Bollettino dell'Unione Matematica Italiana (2020) 13:429–439
https://doi.org/10.1007/s40574-020-00247-4

Numerical solution of a class of third order tensor linear equations

V. Simoncini[1,2]

Received: 4 May 2020 / Accepted: 8 July 2020 / Published online: 20 July 2020

Abstract
We propose a new dense method for determining the numerical solution to a class of third order tensor linear equations. The approach does not require the use of the coefficient matrix in Kronecker form, thus it allows the treatment of structured very large problems. A particular version of the method for symmetric matrices is also discussed. Numerical experiments illustrate the properties of the proposed algorithm.

Keywords Tensors · Linear algebraic equations · Schur decomposition

Mathematics Subject Classification 65F10

1 Introduction

We are interested in the numerical computation of the unique solution $\boldsymbol{X} \in \mathbb{R}^{n\times n\times n}$ to the nonsingular system $\mathcal{A}\boldsymbol{x} = b$ written in the following tensor form

$$(M_1 \otimes A_1 \otimes \boldsymbol{H} + A_2 \otimes \boldsymbol{M} \otimes \boldsymbol{H} + H_3 \otimes \boldsymbol{M} \otimes A_3)\mathrm{vec}(\boldsymbol{X}) = b_3 \otimes b_2 \otimes b_1 \qquad (1.1)$$

where all coefficient matrices are real and have the same $n \times n$ dimensions. Here $\otimes$ denotes the Kronecker product (to be recalled later) and $\mathrm{vec}(X)$ stacks the components of the tensor X one after the other. In particular, in (1.1) two terms share the same matrices, either $\boldsymbol{M}$ or $\boldsymbol{H}$ (purposely in bold face), while all other matrices A_i,$i = 1, 2, 3$ and H_3, M_1 have no relation to each other. The only assumption on the coefficient matrices, in addition to the nonsingularity of $\mathcal{A}$, is that $\boldsymbol{M}$, $\boldsymbol{H}$ and M_1, H_3 be nonsingular. The *unknown* solution tensor will also be highlighted in bold face, to emphasize that this is the array to be determined.

This tensor equation is representative of a large class of problems that can be described by means of tensors and formulated as linear array equation. For instance, the discretization of

This version dated 24 June 2020.

✉ V. Simoncini
valeria.simoncini@unibo.it

[1] Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, 40127 Bologna, Italy

[2] IMATI-CNR, Pavia, Italy

three dimensional partial differential equations by means of a tensor basis, as is the case for finite differences on parallelepipedal domains or certain spectral methods, can lead to tensor equations of type (1.1). Tensor equations have become a fundamental algebraic ingredient for the numerical treatment of mathematical models depending on many parameters, as is the case in uncertainty quantification and parameter-dependent model order reduction methodologies; see, e.g., [1,4,5,15,20,23]. In typical situations, tensor equations with many terms occur, and each term may have a number of Kronecker products. In a simplified context, for instance, the following tensor equation is of interest (see, e.g., [17])

$$\sum_{i=1}^{\ell} \left(I \otimes \cdots \otimes \overset{i}{A_i} \otimes \cdots \otimes I \right) \boldsymbol{x} = \bigotimes_{i=1}^{\ell} b_i$$

For $\ell = 3$ this is a special case of our problem (1.1), where all matrices except A_i, $i = 1, 2, 3$ are equal to the identity matrix I. Finally, we explicitly remark that if the right-hand side in (1.1) were the sum of rank-one tensors, then the solution could be expressed as the sum of solutions to tensor equations with right-hand sides of rank one.

The literature on tensors - their analysis and the associated approximation methods - has grown tremendously in the past twenty years. Different tensor representations and decompositions have been analyzed. We refer the reader to [16] for an introductory and historical account, and to [11] for a literature survey up to 2013. Numerous different decompositions have allowed the developments of various problem dependent strategies, see, e.g., [7,12,14,18,24].

More recently, methods for solving linear equations in tensor format have been proposed and analyzed. In most cases, the authors have been interested in the presence of many summands and many Kronecker products, for which iterative methods appear to be mandatory. In this context, most approaches try to take into account the Kronecker structure and the possible low rank of the involved iteration matrices, see, e.g., [2,3,6,13,19–21]. However, little has been said on "direct" dense methods for low order tensor equations, without the explicit use of the Kronecker form. Here we close this gap for the special case of (1.1), which nonetheless appears to be a feasible algebraic formulation of a quite large set of differential problems.

2 A closed form solution

The numerical solution to (1.1) can be given in closed form by unfolding the 3-mode tensor in one of the three directions. In particular, a tensor $X \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ can be written using the mode-1 unfolding as (see, e.g., [16])

$$X_{(1)} = [X_1, X_2, \ldots, X_{n_3}], \quad X_j \in \mathbb{R}^{n_1 \times n_2},\ j = 1, \ldots, n_3;$$

each X_j is called a slice, and $X_{(1)}$ is a matrix in $\mathbb{R}^{n_1 \times n_2 n_3}$. Some additional standard notation needs to be recalled. The Kronecker product of two matrices X, Y is defined in the standard block form as

$$X \otimes Y = \begin{bmatrix} X_{1,1}Y & \cdots & X_{1,n_2}Y \\ \vdots & \ddots & \vdots \\ X_{n_1,1}Y & \cdots & X_{n_1,n_2}Y \end{bmatrix},$$

where $X_{i,j}$ denotes an element of X. Moreover, $\mathrm{vec}(X)$ is the operator stacking all columns of the matrix X one after the other. In the case of third order tensors, we will apply the vec

operator to the mode-1 unfolding. The reverse operation, for known dimensions of the vector x, will be denoted by $\mathrm{mat}(x, n_1, n_2)$, so that $x = \mathrm{vec}(X)$ and $X = \mathrm{mat}(x, n_1, n_2)$. Similarly, $X = \mathrm{tensor}_{(1)}(x, n_1, n_2, n_3)$ will fold a long vector into a tensor via the mode-1 unfolding. A standard property of the Kronecker product that will be used repeatedly is the following

$$\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X), \tag{2.1}$$

(B^T denotes the real transpose of B), which allows one to go back and forth between the vector and matrix notations. Other properties used in the sequel are i) $(A \otimes B)^T = A^T \otimes B^T$, ii) $(A \otimes B)(C \otimes D) = (AC \otimes BD)$ for compatible matrix dimensions, iii) if A and B are both invertible, then $A \otimes B$ is invertible and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$; see, e.g., [9, Ch.12].

The following result holds. Here Q^* denotes the conjugate transpose of the complex matrix Q, while $H^{-T} = (H^{-1})^T$ and T denotes transposition.

Theorem 2.1 *Let $A_3^T \boldsymbol{H}^{-T} = QRQ^*$ be the Schur decomposition of $A_3^T \boldsymbol{H}^{-T}$, and $[\gamma_1, \ldots, \gamma_n] := b_1^T \boldsymbol{H}^{-T} Q$. Using the mode-1 unfolding, the solution $\boldsymbol{X}$ to (1.1) is given by*

$$\boldsymbol{X}_{(1)} = [Q^{-T}\tilde{Z}_1^T, \ldots, Q^{-T}\tilde{Z}_n^T] \in \mathbb{R}^{n \times n^2},$$

where for $j = 1, \ldots, n$, the matrix $\tilde{Z}_j$ solves the generalized Sylvester equation

$$\boldsymbol{M}Z(H_3^T R_{j,j} + A_2^T) + A_1 Z M_1^T = b_2 \gamma_j b_3^T - \mathrm{mat}(\tilde{Z}_{j-1} R_{1:j-1,j}, n, n) H_3^T.$$

where $R_{j,j}$ denotes the (j, j) element of the upper triangular matrix R and $R_{1:j-1,j}$ the first $j-1$ components of its jth column; we define $\mathrm{mat}(\tilde{Z}_{j-1} R_{1:j-1,j}, n, n) H_3^T$ to be an empty array for $j = 1$.

Proof Using (2.1) for the unfolded tensor we have

$$\begin{aligned}
\boldsymbol{H}\boldsymbol{X}_{(1)}(A_2 \otimes \boldsymbol{M} + M_1 \otimes A_1)^T + A_3 \boldsymbol{X}_{(1)}(H_3 \otimes \boldsymbol{M})^T &= b_1(b_3 \otimes b_2)^T \\
\boldsymbol{X}_{(1)}(A_2 \otimes \boldsymbol{M} + M_1 \otimes A_1)^T (H_3 \otimes \boldsymbol{M})^{-T} + \boldsymbol{H}^{-1} A_3 \boldsymbol{X}_{(1)} &= \boldsymbol{H}^{-1} b_1 (b_3 \otimes b_2)^T (H_3 \otimes \boldsymbol{M})^{-T} \\
\boldsymbol{X}_{(1)}(A_2^T H_3^{-T} \otimes I + M_1^T H_3^{-T} \otimes A_1^T \boldsymbol{M}^{-T}) + \boldsymbol{H}^{-1} A_3 \boldsymbol{X}_{(1)} &= \boldsymbol{H}^{-1} b_1 (b_3^T H_3^{-T} \otimes b_2^T \boldsymbol{M}^{-T})
\end{aligned}$$

For later readability, let us transpose both sides and set $Y = (\boldsymbol{X}_{(1)})^T$. Then we obtain

$$(H_3^{-1}A_2 \otimes I + H_3^{-1}M_1 \otimes \boldsymbol{M}^{-1}A_1)Y + Y(\boldsymbol{H}^{-1}A_3)^T = (H_3^{-1}b_3 \otimes \boldsymbol{M}^{-1}b_2) b_1^T \boldsymbol{H}^{-T}.$$

Using $(\boldsymbol{H}^{-1}A_3)^T = QRQ^*$ and multiplying the equation by Q from the right, we can write

$$(H_3^{-1}A_2 \otimes I + H_3^{-1}M_1 \otimes \boldsymbol{M}^{-1}A_1)YQ + YQR = (H_3^{-1}b_3 \otimes \boldsymbol{M}^{-1}b_2) b_1^T \boldsymbol{H}^{-T} Q.$$

Let $b_1^T \boldsymbol{H}^{-T} Q =: [\gamma_1, \ldots, \gamma_k]$ and $YQ =: [\hat{z}_1, \ldots, \hat{z}_n]$. Thanks to the upper triangular form of R, for the first column $\hat{z}_1$ it holds

$$(H_3^{-1}A_2 \otimes I + H_3^{-1}M_1 \otimes \boldsymbol{M}^{-1}A_1)\hat{z}_1 + \hat{z}_1 R_{1,1} = (H_3^{-1}b_3 \otimes \boldsymbol{M}^{-1}b_2)\gamma_1. \tag{2.2}$$

For the subsequent columns $j = 2, \ldots, n$, taking into account once again the triangular form of R, we set $w_{j-1} = [\hat{z}_1, \ldots, \hat{z}_{j-1}] R_{1:j-1,j}$ so that

$$(H_3^{-1}A_2 \otimes I + H_3^{-1}M_1 \otimes \boldsymbol{M}^{-1}A_1)\hat{z}_j + \hat{z}_j R_{j,j} = (H_3^{-1}b_3 \otimes \boldsymbol{M}^{-1}b_2)\gamma_1 - w_{j-1}. \tag{2.3}$$

Each column can be obtained in sequence by further unmaking the Kronecker product as follows. Let us reshape each $\hat{z}_j$ so that $\hat{z}_j = \mathrm{vec}(\hat{Z}_j)$. By using (2.1) in (2.2) for $j = 1$, we can write

$$\boldsymbol{M}^{-1} A_1 \hat{Z}_1 (H_3^{-1} M_1)^T + \hat{Z}_1 (R_{1,1} I + (H_3^{-1} A_2)^T) = \boldsymbol{M}^{-1} b_2 \gamma_1 (H_3^{-1} b_3)^T,$$

which can be written as

$$\boldsymbol{M}^{-1} A_1 \hat{Z}_1 + \hat{Z}_1 (R_{1,1} H_3^T M_1^{-T} + A_2^T M_1^{-T}) = \boldsymbol{M}^{-1} b_2 \gamma_1 b_3^T M_1^{-T}, \quad j = 2, \ldots, n. \tag{2.4}$$

Analogously, for $j = 2, \ldots, n$ and letting $W_{j-1} = \mathrm{mat}([\hat{z}_1, \ldots, \hat{z}_{j-1}] R_{1:j-1,j})$, from (2.3) we first obtain

$$\boldsymbol{M}^{-1} A_1 \hat{Z}_j (H_3^{-1} M_1)^T + \hat{Z}_j (R_{j,j} I + (H_3^{-1} A_2)^T) = \boldsymbol{M}^{-1} b_2 \gamma_j (H_3^{-1} b_3)^T - W_{j-1} H_3^{-T},$$

or equivalently, for $j = 2, \ldots, n$, as

$$\boldsymbol{M}^{-1} A_1 \hat{Z}_j + \hat{Z}_j (R_{j,j} H_3^T M_1^{-T} + A_2^T M_1^{-T}) = \boldsymbol{M}^{-1} b_2 \gamma_1 b_3^T M_1^{-T} W_{j-1} M_1^{-T}. \tag{2.5}$$

Multiplying both sides by M_1 (from the right) and by $\boldsymbol{M}^T$ (from the left), the result follows. □

We notice that the use of the mode-1 unfolding is related to the specific location of the repeated matrices $\boldsymbol{H}, \boldsymbol{M}$. Different unfoldings could be used if these matrices occupy different positions. We also remark that the same procedure could be applied if data were complex, without any particular change in the proof, or in the algorithm below, except that one should keep in mind that property (2.1) uses real transposition, even if data are complex.

3 The new algorithm

The proof used for Theorem 2.1 is constructive, as it provides an explicit way to generate the tensor solution, one slice at the time. The complete procedure is described in the algorithm below, in the following called the Three-Term-Tensor Sylvester (T^3- SYLV) method.

Algorithm T^3- SYLV.

Input: $A_1, A_2, A_3, M_1, \boldsymbol{M}, \boldsymbol{H}, H_3$ of size $n \times n$, b_1, b_2, b_3 of length n

For $k = 1, \ldots, n$

Compute Q, R such that $A_3^T \boldsymbol{H}^{-T} = Q R Q^*$ (Schur decomposition)

Compute the vector $g = b_1^T \boldsymbol{H}^{-T} Q$

Set $F = \boldsymbol{M}^{-1} b_2 g_k b_3^T M_1^{-T}$

if $k > 1$, $W_{k-1} = \mathrm{mat}(\widehat{Z}_{:,1:k-1} R_{1:k-1,k}, n, n)$, and $F = F - W_{k-1} H_3^T M_1^{-T}$

Solve $\boldsymbol{M}^{-1} A_1 \widehat{Z} + \widehat{Z} (R_{k,k} H_3^T M_1^{-T} + A_2^T M_1^{-T}) = F$ to get $\widehat{Z}_k$

end

Set $\boldsymbol{X}_{(1)} = Q[\mathrm{vec}(\widehat{Z}_1), \ldots, \mathrm{vec}(\widehat{Z}_n)]^T$

Set $\boldsymbol{X} = \mathrm{tensor}_{(1)}(x, n, n, n)$

Output: $\boldsymbol{X}$ solution to (1.1)

In practice, using appropriate transformations, the method is a nested Sylvester solver, which treats one slice at the time, and updates the corresponding coefficient matrix and right-hand side F. The solvability of the Sylvester equations is related to that of the original problem, and in particular to the nonsingularity of $\mathcal{A}$.

The algorithm relies on the initial Schur decomposition, which provides robust unitary transformations. Moreover, for each slice, a matrix Sylvester equation needs to be solved,

Table 1 CPU times (in secs) of T^3- SYLV for increasing dimensions of the coefficient matrices, having uniformly distributed random entries

n	CPU Time
8	1.36e-02
16	8.01e-03
32	4.81e-02
64	3.00e-01
128	3.48e+00
256	3.41e+01

whose solution also involves the Schur decompositions of the coefficient matrices, see, e.g., [26]; its sensitivity has been analyzed in [8, sec.4.1]. Stability of the overall algorithm is also affected by the presence of several inverses, which can be harmful in case of ill-conditioned matrices. Indeed, if some of the involved matrices are severely ill-conditioned, the solution may lose accuracy. This fact was experimentally observed in our experiments, some of which are reported in Example 4.3.

3.1 Numerical experiments

In this section we report on some numerical experiments with the T^3- SYLV method. All experiments were performed using Matlab [22].

Example 3.1 To test the efficiency of the method, we consider dense matrices with random entries (taken from a uniform distribution in the interval (0,1), Matlab function `rand`) of increasing size n. The same is used for the vectors b_1, b_2, b_3. We stress that the Kronecker form of the problem would involve a dense matrix $\mathcal{A}$ of size $n^3 \times n^3$, which could not even be stored.

We readily observe that the method is able to solve a (random) structured dense problem of size $n^3 = 16,777,216$ in about 34 seconds on a standard laptop. The CPU times in Table 1 show that the computational cost of the method approximately grows between six and ten times as the dimension n doubles. However, going from n to $2n$, the problem dimension in the full space would grow from n^3 to $2^3 n^3$. Hence, the actual cost appears to grow linearly with n^3. Since data are dense, Gauss elimination on $\mathcal{A}$ would instead require $\mathcal{O}((n^3)^3)$ floating point operations.

4 The symmetric case

If all matrices are symmetric and positive definite, the solution is also symmetric and positive semidefinite, in the appropriate tensor representation; see, e.g., [10] for a general discussion. With these hypotheses, the derivation of the solution procedure simplifies accordingly, as shown in the following result. We stress that many problems can be brought to this setting. Consider for instance the differential equation $-\Delta u = f$ on the unit cube with zero Dirichlet boundary conditions. By discretizing using linear finite elements in each direction (this may be seen as a linear finite element discretization using $\boldsymbol{Q}_1$ brick elements), we obtain

$$(M \otimes A \otimes M + A \otimes M \otimes M + M \otimes M \otimes A)\mathrm{vec}(\boldsymbol{X}) = F,$$

with $M = \text{tridiag}(-1, \underline{4}, -1) \in \mathbb{R}^{n\times n}$ and $A = \text{tridiag}(-1, \underline{2}, -1) \in \mathbb{R}^{n\times n}$ (the underlined number corresponds to the diagonal entry in the two symmetric and tridiagonal matrices). If f can be well approximated by a separable function in spatial dimensions, then F will have the desired Kronecker form, see, e.g., [10,17] for a similar description[1].

Analogously, one could consider the equation $\mathcal{L}(u) = f(x, y, z)$, $(x, y, z) \in \Omega \subset \mathbb{R}^3$ and

$$\mathcal{L}(u) = -\boldsymbol{m}(z)h_3(y)\phi_1(x)u_{xx} - m_1(x)\boldsymbol{h}(z)\phi_2(y)u_{yy} - \boldsymbol{h}(x)\boldsymbol{m}(y)\phi_3(z)u_{zz},$$

while $f(x, y, z) = f_1(x) f_2(y) f_3(z)$. Finite difference discretization leads to the Kronecker form in (1.1), where, $\boldsymbol{M}$ is a diagonal matrix containing the coefficients in $\boldsymbol{m}(z_k)$ at the interior nodes z_k in the z-direction; similarly for the other coefficients. The matrices A_i, $i = 1, 2, 3$ contain the three-point stencil of the discretized second order derivative in each direction, respectively, together with the coefficient ϕ_i; see, e.g., [25].

We do not report numerical results with data stemming from these discretizations, as they would not significantly differ from those shown in our examples, for dense data.

We then describe the specialized result for symmetric and positive definite matrices. This leads to better stability properties of the algorithm (see Example 4.3).

Proposition 4.1 *Assume all coefficient matrices in (1.1) are symmetric and positive definite, and assume that $b \equiv b_1 = b_2 = b_3$. Let $\boldsymbol{H} = \boldsymbol{L_H}\boldsymbol{L_H}^T$, $\boldsymbol{M} = \boldsymbol{L_M}\boldsymbol{L_M}^T$, $H_3 = L_3 L_3^T$ and $L_3^{-1} M_1 L_3^{-T} = L_1 L_1^T$ be the Cholesky factorizations of the corresponding matrices. Let $\widehat{A}_3 = Q\Lambda Q^T$ be the eigenvalue decomposition of $\widehat{A}_3 = \boldsymbol{L_H}^{-1} A_3 \boldsymbol{L_H}^{-T}$ and $[\gamma_1, \ldots, \gamma_n] := b^T \boldsymbol{L_H}^{-1} Q$. Using the mode-1 unfolding, the solution $\boldsymbol{X}$ to (1.1) is given by*

$$\boldsymbol{X}_{(1)} = \boldsymbol{L_H}^{-1} Q[\text{vec}(\boldsymbol{L_M}\widehat{Z}_1 L_3^{-1}), \ldots, \text{vec}(\boldsymbol{L_M}\widehat{Z}_n L_3^{-1})]^T,$$

where for $j = 1, \ldots, n$, $\widehat{Z}_1 = \widetilde{Z}_j L_1^{-1}$ and the matrix $\widetilde{Z}_j$ solves the generalized Sylvester equation

$$\widetilde{Z}_j L_1^{-1}(\lambda_j I + \widehat{A}_2)L_1^{-T} + \widehat{A}_1\widetilde{Z}_1 = \boldsymbol{L_M}^{-1} b\gamma_j (L_3^{-1}b)^T L_1^{-T}, \quad j = 1, \ldots, n,$$

with $\widehat{A}_2 = L_3^{-1} A_2 L_3^{-T}$ and $\widehat{A}_1 = \boldsymbol{L_M}^{-1} A_1 \boldsymbol{L_M}^{-T}$.

Proof. Consider the following Cholesky factorizations of the given matrices

$$\boldsymbol{H} = \boldsymbol{L_H}\boldsymbol{L_H}^T, \quad H_3 \otimes \boldsymbol{M} = (L_3 \otimes \boldsymbol{L_M})(L_3^T \otimes \boldsymbol{L_M}^T).$$

Using (2.1) for the unfolded tensor we have

$$\begin{aligned}
&\boldsymbol{H}\boldsymbol{X}_{(1)}(A_2 \otimes \boldsymbol{M} + M_1 \otimes A_1) + A_3\boldsymbol{X}_{(1)}(H_3 \otimes \boldsymbol{M}) = b(b \otimes b)^T \\
&\boldsymbol{L_H}^T\boldsymbol{X}_{(1)}(A_2 \otimes \boldsymbol{M} + M_1 \otimes A_1)(L_3^{-T} \otimes \boldsymbol{L_M}^{-T}) + \boldsymbol{L_H}^{-1}A_3\boldsymbol{L_H}^{-T}\boldsymbol{L_H}^T\boldsymbol{X}_{(1)}(L_3 \otimes \boldsymbol{L_M}) \\
&\quad = \boldsymbol{L_H}^{-1} b(b \otimes b)^T (L_3^{-T} \otimes \boldsymbol{L_M}^{-T}) \\
&\widehat{\boldsymbol{X}}_{(1)}(\widehat{A}_2 \otimes I + \widehat{M}_1 \otimes \widehat{A}_1) + \widehat{A}_3\widehat{\boldsymbol{X}}_{(1)} = \widehat{b}(b^T L_3^{-T} \otimes b^T \boldsymbol{L_M}^{-T})
\end{aligned}$$

where we have defined $\widehat{b} = \boldsymbol{L_H}^{-1} b$, $\widehat{\boldsymbol{X}}_{(1)} = \boldsymbol{L_H}^T \boldsymbol{X}_{(1)}(L_3 \otimes \boldsymbol{L_M})$, $\widehat{A}_3 = \boldsymbol{L_H}^{-1} A_3 \boldsymbol{L_H}^{-T}$, $\widehat{A}_2 = L_3^{-1} A_2 L_3^{-T}$, $\widehat{M}_1 = L_3^{-1} M_1 L_3^{-T}$ and $\widehat{A}_1 = \boldsymbol{L_M}^{-1} A_1 \boldsymbol{L_M}^{-T}$. Note that all these "hat" coefficient

[1] We recall that for this and other specialized settings, cyclic reduction and Fourier analysis could be classical viable effective alternatives; see, e.g., [27,28].

matrices are still symmetric and positive definite. For later readability, let us transpose both sides and set $Y = (\widehat{\boldsymbol{X}}_{(1)})^T$. Then we obtain

$$(\widehat{A}_2 \otimes I + \widehat{M}_1 \otimes \widehat{A}_1)Y + Y\widehat{A}_3 = (L_3^{-1}b \otimes \boldsymbol{L}_{\boldsymbol{M}}^{-1}b)\widehat{b}^T.$$

Using $\widehat{A}_3 = Q\Lambda Q^T$ and multiplying the equation by Q from the right, we can write

$$(\widehat{A}_2 \otimes I + \widehat{M}_1 \otimes \widehat{A}_1)YQ + YQ\Lambda = (L_3^{-1}b \otimes \boldsymbol{L}_{\boldsymbol{M}}^{-1}b)\widehat{b}^T Q.$$

Let $\widehat{b}^T Q =: [\gamma_1, \ldots, \gamma_k]$ and $YQ =: [\hat{z}_1, \ldots, \hat{z}_n]$. Thanks to the diagonal form of Λ, namely $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, for each column $\hat{z}_j$ it holds

$$(\widehat{A}_2 \otimes I + \widehat{M}_1 \otimes \widehat{A}_1)\hat{z}_j + \hat{z}_j\lambda_j = (L_3^{-1}b \otimes \boldsymbol{L}_{\boldsymbol{M}}^{-1}b)\gamma_j. \tag{4.1}$$

Each column can then be obtained in sequence by further unmaking the Kronecker product as follows. Let us reshape each $\hat{z}_j$ so that $\hat{z}_j = \mathrm{vec}(\widehat{Z}_j)$. By using (2.1) in (4.1), we can write

$$\widehat{Z}_j(\lambda_j I + \widehat{A}_2) + \widehat{A}_1\widehat{Z}_1\widehat{M}_1 = \boldsymbol{L}_{\boldsymbol{M}}^{-1}b\gamma_j(L_3^{-1}b)^T,$$

which, upon factorization of $\widehat{M}_1$ as $\widehat{M}_1 = L_1L_1^T$ can be written as

$$\widetilde{Z}_j L_1^{-1}(\lambda_j I + \widehat{A}_2)L_1^{-T} + \widehat{A}_1\widetilde{Z}_1 = \boldsymbol{L}_{\boldsymbol{M}}^{-1}b\gamma_j(L_3^{-1}b)^T L_1^{-T}, \quad j = 1, \ldots, n, \tag{4.2}$$

where $\widetilde{Z}_j = \widehat{Z}_j L_1$. The matrix equation (4.2) is a standard Sylvester equation, which can be solved for each j. Once $\widehat{Z}_j$ is recovered, we obtain

$$\boldsymbol{X}_{(1)} = \boldsymbol{L}_{\boldsymbol{H}}^{-1}Q[\mathrm{vec}(\boldsymbol{L}_{\boldsymbol{M}}\widehat{Z}_1L_3^{-1}), \ldots, \mathrm{vec}(\boldsymbol{L}_{\boldsymbol{M}}\widehat{Z}_nL_3^{-1})]^T. \quad \square$$

We will call the corresponding algorithm T^3- SYM- SYLV. Its Matlab implementation is reported in the Appendix.

We notice that the proof does not require all matrices to be positive definite, and A_3 only needs to be symmetric. In fact, the matrices A_2 and A_1 do not even need to be symmetric, although the current proof omits the corresponding transpositions. The proof could be easily adapted to treat this setting. Indeed, for A_1, A_2 nonsymmetric the Sylvester equation (4.2) could be written as

$$\widetilde{Z}_j L_1^{-1}(\lambda_j I + \widehat{A}_2^T)L_1^{-T} + \widehat{A}_1\widetilde{Z}_1 = \boldsymbol{L}_{\boldsymbol{M}}^{-1}b\gamma_j(L_3^{-1}b)^T L_1^{-T}, \quad j = 1, \ldots, n.$$

The rest of the derivation would follow as before.

Remark 4.2 We stated Proposition 4.1 for the right-hand side $b \otimes b \otimes b$ to maintain symmetry in the overall problem. Nonetheless, the "symmetric" simplifications still hold also in the more general situation where the right-hand side is $b_1 \otimes b_2 \otimes b_3$. In this case, the proof goes through in the same way, with obvious modifications to the right-hand side terms, following the steps in the proof of Theorem 2.1.

Example 4.3 We investigate the accuracy of the symmetric procedure, compared with that of the general algorithm T^3- SYLV, as the condition number of the given matrices increases. We consider 5×5 symmetric and positive definite matrices with random entries taken from a uniform distribution in the interval (0,1). To this end, we define each coefficient matrix by using the Matlab function `sprandsym` with density 1 (giving a full matrix), type 1 (positive definite), and the same reciprocal condition number $\kappa^{-1} = .2\,10^{-k}$, with $k = 0, 0.2, 0.4, \ldots, 10$. The results are reported in Fig. 1, where the relative errors

$$\frac{\|x^* - x_{nonsym}\|}{\|x^*\|}, \quad \frac{\|x^* - x_{sym}\|}{\|x^*\|}$$

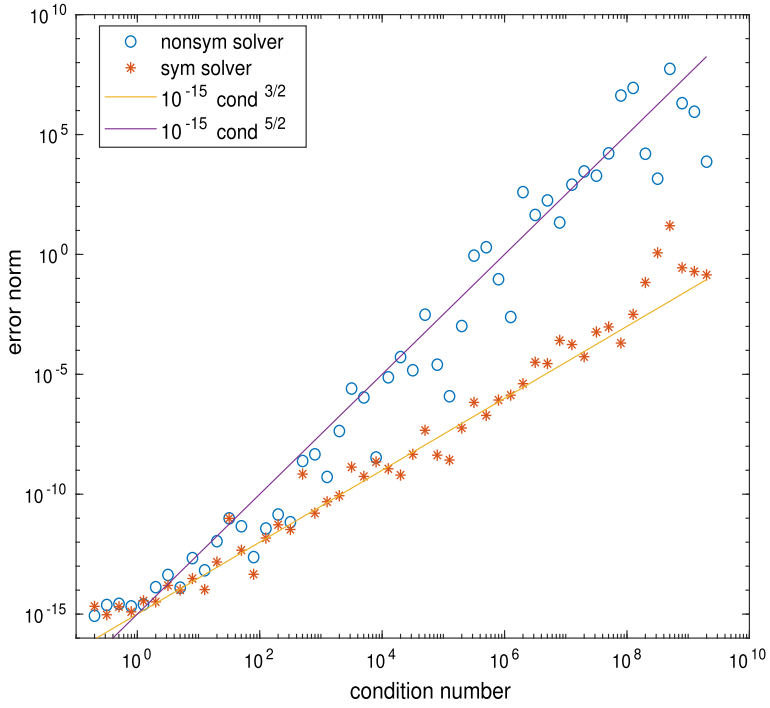

Fig. 1 Dependence of the symmetric and nonsymmetric solver accuracy on the matrices condition number

are displayed, with x_{nonsym} obtained by algorithm T^3-SYLV and x_{sym} by T^3-SYM- SYLV; here x^* is a *reference* solution, obtained by using the Matlab direct solver "\" on the Kronecker form of the problem with coefficient matrix of size 125×125. The figure also displays the quantities $10^{-15}\kappa^{3/2}$ and $10^{-15}\kappa^{5/2}$, which appear to match the dependence of the error on the matrices conditioning when employing either of the two methods, respectively. Clearly, the possibility of using the symmetric solver significantly improves the accuracy of the obtained solution, with respect to the problem condition number. The sensitivity of the method deserves a deeper analysis that will be performed in future work.

5 Conclusions

We have proposed a new method for solving order-3 tensor linear equations. We derived a general approach relying on the Schur decomposition, and a specialized one that effectively exploits the symmetric positive definiteness of the coefficient matrices.

Although the considered class of tensor equations is restricted by the role and position of the two matrices $\boldsymbol{H}$ and $\boldsymbol{M}$, our presentation shows that this setting is sufficiently general to represent a good variety of practical problems. On the other hand, the repeated presence of $\boldsymbol{M}$ and $\boldsymbol{H}$ forced us to use the same dimensions in all modes. We will try to relax this constraint in future work. We also remark that the algebraic problem could be formulated with these two repeated matrices located in other (different) positions in the tensor equation, in a way that a similar solution derivation could be devised.

Since the right-hand side has low numerical tensor rank, we expect $\boldsymbol{X}$ to have low tensor rank [10], [20, Th.3.6]. Tensor-based truncation strategies could be employed to satisfactorily approximate the obtained solution. This would avoid storing the whole dense tensor, if dimensions become large.

The proposed strategy allows us to solve with an essentially direct method problems of structured large dimensions. Nonetheless, if n is required to be significantly larger and the coefficient matrices are sparse, then an iterative variant of the proposed method could be considered. As an alternative, the new method can serve as workhorse for solving the reduced equation in projection type procedures for large and sparse third order tensor equations; see similar strategies in [26] for linear matrix equations.

Acknowledgements Open access funding provided by Alma Mater Studiorum-Universite di Bologna within the CRUI-CARE Agreement. The author is a member of Indam-GNCS. Its support is gratefully acknowledged. Part of this work was also supported by the Grant AlmaIdea 2017-2020 - Università di Bologna. We thank an anonymous reviewer for pointing to a relevant reference and for helpful remarks on the presentation.

Compliance with ethical standards

Conflict of interest There are no conflicts of interest.

Appendix

We report the Matlab code of the symmetric version of the algorithm, T^3- SYM- SYLV, as derived in Proposition 4.1.

```
function [xfinal,X]=T3_sym_Sylv(A1,A2,A3,M1,M2,H1,H3,b,check_res)
%function [xfinal,X]=T3_sym_Sylv(A1,A2,A3,M1,M2,H1,H3,b,check_res)
%
% Solve   G * xfinal - f = 0
%  G = kron(kron(M1,A1),H1)+kron(kron(A2,M2),H1)+kron(kron(H3,M2),A3) ]
%  f = kron(kron(b,b),b))
%
%  All coeff matrices are nxn, symmetric and positive definite
%
%  X = tensor(xfinal,n,n,n)
%

L_H=chol(H1,'lower');
L_M=chol(M2,'lower');
L3=chol(H3,'lower');
[Q,Lambda]=eig(full( L_H\A3/L_H' ));

nrmrhs=norm(kron(kron(b,b),b));
g =  (b'/L_H')*Q;  % row vector
ng=length(g);
[n1,n2]=size(A1);
I=eye(n1,n2);
```

```
A2hat=L3\A2/L3';
M1hat=L3\M1/L3';
A1hat=L_M\A1/L_M';
L1=chol(M1hat,'lower');

for k=1:ng
  rhs = L_M\b*g(k)*(b'/L3')/L1';
  Ztilde=lyap(A1hat,  L1\(Lambda(k,k)*I + A2hat)/L1', -rhs);
  Zhat=L_M'\Ztilde/L1/L3;
  zz(:,k)=reshape(Zhat,n1*n2,1);
end

xx=L_H'\Q*zz';
xfinal=real(xx(:));
X=reshape(real(xx),n1,n2,n2);

if check_res
  G=kron(kron(M1,A1),H1)+kron(kron(A2,M2),H1)+kron(kron(H3,M2),A3);
  normres=norm(G*xfinal- kron(kron(b,b),b))/nrmrhs;
  fprintf('final full relative residual:  %d\n',normres)
end
```

References

1. Bachmayr, M., Schneider, R., Uschmajew, A.: Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Found. Comput. Math. **16**(6), 1423–1472 (2016)
2. Beckermann, B., Kressner, D., Tobler, Ch.: An error analysis of Galerkin projection methods for linear systems with tensor product structure. SIAM J. Numer. Anal. **51**(6), 3307–3326 (2013)
3. Chen, Z., Lu, L.: A gradient based iterative solutions for Sylvester tensor equations. Math. Probl. Eng. **819479**, 7 (2013)
4. Dahmen, W., DeVore, R., Grasedyck, L., Süli, E.: Tensor-sparsity of solutions to high-dimensional elliptic partial differential equations. Found. Comput. Math. **16**, 813–874 (2016)
5. Dolgov, S.V., Khoromskij, B.N., Oseledets, I.V.: Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker-Planck equation. SIAM J. Sci. Comput. **34**(6), A3016–A3038 (2013)
6. Dolgov, S.V.: TT-GMRES: solution to a linear system in the structured tensor format. Russ. J. Numer. Anal. Math. Model. **28**(2), 149–172 (2013)
7. Gavrilyuk, I., Khoromskij, B.N.: Tensor numerical methods: actual theory and recent applications [editorial]. Comput. Methods Appl. Math. **19**(1), 1–4 (2019)
8. Ghavimi, A.R., Laub, A.J.: Backward Error, sensitivity, and refinement of computed solutions of Algebraic Riccati Equations. Num.Linear Algebra Appl. **2**(1), 29–49 (1995)
9. Golub, G., Van Loan, C.F.: Matrix Computations, 4th edn. The Johns Hopkins University Press, Baltimore (2013)
10. Grasedyck, L.: Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. Computing **72**, 247–265 (2004)
11. Grasedyck, L., Kressner, D., Tobler, Ch.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen **36**(1), 53–78 (2013)
12. Hackbusch, W., Khoromskij, B.N., Tyrtyshnikov, E.E.: Hierarchical Kronecker tensor-product approximations. J. Numer. Math. **13**(2), 119–156 (2005)
13. Huang, B., Xie, Y., Ma, Ch.: Krylov subspace methods to solve a class of tensor equations via the Einstein product. *Num. Linear Algebra Appl.*, 26(e2254), (2019)
14. Khoromskij, B.N.: Tensors-structured numerical methods in scientific computing: survey on recent advances. Chemom. Intell. Lab. Syst. **110**, 1–19 (2012)
15. Khoromskij, Boris N., Schwab, Christoph: Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. SIAM J. Sci. Comput. **33**(1), 364–385 (2011)
16. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
17. Kressner, D., Tobler, C.: Krylov subspace methods for linear systems with tensor product structure. SIAM J. Matrix Anal. Appl. **31**(4), 1688–1714 (2010)
18. Kressner, D., Kumar, R., Nobile, F., Tobler, Ch.: Low-rank tensor approximation for high-order correlation functions of gaussian random fields. SIAM/ASA J. Uncertain. Quantif. **3**, 393–416 (2015)

19. Kressner, D., Pleinger, M., Tobler, Ch.: A preconditioned low-rank cg method for parameter-dependent Lyapunov equations. Num. Linear Algebra Appl. **21**(5), 666–684 (2014)
20. Kressner, D., Tobler, Ch.: Low-rank tensor Krylov subspace methods for parametrized linear systems. SIAM. J. Matrix Anal. Appl. **32**(4), 1288–1316 (2011)
21. Lv, Ch., Ma, Ch.: A modified CG algorithm for solving generalized coupled Sylvester tensor equations. Appl. Math. Comput. **365**, 124699 (2020)
22. The MathWorks, Inc. *MATLAB 7*, r2017b edition, 2017
23. Matthies, H.G., Zander, E.: Solving stochastic systems with low-rank tensor compression. Linear Algebra Appl. **436**, 3819–3838 (2012)
24. Oseledets, I.V.: Tensor-Train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
25. Palitta, D., Simoncini, V.: Matrix-equation-based strategies for convection-diffusion equations. BIT Numer. Math. **56**, 751–776 (2016)
26. Simoncini, V.: Computational methods for linear matrix equations. SIAM Rev. **58**(3), 377–441 (2016)
27. Swarztrauber, P.N.: A direct method for the discrete solution of separable elliptic equations. SIAM J. Numer. Anal. **11**(6), 1136–1150 (1974)
28. Swarztrauber, P.N.: The methods of cyclic reduction, Fourier analysis and the facr algorithm for the discrete solution of Poisson's equation on a rectangle. SIAM Rev. **19**(3), 490–501 (1977)