( L_0 + UV \). Thus, in the sequel we will assume that the regularity condition \( (L_0 \) and \( L_0 + UV \) are regular) is fulfilled and that neither \( L_0 \) nor \( I + VL_0^{-1} U \) are ill-conditioned.

Correctness of the Sherman-Morrison-Woodbury formula (11) for the updated Sylvester operator can be shown using a matrix representation of operators (5), (8) and (9) in the standard canonical basis. Thus, in the following theorem we present a matrix representation of equality (11).

**Theorem 1** Let \( L_0, U \) and \( V \) be matrices defined by
\[
L_0 = I_n \otimes A_0 + B_0^T \otimes I_m \in \mathbb{R}^{mn \times mn},
\]
(12)
\[
U = [I_n \otimes U_1 V_2^T \otimes I_m] \in \mathbb{R}^{mn \times (r_1 n + r_2 m)},
\]
(13)
\[
V = \begin{bmatrix} I_n \otimes V_1 \\ U_2^T \otimes I_m \end{bmatrix} \in \mathbb{R}^{(r_1 n + r_2 m) \times mn},
\]
(14)
where \( A_0, B_0 \) and \( U_1, U_2, V_1, V_2 \) are matrices from (4). If \( L_0 \) and \( L_0 + UV \) are regular matrices, then for the solution of equation (4) it holds
\[
\text{vec}(X) = (L_0^{-1} - L_0^{-1} U (I + VL_0^{-1} U)^{-1} VL_0^{-1}) \text{vec}(E).
\]
(15)

---

\(^1\)According to authors’ knowledge, no one has formulated the Sherman-Morrison-Woodbury formula for the Sylvester equation so far. Authors would like to thank Volker Mehrmann for having pointed out this interesting paradox to them during the 17th ILAS Conference in Braunschweig.
Proof. Using the standard Kronecker product and the vectorization operator together with the property \( \text{vec}(ABC) = (C^T \otimes A) \text{vec}(B) \), equation (4) can be rewritten as an \( mn \times mn \) matrix equation

\[
(I_n \otimes (A_0 + U_1 V_1) + (B_0 + U_2 V_2)^T \otimes I_m) \text{vec}(X) = \text{vec}(E).
\]  \hspace{1cm} (16)

Now it is easy to see that equation (16) can be presented in the following form

\[
(L_0 + UV) \text{vec}(X) = \text{vec}(E),
\]

where \( L_0, U \) and \( V \) are given by (12), (13) and (14), respectively.

After applying Sherman-Morrison-Woodbury formula (3) to the matrix \((L_0 + UV)\), we obtain

\[
\text{vec}(X) = (L_0 + UV)^{-1} \text{vec}(E) = (L_0^{-1} - L_0^{-1}U(I + VL_0^{-1}U)^{-1}VL_0^{-1}) \text{vec}(E).
\]

}\hspace{1cm} \square

2.1 Algorithm for solving Sylvester equation using Sherman-Morrison-Woodbury formula

In this section, we present our first algorithm which contains basic procedures for calculating the solution of the Sylvester equation using formula (15).

\begin{algorithm}
\caption{Basic procedure for solving equation (4) using Sherman-Morrison-Woodbury formula}
\begin{algorithmic}
\Input \( A_0 \in \mathbb{R}^{m \times m}, U_1, V_1^T \in \mathbb{R}^{m \times r_1}, B_0 \in \mathbb{R}^{n \times n}, U_2, V_2^T \in \mathbb{R}^{n \times r_2}, E \in \mathbb{R}^{m \times n} \)
\Output \( X \in \mathbb{R}^{m \times n} \)
\State Calculate \( x_1 = L_0^{-1} \text{vec}(E) \) if \( L_0^{-1} \) is given or solve \( L_0 x_1 = \text{vec}(E) \)
\State Calculate \( x_2 = V x_1 \)
\State Solve \((I + VL_0^{-1}U)x_3 = x_2 \)
\State Calculate \( x_4 = U x_3 \)
\State Calculate \( x_5 = L_0^{-1} x_4 \) or solve \( L_0 x_5 = x_4 \)
\State \( \text{vec}(X) = x_1 - x_5 \)
\end{algorithmic}
\end{algorithm}

Without using any structure, a calculation of the solution \( X \) using Algorithm 1 would require \( O(m^3 n^3) \) operations, but fortunately matrices in equation (15) have a special structure which allows a more efficient calculation of \( X \).

In what follows we consider steps 1–5 of Algorithm 1 more carefully. This will allow us to construct the procedures for an efficient calculation of matrix vector products \( UX \) and \( Vx \), and efficient solving of linear systems \( L_0 x = y \) and \((I + VL_0^{-1}U)x = y\).
If inverse $L_0^{-1}$ is not given, then steps 1 and 5 can be performed as follows.

Calculation of $x = L_0^{-1}y$, that is, solving the linear system $L_0x = y$, where

$$L_0 = I_n \otimes A_0 + B_0^T \otimes I_m$$

is equivalent to solving the Sylvester equation

$$A_0 \hat{X} + \hat{X}B_0 = \hat{Y}, \quad (17)$$

where $\text{vec}(\hat{X}) = x$ and $\text{vec}(\hat{Y}) = y$. There is a number of methods for solving (17), but one of the most popular direct solvers is a well-known Bartels-Stewart method [15, 16].

If we do not assume any special structure on $A_0$ and $B_0$, then any method similar to the Bartels-Stewart method (any method based on the Schur decomposition) will have complexity $O(n^3 + m^3)$. On the other hand, if $A_0$ and $B_0$ have some useful structure, a number of operations can be significantly smaller. For example, if $A_0$ and $B_0$ are block diagonal matrices with small blocks on the diagonal, complexity is $O(m^2 + n^2)$ operations. Particularly, if $A_0$ and $B_0$ are block matrices with $2 \times 2$ blocks on the diagonal, then equation (17) can be solved in $8mn$ operations (see [17]). For the sake of simplicity, from now on we will assume that both $A_0$ and $B_0$ have a (block)-diagonal structure.

Multiplication of $U$ and $V$ by a vector $x \in \mathbb{R}^{r_1n + r_2m}$ and $y \in \mathbb{R}^{mn}$ in steps 4 and 2, respectively, can be done as follows:

$$Ux = \begin{bmatrix} I_n \otimes U_1 & V_2^T \otimes I_m \end{bmatrix} \begin{bmatrix} \text{vec}(\hat{X}_1) \\ \text{vec}(\hat{X}_2) \end{bmatrix} = \text{vec}(U_1\hat{X}_1 + \hat{X}_2V_2),$$

$$Vy = \begin{bmatrix} I_n \otimes V_1 \\ U_2^T \otimes I_m \end{bmatrix} \text{vec}(\hat{Y}) = \begin{bmatrix} (I_n \otimes V_1) \text{vec}(\hat{Y}) \\ (U_2^T \otimes I_m) \text{vec}(\hat{Y}) \end{bmatrix} = \begin{bmatrix} \text{vec}(V_1\hat{Y}) \\ \text{vec}(\hat{Y}U_2) \end{bmatrix},$$

where $\hat{X}_1 \in \mathbb{R}^{r_1 \times n}$, $\text{vec}(\hat{X}_1) = x(1 : r_1n)$, $\hat{X}_2 \in \mathbb{R}^{m \times r_2}$, $\text{vec}(\hat{X}_2) = x(r_1n + 1 : r_1n + r_2m)$ and $\text{vec}(\hat{Y}) = y$.

It is easy to see that the number of operations needed for each of the products $Ux$ and $Vy$ is $2mn(r_1 + r_2)$.

For step 3 we have to derive matrix $\Delta := VL_0^{-1}U$. Since matrices $U, V$ have $mn(r_1n + r_2m)$ and $L_0$ has $m^2n^2$ entries, for larger $m$ and $n$, we cannot multiply them directly so we need a procedure for construction of $\Delta$ without explicit construction of $U, V, L_0^{-1}$.

Every column of $L_0^{-1}U$, that is a column vector $\text{vec}(\hat{X}_i) := L_0^{-1}U(:, i)$ can be obtained using (17) as a solution of equation

$$A_0\hat{X}_i + \hat{X}_iB_0 = \hat{U}_i,$$

where $\text{vec}(\hat{U}_i) = U(:, i)$, $i = 1, \ldots, r_1n + r_2m$. Further multiplication by matrix $V$ can be done as follows

$$\Delta = V \begin{bmatrix} \text{vec}(\hat{X}_1) & \text{vec}(\hat{X}_2) & \cdots & \text{vec}(\hat{X}_{r_1n+r_2m}) \end{bmatrix}$$

$$= \begin{bmatrix} \text{vec}(V_1\hat{X}_1) & \text{vec}(V_1\hat{X}_2) & \cdots & \text{vec}(V_1\hat{X}_{r_1n+r_2m}) \\ \text{vec}(\hat{X}_1U_2) & \text{vec}(\hat{X}_2U_2) & \cdots & \text{vec}(\hat{X}_{r_1n+r_2m}U_2) \end{bmatrix}.$$
Now, since the special block structure of $A_0$ and $B_0$ has been assumed, each $\hat{X}_i$ can be calculated in $8mn$ operations, so the number of operations needed for construction of $\Delta$ is $2mn(r_1n + r_2m)(4 + r_1 + r_2)$.

The last part in step 3 consists of solving the linear system

$$\begin{equation}
(I + \Delta)x_3 = x_2,
\end{equation}$$

where $x_2$ is obtained in step 2 of Algorithm 1.

A method of choice for solving system (18) efficiently depends on the structure of the problem. In the following section, few methods will be proposed in concrete applications. The main part of the number of floating point operations needed for solving the Sylvester equation using the Sherman-Morrison-Woodbury formula mainly belongs to the number of operations needed for solving (18) and it will be more precisely investigated later for concrete applications.

2.2 Applications

This section contains two illustrations of possible applications of the Sherman-Morrison-Woodbury formula for the solution of the Sylvester and Lyapunov equation. As the first illustration, we will use the algorithm for solving the algebraic Riccati equation presented in [18] which typically arises from transport theory.

The second illustration considers the dampers’ viscosity optimization in mechanical systems and it has been widely investigated in the last decade (for example see [17, 19, 20, 21]).

2.2.1 Newton method for nonsymmetric algebraic Riccati equation

As it has been mentioned above we consider a fast Newton’s method for a nonsymmetric algebraic Riccati equation presented in [18] as the first possible application. It has been shown that solving one specific nonsymmetric algebraic Riccati equation requires solving many low-rank updates of the Sylvester equation.

More precisely, we consider the problem of solving the algebraic Riccati equation

$$\begin{equation}
XCX - XE - AX + B = 0,
\end{equation}$$

where $A, B, C, E$ are real matrices of order $n$ of the form

$$\begin{equation}
A = A_0 - \tilde{e}q^T, \quad B = \tilde{e}e^T, \quad C = \tilde{q}q^T, \quad E = B_0 - \tilde{q}e^T,
\end{equation}$$

where $A_0$ and $B_0$ are diagonal matrices, and $q, \tilde{q}, e, \tilde{e}$ are nonnegative vectors. A minimal nonnegative solution of equation (19) can be obtained by the Newton method in which every step consists of solving the Sylvester equation of the form

$$\begin{equation}
(X^{(k+1)} - X^{(k)}) (E - CX^{(k)}) + (A - X^{(k)}C) (X^{(k+1)} - X^{(k)}) = R(X^{(k)}),
\end{equation}$$

Authors would like to thank Federico Poloni for point them out on this interesting problem during the Summer School on Numerical Linear Algebra for Dynamical and High-Dimensional Problems, Trogir, 2011.
where \( R(X) = XCX -XE - AX + B \). Equation (21) with matrices from (20) can be written as

\[
(L_0 - U^{(k)} V) \text{vec}(X^{(k+1)} - X^{(k)}) = \text{vec}(R(X^{(k)})),
\]

where

\[
L_0 = I_n \otimes A_0 + B_0^T \otimes I_n,
\]

\[
U^{(k)} = \begin{bmatrix}
I_n \otimes (\tilde{e} + X^{(k)} \tilde{q}) & (e^T + q^T X^{(k)})^T \otimes I_n
\end{bmatrix},
\]

\[
V = \begin{bmatrix}
I_n \otimes q^T \\
\tilde{q}^T \otimes I_n
\end{bmatrix},
\]

and then the solution is given by

\[
\text{vec}(X^{(k+1)} - X^{(k)}) = (L_0^{-1} + L_0^{-1} U^{(k)} (I_{2n} - V L_0^{-1} U^{(k)})^{-1} V L_0^{-1}) \text{vec}(R(X^{(k)})).
\] (22)

For solving the system with \((I_{2n} - V L_0^{-1} U^{(k)})\), a fast Gaussian elimination method of complexity \(O(n^2)\) is proposed in [18].

Usually, every step in Newton’s method has complexity \(O(n^3)\). Using structure of matrices in (20) and equation (22), every step of Newton’s method can be done in \(O(n^2)\) operations. For more details see [18].

**Example 1** This example shows performance of Newton method for solving (19) implemented in two different ways. The first approach, denoted by \(\text{lyap}\), uses the standard Bartels-Stewart method for solving equation (21) and the second approach, denoted by \(\text{SMW}\), uses formula (22) for solving (21). We will compare accuracy as well as complexity of the above approaches. Let us consider the following configuration:

\[
A_0 = \text{diag}(\text{rand}(n, 1)), \quad B_0 = \text{diag}(\text{rand}(n, 1)),
\]

\[
q = \text{rand}(n, 1), \quad \tilde{q} = \text{rand}(n, 1), \quad e = \text{ones}(n, 1), \quad \tilde{e} = \text{rand}(n, 1).
\]

Norms of residuals \(\|XCX -XE - AX + B\|\) for different values of \(n\) for both approaches are given in Table 1. The residuals in both approaches are calculated after the same number of steps in Newton method, denoted by \(k\).

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Residual in \text{lyap}</th>
<th>Residual in \text{SMW}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n = 25, k = 10)</td>
<td>(4.2 \cdot 10^{-14})</td>
<td>(1.8 \cdot 10^{-14})</td>
</tr>
<tr>
<td>(n = 50, k = 10)</td>
<td>(1.73 \cdot 10^{-14})</td>
<td>(8.7 \cdot 10^{-15})</td>
</tr>
<tr>
<td>(n = 100, k = 10)</td>
<td>(4.8 \cdot 10^{-13})</td>
<td>(1.04 \cdot 10^{-13})</td>
</tr>
</tbody>
</table>

Table 1: Results of comparison of approaches \text{lyap} and \text{SMW}

The growth in the number of flops, depending on the problem dimension, for one step of Newton’s method with \text{lyap} and \text{SMW} approaches is illustrated in Figure 1.
2.2.2 Optimization of dampers’ viscosities

The second example, where many low-rank updates of the Lyapunov equation have to be solved, arises in the process of optimization of dampers’ viscosities in a linear vibrational system.

We will consider an $n$-mass oscillator given in Figure 2. It can be described by the system of differential equations

\[ M\ddot{x} + D\dot{x} + Kx = 0, \]

\[ x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0 \]

where mass, damping and stiffness matrices ($M$, $D$ and $K$) are real, symmetric and positive.
definite matrices of order $n$ given by

$$M = \text{diag}(m_1, m_2, \ldots, m_n),$$

$$K = \begin{pmatrix}
 k_1 + k_2 & -k_2 & \cdots & \cdots & -k_n \\
 -k_2 & k_2 + k_3 & -k_3 & \cdots & \cdots \\
 \cdots & \cdots & \cdots & \cdots & \cdots \\
 -k_{n-1} & k_{n-1} + k_n & -k_n & \cdots & \cdots \\
 -k_n & -k_n & k_n + k_{n+1} & \cdots & \cdots
\end{pmatrix},$$

where $m_i > 0$, $i = 1, \ldots, n$ are masses and $k_i > 0$, $i = 1, \ldots, k + 1$ are spring constants or stiffnesses. The damping matrix is of the form $D = C_u + C$, where $C_u$ is internal damping and $C$ is external damping and it depends on damper’s positions and viscosities. For example, the oscillator in Figure 2 has two dampers at positions one and three with viscosities $v_1$ and $v_2$. This means that $D = C_u + C = C_u + v_1 e_1 e_1^T + v_2 e_3 e_3^T$, where $e_i$ is the $i$-th canonical basis vector.

Linearization of the system of differential equations (23) can be obtained using simultaneous diagonalization of the matrix pair $M$ and $K$, i.e. there exists a regular matrix $\Phi$ such that $\Phi^T M \Phi = \Omega$ and $\Phi^T K \Phi = I$, where $\Omega$ is a diagonal matrix. Then, after substitution $x = \Phi x_\Phi$, equation (23) is equivalent to

$$\ddot{x}_\Phi + \Phi^T D \Phi \dot{x}_\Phi + \Omega^2 x_\Phi = 0.$$

If we define $y_1 = \Omega x_\Phi$ and $y_2 = \dot{x}_\Phi$, then

$$\dot{y}_1 = \Omega y_2,$$

$$\dot{y}_2 = \ddot{x}_\Phi = -\Phi^T D \Phi \dot{x}_\Phi - \Omega \Omega x_\Phi = -\Phi^T D \Phi y_2 - \Omega y_1.$$

Now we can write the differential equation in the so-called phase space as

$$\dot{y} = Ay,$$

where

$$A = \begin{bmatrix}
 0 & \Omega \\
 -\Omega & -\Phi^T D \Phi
\end{bmatrix}, \quad y = \begin{bmatrix}
y_1 \\
y_2
\end{bmatrix}.$$ (24)

The solution of equation (2.2.2) is given by

$$y = e^{At} y_0,$$ where $y_0$ contains the initial data.

The problem of damping optimization reads: for given matrices $M$ and $K$ find the best matrix $D$ which insures optimal evanescence of each component of $y$. This optimization problem can be studied with respect to different optimization criteria. Here we will use a criterion based on minimization of the average total energy of the system over the set of initial conditions. It can be shown (see e.g. [19]) that this criterion is equivalent to trace minimization of the solution $X$ of the corresponding Lyapunov equation

$$AX + XA^T = -Z,$$
where $A$ is given in (24) and the matrix $Z$ determines which part of undamped eigenfrequencies has to be damped. The case $Z = GG^T$ where

$$G = \begin{bmatrix} I_s & 0 \\ 0 & 0 \\ 0 & I_s \\ 0 & 0 \end{bmatrix}$$

corresponds to the case when just first $s, (s < n)$ eigenfrequencies of the undamped system have to be damped. Particularly, if $G = I$, then all undamped eigenfrequencies have to be damped (for more details about construction of the matrix $Z$ see [19]).

The problem of calculating optimal viscosities of $r$ dampers with the same viscosity placed on positions $i_1, \ldots, i_r$ leads to the optimization process in which one has to solve many $m \times m$ parameter dependent Lyapunov equations of the form

$$(A_0 - vCC^T)X(v) + X(v)(A_0 - vCC^T)^T = -Z.$$  

Matrices in the previous equation are strongly structured. $A_0$ is of the form

$$A_0 = \begin{bmatrix} 0 & \Omega \\ -\Omega & -\alpha\Omega \end{bmatrix}$$

($C_u$ is usually taken to be between $2-10\%$ of critical damping (see [22]), i.e. $C_u = \alpha\Phi\Omega\Phi^T$, $0.02 \leq \alpha \leq 0.1$) and

$$C = \begin{bmatrix} 0 \\ C_0 \end{bmatrix}, \text{ where } C_0 = \Phi^T[e_{i_1}, \ldots, e_{i_r}]$$

and $e_{i_k}$ is the $i_k$-th canonical base vector. Optimization is done with respect to parameter $v \in \mathbb{R}, v > 0$ and it can be done e.g. with Newton’s method.

Application of the Sherman-Morrison-Woodbury formula to the problem described above is observed in detail in [17, 20, 21]. In [20], Algorithm 1 is used where a system with $(I + v\Delta)$ is solved by reducing $\Delta$ to Hessenberg form

$$\Delta = U_\Delta H_\Delta U_\Delta^T.$$  

Then system (18) becomes a $2rm \times 2rm$ Hessenberg system

$$(I + vH_\Delta)x = U_\Delta^T y U_\Delta,$$  

and it can be solved in $O(r^2m^2)$ operations. Matrices $U_\Delta$ and $H_\Delta$ can be obtained in $\frac{14}{3}(rm)^3$ operations.

Now, the total number of operations for solving equation (2.2.2) for the first time is $37.33r^3m^3 + O(r^2m^2)$. Each further solving of (4) with a different value of $v$ takes only $O(r^2m^2)$ additional operations. In [20], it is also shown that $X'(v)$ and $X''(v)$ can also be calculated in additional $O(r^2m^2)$ operations. This means that the whole optimization process will have complexity $37.33r^3m^3 + O(r^2m^2)$, $m = 2n$. 

11
System (18) can also be solved by some of Krylov subspace methods, for example FOM - Full Orthogonalization Method (see [23]). In this case, \( \Delta \) does not have to be formed, it is enough to have a procedure for calculation of a product of \( \Delta \) by a vector. For more details see [17, 23]. If we solve equation system (18) using FOM, then the number of operations for the whole process is \( 8(rk + r + k + 2)m^2 + O(rmk) \), where \( m = 2n \) and \( k \) is a dimension of Krylov subspace in FOM (see [17]).

On the other hand, Newton’s type routine with the Bartels-Stewart solver requires solving of three Lyapunov equations per iteration to obtain \( X(v), X'(v), X''(v) \) so every iteration in the Newton’s optimization process costs \( (26+3\cdot6)m^3+O(m^2) = 44m^3+O(m^2) \) operations and it usually takes 10–20 iterations until convergence.

In the following example we will compare the number of operations as well as accuracy of the standard algorithm for optimization of viscosities with both versions of our algorithm - exact and approximate.

**Example 2** We will compare a standard Newton’s type routine for optimization of dampers’ viscosities based on: the Bartels-Stewart solver (denoted by BS) and the novel Sherman-Morrison-Woodbury formula. In the algorithm based on the Sherman-Morrison-Woodbury formula we will use two different linear solvers for system (18). The first one is an exact solver based on Hessenberg decomposition as in (25), denoted by SMW-exact, and the second one is an approximate solver based on the Krylov subspace method FOM, denoted by SMW-approx. A comparison will be done for the following configuration

\[
M = I_n, \quad k_1 = k_2 = \cdots = k_{n+1} = 1, \quad Z = I_m
\]

\[
r = 2, \quad C = v(e_{n/10}e_{n/10}^T + e_{n/5}e_{n/5}^T)
\]

for several dimensions: \( n = 200 (m = 400) \), \( n = 500 (m = 1000) \), \( n = 1000 (m = 2000) \) and \( n = 1500 (m = 3000) \). Obtained results are shown in Table 2, where \( v_{BS}^* \) and \( Tr_{BS}^* \), \( v_{E}^* \) and \( Tr_{E}^* \), \( v_{A}^* \) and \( Tr_{A}^* \) are optimal viscosity and optimal trace obtained by BS, SMW-exact and SMW-approx algorithms, respectively. \( RErr_E = \frac{|Tr_{BS}^* - Tr_{E}^*|}{Tr_{BS}^*} \) is a relative error in optimal trace between BS and SMW-exact, while \( RErr_A = \frac{|Tr_{BS}^* - Tr_{A}^*|}{Tr_{BS}^*} \) is a relative error in optimal trace between BS and SMW-approx.

It can be seen that in the SMW-approx approach for smaller \( n = O(100) \) dimension \( k \) of the Krylov subspace in FOM for this kind of problems should not be less than 10% of problem dimension. On the other hand, for \( n = O(1000) \), \( k \) can be chosen to be 1% – 3% of problem dimension. For greater dimensions algorithm SMW-exact fails because of a lack of computer memory.

In Figure 3, growth in the number of flops has been illustrated, depending on the problem dimension.
In this section we consider a matrix equation of the form

$$A_0 + U_1 V_1^T X + X^T (B_0 + U_2 V_2) = E,$$  \hspace{1cm} (26)

where $A_0, B_0, E$ are $n \times n$ matrices, $U_1, V_1^T$ are $n \times r_1$ matrices, $U_2, V_2^T$ are $n \times r_2$ real matrices and $X$ is an $n \times n$ unknown matrix. Equation (26) is usually called a Sylvester equation for $T$-congruence (see [13]) or shortened a $T$-Sylvester equation.

In [13], an algorithm for solving a $T$-Sylvester equation $AX + X^T B = C$ based on a generalized Schur decomposition of the pair $(A, B^T)$ of complexity $76n^3$ is given. If $U_1, U_2, V_1, V_2$ are small-rank matrices and if $A_0$ and $B_0$ are well-structured, equation (26) can be solved more efficiently, especially if it has to be solved many times with different matrices $U_1, U_2, V_1, V_2$.

Table 2: Results of comparison of algorithms BS, SMW-exact and SMW-approx

<table>
<thead>
<tr>
<th>Dimension</th>
<th>BS and SMW-exact</th>
<th>SMW-approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 200$ ($m = 400$)</td>
<td>$v^<em>_B = 1.3633$ $T v^</em>_B = 9711.4087$</td>
<td>$k = n/5$, $v^<em>_A = 1.3633$ $T v^</em>_A = 9711.4087$ $R\text{Err}_A = 2.8 \cdot 10^{-11}$</td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.3633$ $T v^</em>_E = 9711.4087$ $R\text{Err}_E = 1.1 \cdot 10^{-14}$</td>
<td></td>
</tr>
<tr>
<td>$n = 500$ ($m = 1000$)</td>
<td>$v^<em>_B = 1.3893$ $T v^</em>_B = 29909.3555$</td>
<td>$k = n/5$, $v^<em>_A = 1.3893$ $T v^</em>_A = 29909.3555$ $R\text{Err}_A = 4.4 \cdot 10^{-14}$</td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.3893$ $T v^</em>_E = 29909.3555$ $R\text{Err}_E = 4.5 \cdot 10^{-14}$</td>
<td></td>
</tr>
<tr>
<td>$n = 1000$ ($m = 2000$)</td>
<td>$v^<em>_B = 1.3996$ $T v^</em>_B = 68495.2881$</td>
<td>$k = n/50$, $v^<em>_A = 1.3999$ $T v^</em>_A = 68495.1769$ $R\text{Err}_A = 1.7 \cdot 10^{-6}$</td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.3996$ $T v^</em>_E = 68495.2881$ $R\text{Err}_E = 3.6 \cdot 10^{-14}$</td>
<td></td>
</tr>
<tr>
<td>$n = 1500$ ($m = 3000$)</td>
<td>$v^<em>_B = 1.4033$ $T v^</em>_B = 110409.5415$ SMW-exact: out of memory</td>
<td>$k = n/100$, $v^<em>_A = 1.4045$ $T v^</em>_A = 110407.9622$ $R\text{Err}_A = 1.4 \cdot 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.4033$ $T v^</em>_E = 110409.5428$ $R\text{Err}_E = 1.1 \cdot 10^{-8}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.4045$ $T v^</em>_E = 110407.9622$ $R\text{Err}_E = 1.4 \cdot 10^{-5}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$v^<em>_E = 1.4108$ $T v^</em>_E = 110397.4021$ $R\text{Err}_E = 1.1 \cdot 10^{-4}$</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3: Growth in the number of flops for algorithms BS, SMW-exact and SMW-approx

Similarly to the case of the Sylvester equation, the Sherman-Morrison-Woodbury-type formula can also be formulated for the $T$-Sylvester equation. For that purpose we will define $T$-Sylvester operators

$$\hat{L}_0(X) = A_0X + X^TB_0, \quad (27)$$

$$\hat{L}_{UV}(X) = U_1V_1X + X^T U_2V_2. \quad (28)$$

Using a similar approach as in Section 2, we will define operators:

$$\hat{U} : \mathbb{R}^{r_1 \times n} \times \mathbb{R}^{r_2 \times n} \to \mathbb{R}^{n \times n}, \quad \hat{U}(X_1, X_2) = U_1X_1 + X_2^TV_2, \quad (29)$$

and

$$\hat{V} : \mathbb{R}^{n \times n} \to \mathbb{R}^{r_1 \times n} \times \mathbb{R}^{r_2 \times n}, \quad \hat{V}(Y) = (V_1Y, U_2^TY), \quad (30)$$

where $U_1, U_2, V_1$ and $V_2$ are matrices defined in (26). Again, the key observation is that $T$-Sylvester operator $\hat{L}_{UV}(X)$ from (28) can be decomposed as

$$\hat{L}_{UV}(X) = \left(\hat{U}\hat{V}\right)(X). \quad (31)$$

Now, using (31), the $T$-Sylvester equation (in operator form) can be written as

$$\left(\hat{L}_0 + \hat{U}\hat{V}\right)(X) = E.$$

If $\hat{L}_0$ and $\hat{L}_0 + \hat{U}\hat{V}$ are regular operators, one can apply the Sherman-Morrison-Woodbury formula to obtain

$$X = (\hat{L}_0^{-1} - \hat{L}_0^{-1}\hat{U}(I + \hat{V}\hat{L}_0^{-1}\hat{U})^{-1}\hat{V}\hat{L}_0^{-1})(E), \quad (32)$$
where $I$ is an identity operator.

Similarly to the case of the Sylvester equation, we will not discuss the conditions for regularity of $\hat{L}_0$ and $I + \hat{V}\hat{L}_0^{-1}\hat{U}$, since this problem is out of the scope of this paper. Instead, we will assume that the regularity condition ($\hat{L}_0$ and $\hat{L}_0 + \hat{U}\hat{V}$ are regular) is fulfilled. Here one can also verify the correctness of the Sherman-Morrison-Woodbury formula for an updated $T$-Sylvester operator (32) using matrix representation of operators (27), (29) and (30) in the standard canonical basis which will be done in the following theorem.

Note that $\text{vec}(X^T) = \Pi \text{vec}(X)$, where $\Pi$ is defined as

$$
\Pi = \sum_{1 \leq i,j \leq n} e_i e_j^T \otimes e_j e_i^T,
$$

and where $e_i$, $i = 1, \ldots, n$ is the $i$-th canonical vector. This observation allows us to derive the matrix version of the Sherman-Morrison-Woodbury formula (32).

**Theorem 2** Let $\hat{L}_0$, $\hat{U}$ and $\hat{V}$ be matrices defined by

\[
\begin{align*}
\hat{L}_0 &= I_n \otimes A_0 + (B_0^T \otimes I_n)\Pi \in \mathbb{R}^{n^2 \times n^2}, \\
\hat{U} &= [I_n \otimes U_1 \ V_2^T \otimes I_n] \in \mathbb{R}^{n^2 \times (r_1 + r_2)n}, \\
\hat{V} &= \left[ \begin{array}{c}
I_n \otimes V_1 \\
(U_2^T \otimes I_n)\Pi
\end{array} \right] \in \mathbb{R}^{(r_1 + r_2)n \times n^2},
\end{align*}
\]

where $A_0, B_0, U_1, U_2, V_1, V_2$ are matrices from (26) and $\Pi$ is a permutation matrix such that $\Pi \text{vec}(X) = \text{vec}(X^T)$. If matrices $\hat{L}_0$ and $\hat{L}_0 + \hat{U}\hat{V}$ are regular, then the solution of equation (26) is given by

$$
\text{vec}(X) = (\hat{L}_0^{-1} - \hat{L}_0^{-1}\hat{U}(I + \hat{V}\hat{L}_0^{-1}\hat{U})^{-1}\hat{V}\hat{L}_0^{-1}) \text{vec}(E). 
$$

**Proof.** Since $\text{vec}(X^T) = \Pi \text{vec}(X)$, where $\Pi$ is defined as in (33), one can use the Kronecker product and the vectorization operator to rewrite equation (26) as an $n^2 \times n^2$ linear matrix equation

\[
(I_n \otimes (A_0 + U_1 V_1) + ((B_0 + U_2 V_2)^T \otimes I_n)\Pi) \text{vec}(X) = \text{vec}(E).
\]

Now it is easy to see that equation (26) can be written as

\[
(\hat{L}_0 + \hat{U}\hat{V}) \text{vec}(X) = \text{vec}(E).
\]

After applying the Sherman-Morrison-Woodbury formula on the left-hand side of the previous equation, we obtain

\[
\text{vec}(X) = (\hat{L}_0 + \hat{U}\hat{V})^{-1} \text{vec}(E) = (\hat{L}_0^{-1} - \hat{L}_0^{-1}\hat{U}(I + \hat{V}\hat{L}_0^{-1}\hat{U})^{-1}\hat{V}\hat{L}_0^{-1}) \text{vec}(E).
\]

$\square$
3.1 Algorithm for solving $T$-Sylvester equation using Sherman-Morrison-Woodbury formula

Similarly to the case of the Sylvester equation, direct solving of equation (26) using formula (34) in general has complexity $O(n^6)$. But if matrices in equation (34) have a special structure, then we can construct an algorithm for a more efficient calculation of $X$.

Algorithm 2 contains a procedure for solving $T$-Sylvester equation (26) using formula (34) and taking into account the structures of input matrices.

**Algorithm 2** Basic procedure for solving equation (26) using Sherman-Morrison-Woodbury formula (34)

**Input:** $A_0, B_0, E \in \mathbb{R}^{n \times n}, U_1, V_1^T \in \mathbb{R}^{n \times r_1}, U_2, V_2^T \in \mathbb{R}^{n \times r_2}$

**Output:** $X \in \mathbb{R}^{n \times n}$

1. Calculate $x_1 = \hat{L}_0^{-1} \text{vec}(E)$ if $\hat{L}_0^{-1}$ is given or solve system $\hat{L}_0 x_1 = \text{vec}(E)$
2. Calculate $x_2 = \hat{V} x_1$
3. Solve $(I + \hat{V} \hat{L}_0^{-1} \hat{U}) x_3 = x_2$
4. Calculate $x_4 = \hat{U} x_3$
5. Calculate $x_5 = \hat{L}_0^{-1} x_4$ or solve $\hat{L}_0 x_5 = x_4$
6. $\text{vec}(X) = x_1 - x_5$

Basic parts in Algorithm 2 which have to be calculated carefully using the structure of given matrices are products of matrices $\hat{U}$, $\hat{V}$ by vectors and solving linear systems $\hat{L}_0 x = y$ and $(I + \hat{V} \hat{L}_0^{-1} \hat{U}) x = y$. 

If $\hat{L}_0^{-1}$ is not given, then solving the linear system $\hat{L}_0 x = y$ with $\hat{L}_0 = I_n \otimes A_0 + (B_0^T \otimes I_n) \Pi$ is equivalent to solving $T$-Sylvester equation 

$$A_0 \hat{X} + \hat{X}^T B_0 = \hat{Y},$$

where $\text{vec}(\hat{X}) = x$ or $\text{vec}(\hat{Y}) = y$. Without any assumed structure of $A_0, B_0$, this equation can be solved in $76n^3$ operations (see [13]). On the other hand, if $A_0$ and $B_0$ have some special structure, the number of operations can be significantly reduced. For example, if $A_0$ and $B_0$ are block diagonal matrices with small-dimensional blocks on the diagonal, the number of operations will be $O(n^2)$.

The product of matrices $\hat{U}$ and $\hat{V}$ by vectors $x \in \mathbb{R}^{(r_1+r_2)n}$ and $y \in \mathbb{R}^{n^2}$ in steps 4 and 2 can be done in the following way:

$$\hat{U} x = \begin{bmatrix} I_n \otimes U_1 & V_2^T \otimes I_n \end{bmatrix} \begin{bmatrix} \text{vec}(\hat{X}_1) \\ \text{vec}(\hat{X}_2) \end{bmatrix} = \text{vec}(U_1 \hat{X}_1 + \hat{X}_2 V_2),$$

where $\hat{X}_1 \in \mathbb{R}^{r_1 \times n}$, $\text{vec}(\hat{X}_1) = x(1 : r_1 n)$ and $\hat{X}_2 \in \mathbb{R}^{n \times r_2}$, $\text{vec}(\hat{X}_2) = x(r_1 n + 1 : (r_1 + r_2) n)$, and

$$\hat{V} y = \begin{bmatrix} I_n \otimes V_1 \\ (U_2^T \otimes I_n) \Pi \end{bmatrix} \text{vec}(\hat{Y}) = \begin{bmatrix} \text{vec}(V_1 \hat{Y}) \\ \text{vec}(\hat{Y}^T U_2) \end{bmatrix},$$

where $\hat{Y} \in \mathbb{R}^{n \times n}$, $\text{vec}(\hat{Y}) = y$. 

16
It is now easy to see that the number of arithmetic operations needed for each of products $\hat{U}x$ and $\hat{V}y$ is equal to $2n^2(r_1 + r_2)$.

For step 3 we need to form matrix $\hat{\Delta} := \hat{V}\hat{L}_0^{-1}\hat{V}$. Matrices $\hat{U}, \hat{V}$ and $\hat{L}_0$ are large so we cannot multiply them directly. Thus, we need a procedure for forming $\hat{\Delta}$ without forming $\hat{U}, \hat{V}, \hat{L}_0^{-1}$ and their direct multiplication. Every column of product $\text{vec}(\hat{X}_i) := \hat{L}_0^{-1}\hat{U}(:,i)$ can be calculated as a solution of the $T$-Sylvester equation

$$A_0\hat{X}_i + (\hat{X}_i)^TB_0 = \hat{U}_i,$$

where $\text{vec}(\hat{U}_i) = \hat{U}(:,i)$, $i = 1,\ldots,(r_1 + r_2)n$. Further, multiplication by matrix $\hat{V}$ can be done as follows:

$$\hat{\Delta} = \hat{V}\begin{bmatrix} \text{vec}(\hat{X}_1) & \text{vec}(\hat{X}_2) & \cdots & \text{vec}(\hat{X}_{(r_1+r_2)n}) \end{bmatrix}$$

$$= \begin{bmatrix} \text{vec}(V_1\hat{X}_1) & \text{vec}(V_1\hat{X}_2) & \cdots & \text{vec}(V_1\hat{X}_{(r_1+r_2)n}) \\ \text{vec}(X_1^TU_2) & \text{vec}(X_2^TU_2) & \cdots & \text{vec}(X_{(r_1+r_2)n}^TU_2) \end{bmatrix}$$

so $\hat{\Delta}$ can be constructed in $2n^3(r_1 + r_2)^2 + O(n^2(r_1 + r_2)^2)$ operations.

The choice of a method for solving a system with $(I + \Delta)$ depends on the problem structure.

In the following example we will compare accuracy and the number of operations when equation (26) is solved by the method given in [13] and by Algorithm 2 based on formula (34).

**Example 3** In this example we compare residuals and the number of operations for the algorithm given in [13] and Algorithm 2 based on formula (34) for solving the parameter dependent $T$-Sylvester equation of the form

$$(A_0 + vU_1V_1)X + X^TB_0 + vU_2V_2 = E, \quad (35)$$

where

$$A_0 = \text{rand}(n), \quad B_0 = A_0 - 5I,$$

$$U_1 = [\text{rand}(n,1),e_2], \quad U_2 = [\text{rand}(n,1),\text{rand}(n,1)], \quad V_1 = V_2 = U_1^T.$$ 

The linear system $(I + \Delta)x = y$ in step 3 of Algorithm 2 will be solved by using Hessenberg form of $\Delta$ (denoted by $\text{SMW-exact}$) and by FOM (denoted by $\text{SMW-approx}$).

Results are given in Table 3, where $d$ denotes the order of the matrix $\hat{\Delta}$. Values $r_S,r_E,r_A$ are norms of residuals, i.e. $\|(A_0 + vU_1V_1)X + X^TB_0 + vU_2V_2 - E\|$, where $X$ is obtained by the method from [13] (denoted by $\text{GShur}$), $\text{SMW-exact}$ and $\text{SMW-approx}$ method. In $\text{SMW-approx}$ column, $k$ denotes a dimension of Krylov subspace in FOM.

In what follows we will observe a number of operations in the case when equation (35) has to be solved many times with different values of parameter $v \in \mathbb{R}$. For example, this situation can occur during the optimization of the solution $X(v)$ with respect to $v \in \mathbb{R}$. 

17
The method from [13] has complexity $76n^3$ for every value of $v$.

Solving (35) when the system with matrix $(I + v\Delta)$ is solved using reduction of $\Delta$ to Hessenberg form for the first value of $v$ costs $O(r^3n^3)$ and every additional solving of (35) with other value of $v$ costs only $O(r^2n^2)$.

If the system is solved approximately by the FOM method, the number of operations for the first value of $v$ is $O(rkn^2)$, and every additional solving of (35) with other value of $v$ costs $O(rn^2)$.

In Figure 4, growth in the number of flops for all three methods is given when equation (35) is solved a hundred times with different values of $v$, where $k$ in the FOM method is taken to be $n/10$.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>GSchur</th>
<th>SMW-exact</th>
<th>SMW-approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 50$ ($d = 200$)</td>
<td>$r_S = 2.3 \cdot 10^{-14}$</td>
<td>$r_E = 3.1 \cdot 10^{-14}$</td>
<td>$k = 15, r_A = 1.1 \cdot 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$k = 25, r_A = 4.6 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>$n = 100$ ($d = 400$)</td>
<td>$r_S = 1.1 \cdot 10^{-13}$</td>
<td>$r_E = 2.1 \cdot 10^{-13}$</td>
<td>$k = 20, r_A = 1.1 \cdot 10^{-6}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$k = 30, r_A = 2.9 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>$n = 150$ ($d = 600$)</td>
<td>$r_S = 5.1 \cdot 10^{-13}$</td>
<td>$r_E = 1.6 \cdot 10^{-12}$</td>
<td>$k = 20, r_A = 5.2 \cdot 10^{-6}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$k = 30, r_A = 3.2 \cdot 10^{-11}$</td>
</tr>
</tbody>
</table>

Table 3: Results of comparing GSchur, SMW-exact and SMW-approx approaches

![Figure 4: Operation number growth for GSchur, SMW-exact and SMW-approx algorithms](image)

References


