Multitask kernel-learning parameter prediction method for solving time-dependent linear systems

Kai Jiang\textsuperscript{1,*}, Juan Zhang\textsuperscript{1} and Qi Zhou\textsuperscript{1}

\textsuperscript{1} Key Laboratory of Intelligent Computing and Information Processing of Ministry of Education, Hunan key Laboratory for Computation and Simulation in Science and Engineering, School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan, China, 411105, P.R. China.

Abstract. Matrix splitting iteration methods play a vital role in solving large sparse linear systems. Their performance heavily depends on the splitting parameters, however, the approach of selecting optimal splitting parameters has not been well developed. In this paper, we present a multitask kernel-learning parameter prediction method to automatically obtain relatively optimal splitting parameters, which contains simultaneous multiple parameters prediction and a data-driven kernel learning. For solving time-dependent linear systems, including linear differential systems and linear matrix systems, we give a new matrix splitting Kronecker product method, as well as its convergence analysis and preconditioning strategy. Numerical results illustrate our methods can save an enormous amount of time in selecting the relatively optimal splitting parameters compared with the exists methods. Moreover, our iteration method as a preconditioner can effectively accelerate GMRES. As the dimension of systems increases, all the advantages of our approaches becomes significantly. Especially, for solving the differential Sylvester matrix equation, the speedup ratio can reach tens to hundreds of times when the scale of the system is larger than one hundred thousand.

AMS subject classifications: 62F15, 62J05, 65F08, 65F45, 65M22

Key words: Multitask kernel-learning parameter prediction, Time-dependent linear systems, Matrix splitting Kronecker product method, Convergence analysis, Preconditioning.

1 Introduction

In this paper, we consider the time-dependent linear systems (TDLs) of the form

\[ \dot{x}(t) = \mathcal{L} \circ x(t), \quad t \in [0,T], \]  

(1.1)

*Corresponding author. Email addresses: kaijiang@xtu.edu.cn (K. Jiang), zhangjuan@xtu.edu.cn (J. Zhang), qizhou@smail.xtu.edu.cn (Q. Zhou)
where \( x(t) : [0,T] \to V, \ V \) is \( \mathbb{R}^n \) \((n \in \mathbb{N})\), \( x(0) = x_0 \in V \) is an initial value, and \( \mathcal{L} \) is a linear operator. TDLSs appear in many branches of science and engineering, such as dynamical systems, quantum mechanics, semi-discretization of partial differential equations, differential matrix equations, etc [2, 13, 18, 19, 24, 28]. A series of discrete methods suitable for them have been developed, such as linear multistep schemes, Runge-Kutta methods, general linear methods, block implicit methods, and boundary value methods (BVMs) [7, 8, 10, 16, 17, 22, 31]. After temporal discretization, each TDLS can be transformed into a sparse linear system

\[
Qx = b, \quad Q \in \mathbb{R}^{n \times n} \text{ is nonsingular and } b \in \mathbb{R}^n.
\]

For solving linear systems, matrix splitting iteration methods play an important role as either solvers or preconditioners. The classic matrix splitting forms are all based on \( Q = M - N \), where \( M \) is a nonsingular matrix such that a linear system with the coefficient matrix \( M \) can easily be solved, such as Jacobi, Gauss-Seidel, and successive over-relaxation iteration methods [29]. Alternating direction implicit (ADI) schemes can effectively improve the performance by alternately updating approximate solution. They were initially designed to solve partial differential equations [15, 25], and were gradually extended to more branches, including numerical algebra and optimization [23, 29]. The typical schemes in numerical algebra are Hermitian and skew-Hermitian splitting type methods [3–5, 30]. Further, a general ADI (GADI) framework has recently been developed to put most existing ADI methods into a unified framework [21].

Matrix splitting iteration methods require coefficient matrix \( Q \) into different parts with splitting parameters. The convergence and performance of them are very sensitive to splitting parameters, therefore, choosing the optimal splitting parameters is critical. There have been several approaches to selecting splitting parameters. Experimental traversal method is limited due to an unbearable computational cost, especially for large-scale systems. A more common approach is using theoretical analysis to estimate the bound of splitting parameters in a case-by-case way [3, 11], while its performance heavily depends on the theoretical bound and the scale of systems. Recently, a data-driven approach, Gaussian process regression (GPR) method [21], has presented to predict optimal splitting parameters by choosing an appropriate kernel. GPR method can efficiently predict one splitting parameter at one time.

However, matrix splitting iteration methods can have two or more splitting parameters, among which there are complicated links. Independently predicting each splitting parameter would inevitably affect the predicted accuracy of the original GPR method. Therefore, it requires improving the GRP method to predict multi-parameters simultaneously. Another critical component that determines GPR’s availability is kernel function [33, 35]. The original GPR method chooses kernel function by the problem’s properties [21], which might produce an artificial error. Moreover, the chosen kernel in a kind of problems may be difficult to extend to others. Therefore, automatically learning kernel functions based on distinct problems is still a challenge.
In this work, we propose a new splitting parameter selection method for matrix splitting iteration methods. Concretely, we present multitask kernel learning (MTKL) method for obtaining the optimal splitting parameters which contains simultaneous multiple splitting parameters prediction and a data-driven kernel learning. Furthermore, we propose a new matrix splitting Kronecker product (MSKP) method to solve TDLSs (1.1). Its convergence analysis and preconditioning strategy are also given. We apply our developed methods to some standard TDLSs, including two-dimensional diffusion and convection-diffusion equations, and a differential Sylvester matrix equation. Numerical results illustrate our methods can save an enormous amount of time in selecting the relatively optimal splitting parameters compared with the exists methods. Moreover, our iteration method as a preconditioner can effectively accelerate GMRES. Especially for solving the differential Sylvester matrix equation, the speedup ratio can reach tens to hundreds of times when the scale of the system is larger than one hundred thousand. As the dimension of the systems increases, all the advantages of our approaches becomes significantly.

The rest of this paper is organized as follows. Section 2 proposes the MTKL method. Section 3 presents the MSKP method. Section 4 shows the efficiency and superiority of our proposed methods by numerical experiments. In Section 5, we carry out the summary and give an outlook of future work.

2 MTKL method

In this section, we propose the MTKL method to predict the relatively optimal splitting parameters. MTKL approach contains a multi-task GPR method for multiple splitting parameters and a data-driven kernel learning method.

2.1 Multitask GPR

Matrix splitting iteration methods can have two or more splitting parameters which heavily affect the efficiency. Now we propose a practical multitask GPR method to simultaneously predict these relatively optimal splitting parameters. The multitask method learns M related functions $f_l$ ($l = 1, \cdots, M$) from training data \{($x_{li}, y_{li}$) | $l = 1, \cdots, M, i = 1, \cdots, n, x_{li} \in \mathbb{R}^d, y_{li} \in \mathbb{R}$\}. In practice, we consider the following noised model to avoid singularity:

$$y_{li} = f_l(x_{li}) + \epsilon_l, \quad \epsilon_l \sim \mathcal{N}(0, \sigma_l^2),$$

where $y_{li}$ ($x_{li}$) is the $i$-th output (input) of the $l$-th task, $\epsilon_l$ is the white noise of the $l$-th task.

Let $y^T = [y_{11}, \cdots, y_{1n}, \cdots, y_{M1}, \cdots, y_{Mn}] = \text{vec}(Y^T)$ be the output vector, and $f^T = [f_1, \cdots, f_1, \cdots, f_M, \cdots, f_M] = \text{vec}(F^T)$ be the latent function. The multitask regression problem can be presented as a Gaussian process prior over the latent function

$$f \sim \text{GP}(0, K^l \otimes K^\gamma),$$
where $K_t \in \mathbb{R}^{M \times M}$ and $K_x \in \mathbb{R}^{n \times n}$ are the task and data covariance matrices, respectively [6]. The noised model (2.1) becomes

$$y \sim N(0, K_t \otimes K_x + D \otimes I_n),$$

where $D = \text{diag} [\sigma_1^2, \ldots, \sigma_M^2] \in \mathbb{R}^{M \times M}$. The prediction distribution for the $l$-th task $y_t | y$ on a new point $x_*$ is

$$y_t | y \sim N(\mu_{st}, \Sigma_{st}),$$

where

$$\mu_{st} = (k_t^l \otimes k_{xx, *})^T \Sigma^{-1} y,$$

$$\Sigma_{st} = k_{tt, *}, - (k_t^l \otimes k_{xx, *})^T \Sigma^{-1} (k_t^l \otimes k_{xx, *}),$$

and $\Sigma = K_t \otimes K_x + D \otimes I_n$. $k_t^l$ and $k_x^l$ are the $l$-th diagonal element and the $l$-th column of $K_t$, respectively. $k_{xx, *}$ is a covariances vector between test point $x_*$ and training point $x$, and $K_x$ denotes the covariance matrix of all training points. Hyperparameters $\theta_t$ and $\theta_x$ appear both in the task and data covariance functions. To obtain the hyperparameter estimation, we apply L-BFGS method to maximize marginal likelihood function $L$ in logarithmic form

$$L = \log p(y | x, \theta_x, \theta_t)$$

$$= -\frac{n}{2} \log |K_t| - \frac{M}{2} \log |K_x| - \frac{1}{2} \text{trace} \left[ (K_t)^{-1} F^T (K_x)^{-1} F \right]$$

$$- \frac{n}{2} \sum_{l=1}^M \log \sigma_l^2 - \frac{1}{2} \text{trace} \left[ (Y-F) D^{-1} (Y-F)^T \right] - \frac{M n}{2} \log 2\pi.$$

### 2.2 Kernel learning approach

Choosing an appropriate kernel function determines whether GRP method succeeds or not. The original GRP method directly selects kernel function from the feature of problems [21]. It might result in an artificial error and could be hardly extended to other problems. For example, Figure 3 shows that the GPR method with a manual kernel function will produce a wrong regression curve. How to give an automatic way to choose kernel functions is worth investigating. Here, we present a data-driven method to learn the kernel function. Give a kernel library $K = \{k_{\xi}(x, x') \mid \xi = 1, \ldots, N \}$ that contains basic kernel functions and their multiplicative combinations. For the $l$-th ($l = 1, \ldots, M$) training task, the required kernel function is the linear combination of library elements

$$k(x, x') = \sum_{\xi=1}^N c_{l, \xi} k_{\xi}(x, x').$$
For $N$ training tasks, the weighted matrix is

$$
C = \begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1N} \\
c_{21} & c_{22} & \cdots & c_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
c_{M1} & c_{M2} & \cdots & c_{MN}
\end{bmatrix}.
$$

All weights can be obtained by training the data from concrete TDLS. A similar idea can be also found in pattern discovery \[32, 34\].

**Remark 2.1.** In GPR method \[21\], the training data comes from smaller systems which is a small data set. Learning kernel function by a (deep) neural network from small data is difficult. Therefore, we predetermine the kernel library instead of directly learning kernel function from data.

### 3 Solving TDLSs

In this section, we propose a new matrix splitting iteration method, the MSKP method, for solving the TDLSs (1.1). We also give the convergence analysis and preconditioning strategy.

#### 3.1 Discretization of TDLSs

Here, we use BVMs to discretize TDLS (1.1) in temporal direction to obtain a sparse linear system. We first give a brief description of BVMs. More details can refer to \[8, 9\]. For $i = k_1, \cdots, m - k_2$ ($k = k_1 + k_2$), the $k$-step BVM is

$$
\sum_{j=-k_1}^{k_2} \alpha_{k_1+j} x_{i+j} = \tau \sum_{j=-k_1}^{k_2} \beta_{k_1+j} L x_{i+j},
$$

(3.1)

where $\tau = T/m$ is step size, $x_i \approx x(t_i)$, $t_i = i \tau$, and $\alpha_j, \beta_j, j = -k_1, \cdots, k_2$ are parameters. The extra $k_1 - 1$ initial and $k_2$ final equations are

$$
\sum_{j=0}^{k} \alpha_i^j x_{j} = \tau \sum_{j=0}^{k} \beta_i^j L x_{j}, \quad i = 1, \cdots, k_1 - 1,
$$

$$
\sum_{j=0}^{k} \alpha_i^j x_{m-k+j} = \tau \sum_{j=0}^{k} \beta_i^j L x_{m-k+j}, \quad i = m - k_2 + 1, \cdots, m,
$$

where the coefficients $\alpha_i^j$ and $\beta_i^j$ are chosen such that the truncation errors over all node are consistent. Applying BVM (3.1) leads to the discretization form of TDLS (1.1)

$$
Qx := (A - \tau BL) x = b,
$$

(3.3)
where \( x = (x_0, \ldots, x_m)^T \), \( b = e_1 x_0, e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{m+1} \), the matrices \( A, B \in \mathbb{R}^{(m+1) \times (m+1)} \) have the following structures

\[
A = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
\alpha_0^1 & \alpha_1^1 & \cdots & \alpha_k^1 \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^{k-1} & \alpha_1^{k-1} & \cdots & \alpha_k^{k-1} \\
\alpha_0^k & \alpha_1^k & \cdots & \alpha_k^k \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^{m-k+1} & \alpha_1^{m-k+1} & \cdots & \alpha_k^{m-k+1} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^m & \alpha_1^m & \cdots & \alpha_k^m \\
\end{bmatrix},
\]

\[
B = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
\beta_0^1 & \beta_1^1 & \cdots & \beta_k^1 \\
\vdots & \vdots & \ddots & \vdots \\
\beta_0^{k-1} & \beta_1^{k-1} & \cdots & \beta_k^{k-1} \\
\beta_0^k & \beta_1^k & \cdots & \beta_k^k \\
\vdots & \vdots & \ddots & \vdots \\
\beta_0^{m-k+1} & \beta_1^{m-k+1} & \cdots & \beta_k^{m-k+1} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_0^m & \beta_1^m & \cdots & \beta_k^m \\
\end{bmatrix}.
\]

In this work, we mainly consider linear differential systems and linear matrix systems. If \( L \) is a differential operator on space, TDLS becomes a linear differential system

\[
\begin{cases}
\frac{\partial u}{\partial t} = L u + f, & \text{in } \Omega \times [0, T], \\
u = g(t), & \text{on } \partial \Omega \times [0, T], \\
u(\cdot, 0) = \psi, & \text{in } \Omega,
\end{cases}
\]

where \( \Omega \subset \mathbb{R}^d \) with \( d \geq 1 \) is a bounded and open domain. \( \psi \) is the initial condition and \( f \) is the source term. By proper spatial discretization, such as finite difference and finite element methods, we can obtain an ordinary differential equation

\[
\begin{cases}
M \dot{U}(t) = -K U(t) + F(t), & t \in [0, T], \\
U(0) = \Psi,
\end{cases}
\]
where \( U(t) \in \mathbb{R}^{n_d} \) contains approximate values of \( u(\cdot, t) \) over spatial grid nodes. \( n_d \) is the degree of freedom of spatial discretization. \( F(t) \) and \( \Psi \) are similar notations corresponding to \( f \) and \( \psi \). \( M, K \in \mathbb{R}^{n_d \times n_d} \) are mass and stiff matrices, respectively.

For linear matrix systems, we consider the differential Sylvester matrix equation which has the following form

\[
\begin{aligned}
\dot{X}(t) &= AX(t) + XB + \mathcal{E} \mathcal{F}^T, \quad t \in [0, T], \\
X(t_0) &= X_0,
\end{aligned}
\tag{3.6}
\]

where \( X(t) \in \mathbb{R}^{n \times n} \) for each \( t \in [0, T] \), \( A, B \in \mathbb{R}^{n \times n} \), and \( \mathcal{E}, \mathcal{F} \in \mathbb{R}^{n \times s} \) are full rank matrices with \( s \ll n \). The initial condition is \( X_0 = \tilde{Z}_0 \tilde{Z}_0^T \), \( \tilde{Z}_0, \tilde{Z}_0 \in \mathbb{R}^{n \times s} \). It originate from many specific problems, such as dynamical systems, filter design theory, model reduction problems, differential equations, and robust control problems [1,14]. The equivalent ordinary differential equation of (3.6) is

\[
\begin{aligned}
M \dot{x}(t) &= -K x(t) + f(t), \quad t \in [0, T], \\
x(0) &= \text{vec}(X_0),
\end{aligned}
\tag{3.7}
\]

where \( M \) is an \( n^2 \times n^2 \) identity matrix, \( K = -(I_n \otimes A + B^T \otimes I_n) \), \( x(t) = \text{vec}(X(t)) \), and \( f(t) = \text{vec}(\mathcal{E} \mathcal{F}^T) \).

Applying BVM to (3.5), we obtain a linear system

\[
Q u = (A \otimes M + \tau B \otimes K) u = b, \tag{3.8}
\]

where \( u = [\Psi, U_1, \cdots, U_m]^T, b = \tau (B \otimes I_m) f + e_1 \otimes \Psi, f = [F_0, F_1, \cdots, F_m]^T \). And using the BVM to (3.7), the corresponding linear system can also be obtained.

### 3.2 MSKP method

In this subsection, we propose the MSKP method and also give a fast version by the properties of Kronecker product. Let \( A = (a_{ij}) \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{p \times d} \), the Kronecker product of \( A \) and \( B \) is defined by

\[
A \otimes B = \begin{bmatrix}
a_{11}B & a_{12}B & \cdots & a_{1n}B \\
a_{21}B & a_{22}B & \cdots & a_{2n}B \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1}B & a_{m2}B & \cdots & a_{mn}B
\end{bmatrix}.
\]

Lemma 3.1 gives required properties of Kronecker product. More context about the Kronecker product can refer to [20].

**Lemma 3.1** ([20]). Let \( A, C \in \mathbb{R}^{m \times m}, B, D \in \mathbb{R}^{n \times n} \), then

(I) \( (A + C) \otimes (B + D) = A \otimes B + A \otimes D + C \otimes B + C \otimes D \);
(2) \((A \otimes B)(C \otimes D) = (AC) \otimes (BD)\);
(3) \((A \otimes B)^{-1} = A^{-1} \otimes B^{-1}\);
(4) \((A \otimes B)x = \text{vec}(BX^T)\), vec(X) = x;
(5) \(\sigma(A \otimes B) = \{\lambda \mu | \lambda \in \sigma(A), \mu \in \sigma(B)\}\).

Here, the ‘vec’ operator transforms matrices into vectors by stacking columns

\[ X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n \times m} \iff \text{vec}(X) = [x_1^T, x_2^T, \ldots, x_m^T]^T \in \mathbb{R}^{nm}. \]

Based on the properties of Kronecker product, we give the derivation process of MSKP scheme. First, we introduce generalized Kronecker product splitting (GKPS) method [12]

\[
\begin{align*}
((A + \alpha B) \otimes M)u^{(k+\frac{1}{2})} &= b - (B \otimes (\tau K - \alpha M))u^{(k)}, \\
(B \otimes (\tau K + \alpha M))u^{(k+1)} &= b - ((A + \beta B) \otimes M)u^{(k+\frac{1}{2})}.
\end{align*}
\]  

(3.9a)  (3.9b)

Inspired by GADI framework [21], we add a ‘viscosity’ term to the right side of (3.9b)

\[
\begin{align*}
b - (A - \beta B) \otimes Mu^{(k+\frac{1}{2})} &= \frac{\alpha + \beta}{2} \omega B \otimes M(u^{(k)} - u^{(k+\frac{1}{2})}) \\
&= b - (A - \beta B) \otimes Mu^{(k+\frac{1}{2})} + \alpha + \beta \frac{\omega B \otimes Mu^{(k)} - \alpha + \beta \frac{\omega B \otimes Mu^{(k+\frac{1}{2})}}{2}}{2}
\end{align*}
\]

\[
\begin{align*}
b &= b - (A + \alpha B) \otimes Mu^{(k+\frac{1}{2})} + \alpha + \beta \frac{\omega B \otimes Mu^{(k)} - \alpha + \beta \frac{\omega B \otimes Mu^{(k+\frac{1}{2})}}{2}}{2}
\end{align*}
\]

Then we obtain the MSKP scheme

\[
\begin{align*}
(A + \alpha B) \otimes Mu^{(k+\frac{1}{2})} &= b - \tau K - \alpha Mu^{(k)}, \\
B \otimes (\tau K + \beta M)u^{(k+1)} &= B \otimes (\tau K + \beta M)u^{(k)} + \frac{\alpha + \beta}{2} \omega B \otimes Mu^{(k+\frac{1}{2})},
\end{align*}
\]

(3.10)

where \(k = 0, 1, \ldots\), the splitting parameters \(\alpha, \beta > 0\) and \(0 \leq \omega < 2\). Note that, when \(\omega = 0\), the MSKP method reduces to the GKPS scheme [12], when \(\omega = 0\) and \(\alpha = \beta\), the MSKP method becomes the Kronecker product splitting (KPS) approach [11].

Eliminating the intermediate vector \(u^{(k+\frac{1}{2})}\), we can rewrite the MSKP method in a fixed point form

\[
u^{(k+1)} = T(\alpha, \beta, \omega)u^{(k)} + P(\alpha, \beta, \omega)^{-1}b,
\]
where the iteration matrix is
\[
T(\alpha, \beta, \omega) = \left[ \left( A + \alpha B \right) \otimes (\tau K + \beta M) \right]^{-1} \\
\left( (A - \beta B) \otimes (\tau K - \alpha M) + (A + \alpha B) \otimes \frac{\alpha + \beta}{2} \omega M + \frac{\alpha + \beta}{2} \omega B \otimes (\tau K - \alpha M) \right),
\]
and
\[
P(\alpha, \beta, \omega) = \frac{2}{(\alpha + \beta)(2 - \omega)} \left(A + \alpha B \right) \otimes (\tau K + \beta M).
\]
Obviously, there is a unique splitting
\[
Q = P(\alpha, \beta, \omega) - R(\alpha, \beta, \omega),
\]
where
\[
R(\alpha, \beta, \omega) = \frac{2}{(\alpha + \beta)(2 - \omega)} \left( (A - \beta B) \otimes (\tau K - \alpha M) + (A + \alpha B) \otimes \frac{\alpha + \beta}{2} \omega M + \frac{\alpha + \beta}{2} \omega B \otimes (\tau K - \alpha M) \right).
\]
Note that
\[
T(\alpha, \beta, \omega) = P(\alpha, \beta, \omega)^{-1} R(\alpha, \beta, \omega) = I - P(\alpha, \beta, \omega)^{-1} Q.
\]
Therefore, the MSKP method (3.10) can become
\[
u^{(k+1)} = \nu^{(k)} + P(\alpha, \beta, \omega)^{-1} r^{(k)}, \quad r^{(k)} = b - Q\nu^{(k)}.
\]
Using the properties of Kronecker product, we can obtain a fast implementation of MSKP method (see Algorithm 1).

**Remark 3.1.** Here we give a computational complexity analysis on MSKP method. For each iteration, directly solving will result in \(O((n^d m)^3)\), while Algorithm 1 only needs \(O((n^d m)^2 + n^d m^2 + n^d m)\).

### 3.3 Convergence analysis of MSKP method

In this subsection, we analyze the convergence of MSKP method. Let \(\sigma(A)\) and \(\rho(A)\) be the spectral set and the spectral radius of \(A\), respectively.

**Theorem 1.** Assume that all eigenvalues of \(B^{-1} A\) and \(M^{-1} K\) have positive and nonnegative real parts, respectively. Then for any \(\alpha > 0, 0 < \beta \leq \tau \min_{\xi \in \sigma(M^{-1} K)} \text{Re}(\xi)\) and \(\omega \in [0, 2)\), the MSKP method (3.10) converges to the unique solution \(u^*\) of the linear system (3.8). The spectral radius \(\rho(T(\alpha, \beta, \omega))\) of iteration matrix \(T(\alpha, \beta, \omega)\) satisfies
\[
\rho(T(\alpha, \beta, \omega)) \leq \frac{1}{2} [(2 - \omega) \varphi(\alpha, \beta) + \omega] < 1,
\]
where
\[
\varphi(\alpha, \beta) = \max_{\lambda \in \sigma(B^{-1}A + \frac{\omega}{2} I_n)} \left| \frac{\lambda - \frac{\alpha + \beta}{2}}{\lambda + \frac{\alpha + \beta}{2}} \right|.
\]
Algorithm 1: MSKP method for solving linear system (3.8).

Data: $u^{(0)} \in \mathbb{R}^{n \times m}$, $\alpha, \beta > 0$, $0 \leq \omega < 2$

1. Calculate $\tau K + \beta M$ and $(A + \alpha B)^{-T}$

2. for $k = 0, 1, 2, \cdots$ until $\{u^{(k)}\}$ converges do

3. $r^{(k)} = \frac{(\alpha + \beta)(2 - \omega)}{2} (b - Qu^{(k)})$

4. $\text{vec}(\{r_1^{(k)}, r_2^{(k)}, \cdots, r_m^{(k)}\}) = r^{(k)}$

5. for $i = 1, 2, \cdots, m$ do

6. $(\tau K + \beta M)v_i^{(k)} \approx r_i^{(k)}$ (use GMRES)

7. end

8. $v^{(k)} = \text{vec} \left( \{v_1^{(k)}, v_2^{(k)}, \cdots, v_m^{(k)}\} \cdot (A + \alpha B)^{-T} \right)$

9. $u^{(k+1)} = u^{(k)} + v^{(k)}$

end

Proof. Let $\hat{\lambda}$ and $\hat{\mu}$ be the eigenvalues of matrices $B^{-1}A$ and $\tau M^{-1}K$, respectively. By the property of Kronecker product, the eigenvalues of iteration matrix $T(\alpha, \beta, \omega)$ (3.11) have the following form

$$p = \frac{(\hat{\lambda} - \beta)(\hat{\mu} - \alpha) + (\alpha + \beta)\omega(\hat{\lambda} + \alpha)/2 + \alpha + \beta \omega(\hat{\mu} - \alpha)/2}{(\hat{\lambda} + \alpha)(\hat{\mu} + \beta)}.$$

Let $\alpha = \hat{\alpha} + \hat{\beta}$ and $\beta = \hat{\alpha} - \hat{\beta}$, the above equation becomes

$$p = \frac{(\hat{\lambda} - \hat{\alpha} + \hat{\beta})(\hat{\mu} - \hat{\alpha} - \hat{\beta}) + \hat{\alpha} \omega(\hat{\lambda} + \hat{\alpha} + \hat{\beta}) + \hat{\alpha} \omega(\hat{\mu} - \hat{\alpha} - \hat{\beta})}{(\hat{\lambda} + \hat{\alpha} + \hat{\beta})(\hat{\mu} + \hat{\alpha} - \hat{\beta})},$$

which is equivalent to

$$p = \frac{(\lambda - \hat{\alpha})(\mu - \hat{\alpha}) + \hat{\alpha} \omega(\lambda + \hat{\alpha}) + \hat{\alpha} \omega(\mu - \hat{\alpha})}{(\lambda + \hat{\alpha})(\mu + \hat{\alpha})} = \frac{(\lambda - \hat{\alpha})(\mu - \hat{\alpha}) + \hat{\alpha} \omega(\lambda + \mu)}{(\lambda + \hat{\alpha})(\mu + \hat{\alpha})},$$

where $\lambda \in \sigma(B^{-1}A + \hat{\beta}I_m)$, $\mu \in \sigma(\tau M^{-1}K - \hat{\beta}I_m)$.

Note that

$$2p = 2\frac{\lambda - \hat{\alpha}}{\lambda + \hat{\alpha}} \frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}} + \omega \frac{2\hat{\alpha}(\lambda + \mu)}{(\lambda + \hat{\alpha})(\mu + \hat{\alpha})} = (2 - \omega) \frac{\lambda - \hat{\alpha}}{\lambda + \hat{\alpha}} \frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}} + \omega \frac{\lambda + \hat{\alpha}}{\lambda + \hat{\alpha}} \frac{\mu + \hat{\alpha}}{\mu + \hat{\alpha}} = (2 - \omega) \frac{\lambda - \hat{\alpha}}{\lambda + \hat{\alpha}} \frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}} + \omega.$$

Denote $q = \frac{\lambda - \hat{\alpha}}{\lambda + \hat{\alpha}} \frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}}$

$$p = \frac{1}{2} [(2 - \omega)q + \omega].$$
Let \( q = a + bi \) (\( i = \sqrt{-1} \)),

\[
|p| = \frac{1}{2} |(2 - \omega)(a + bi) + \omega| \\
= \frac{1}{2} \sqrt{2(2 - \omega)^2 a^2 + \omega^2 + 2(2 - \omega)\alpha \omega + (2 - \omega)^2 b^2} \\
\leq \frac{1}{2} \sqrt{2(2 - \omega)^2 (a^2 + b^2) + \omega^2 + 2(2 - \omega)\omega \sqrt{a^2 + b^2}} \\
= \frac{1}{2} \sqrt{(2 - \omega)^2 |q|^2 + \omega^2 + 2(2 - \omega)\omega |q|} \\
= \frac{1}{2} [(2 - \omega)|q| + \omega].
\]

If \( 0 < \beta \leq \tau_{\min} \xi \in \sigma(M^{-1}K) \text{Re}(\xi) \) and \( \alpha > 0 \), then

\[
\frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}} \leq 1, \quad \mu \in \sigma(\tau M^{-1}K - \hat{\beta}I_m).
\]

When \( 0 \leq \omega < 2 \), \( \text{Re}(\lambda) > 0 \),

\[
\rho(T(\alpha, \beta, \omega)) = \max_{\lambda \in \sigma(B^{-1}A + \beta I_m)} |p| \leq \max_{\mu \in \sigma(\tau M^{-1}K - \hat{\beta}I_m)} \frac{1}{2} [(2 - \omega)|q| + \omega] \\
\leq \frac{1}{2} \left(2 - \omega\right) \max_{\lambda \in \sigma(B^{-1}A + \beta I_m)} \left| \frac{\lambda - \hat{\alpha}}{\lambda + \hat{\alpha}} + \omega \right| \\
= \frac{1}{2} \left(2 - \omega\right) \max_{\lambda \in \sigma(B^{-1}A + \frac{\mu + \lambda}{\mu - \lambda} \beta I_m)} \left| \frac{\lambda - \left( \alpha + \beta \right) / 2}{\lambda + \left( \alpha + \beta \right) / 2} + \omega \right| \\
= \frac{1}{2} [(2 - \omega) \phi(\alpha, \beta) + \omega] < 1.
\]

It implies that the MSKP method (3.10) converges to the unique solution \( u^* \) of linear system (3.8).

### 3.4 Accelerating GMRES by MSKP preconditioner

Based on MSKP method, we give a preconditioning strategy to GMRES. From (3.13), the linear system (3.8) is equivalent to

\[
(I - T(\alpha, \beta, \omega)) u = P(\alpha, \beta, \omega)^{-1} Q u = c,
\]

where \( c = P(\alpha, \beta, \omega)^{-1} b \) and \( P(\alpha, \beta, \omega) \) is a preconditioner. This equivalent system can be solved by GMRES [27]. Algorithm 2 gives the solving process for (3.8) by putting \( P(\alpha, \beta, \omega) \) as a preconditioner in GMRES. Note that using the MSKP preconditioner within
Algorithm 2: GMRES-MSKP method for solving linear system (3.8).

Data: $u^{(0)} \in \mathbb{R}^{n^m}, \alpha, \beta > 0, 0 \leq \omega < 2$

1. Calculate preconditioner $P(\alpha, \beta, \omega)$ according to (3.12)

   2. for $k = 1, 2, \cdots$ until $\{u_k\}$ converges do

      3. Calculate $r_0 = P(\alpha, \beta, \omega)^{-1}(b - Qu_0)$ and $v_1 = r_0 / \|r_0\|_2$

      4. for $j = 1, 2, \cdots, k$ do

         5. $\bar{w}_j = Qv_j$

         6. $w_j = P(\alpha, \omega)^{-1}\bar{w}_j$

         7. for $i = 1, 2, \cdots, j$ do

            8. $h_{ij} = (w_j, v_i)$

         end

         9. $\hat{\omega}_{j+1} = w_j - \sum_{i=1}^{j} h_{ij} / v_i$

         10. $h_{j+1,j} = \|\hat{\omega}_{j+1}\|_2$

         11. $v_{j+1} = \hat{\omega}_{j+1} / h_{j+1,j}$

      end

   12. $u_k = u_0 + V_ky_k$, where $y_k$ is the solution of $\min_y \|r_0\|_2e_1 - H_ky\|_2$

GMRES requires solving a linear system $P(\alpha, \beta, \omega)v = r$ at each iteration, which can be economically solved by Algorithm 1.

Assume that all the eigenvalues of $B^{-1}A$ and $M^{-1}K$ have positive and nonnegative real parts, respectively. $\alpha > 0, 0 < \beta \leq \tau \min_{\xi \in \sigma(M^{-1}K)} \Re(\xi)$ and $0 \leq \omega < 2$. Since $\rho(P(\alpha, \beta, \omega)^{-1}Q) = \rho(I - T(\alpha, \beta, \omega)) \leq 1$, we can conclude that all the eigenvalues of the preconditioned matrix $P(\alpha, \beta, \omega)^{-1}Q$ are located in a circle with radius 1. Compared with KPS and GKPS preconditioners (see Figure 1), MSKP preconditioner can further reduce the condition number of coefficient matrix $Q$, and the eigenvalue distribution of preconditioned system has a more tighter bound.

Figure 1: The eigenvalue distribution of $Q$ with different preconditioners. $Q$ is the coefficient matrix generated by discretizing 2D diffusion equation with $h = \tau = 1/16$ (see subsection 4.1 for details).
4 Numerical experiments

In this section, we present three numerical examples, a 2D diffusion equation, a 2D convection-diffusion equation, and a differential Sylvester matrix equation, to show the performance of MSKP and GMRES with MSKP preconditioner (GMRES-MSKP) methods. As a comparison, we also give experimental results of KPS, GKPS, GMRES, and GMRES-GKPS methods. All computations are carried out using PyCharm 2020.2 on a Mac laptop with 2.3 GHz Quad Intel Core i5. All tests are started with zero vector. All iteration methods are terminated if the relative residual error satisfies $\text{RES} = \frac{\|r^{(k)}\|_2}{\|r^{(0)}\|_2} \leq 10^{-6}$, where $r^{(k)} = b - Q u^{(k)}$ is the $k$-step residual. “IT” and “CPU” denote the required iterations and the CPU time (in seconds), respectively. In the following calculations, the relatively optimal parameters of KPS, GKPS, and GMRES-GKPS methods are obtained by traversing approach, and the ones of MSKP and GMRES-MSKP methods are obtained by MTKL method.

We use the MTKL method with the kernel library $\mathcal{K}$ contains linear, Gaussian, periodic kernels, and their multiplicative combinations. Concretely, three basic kernel functions are

- **Linear kernel:** $k_l(x,x') = \sigma_f^2(x-c)(x'-c)$,
- **Gaussian kernel:** $k_g(x,x') = \sigma_f^2 \exp\left(-\frac{\|x-x'\|^2}{2\iota^2}\right)$,
- **Periodic kernel:** $k_p(x,x') = \sigma_f^2 \exp\left(-\frac{2}{\pi^2} \sin^2\left(\pi \frac{x-x'}{p}\right)\right)$.

The output variance $\sigma_f^2$ determines the average distance of the kernel function away from its mean. The offset $c$ determines the $x$-coordinate of a point, at which the kernel function has zero variance. The lengthscale $\iota$ determines the length of the ‘wiggles’. The period $p$ measures the distance between repetitions of the kernel function. The multiplicative combination kernels are $k_{ll} = k_l k_l$, $k_{lg} = k_l k_g$, $k_{lp} = k_l k_p$, and $k_{gp} = k_g k_p$. Consequently, the kernel library is $\mathcal{K} = \{k_l, k_g, k_p, k_{ll}, k_{lg}, k_{lp}, k_{gp}\}$.

4.1 2D diffusion equation

We consider a 2D diffusion equation

$$u_t = u_{xx} + u_{yy} + f, \quad (x,y) \in [0,1]^2, \quad t \in [0,1],$$

with homogeneous Dirichlet boundary condition. The exact solution is

$$u(x,y,t) = \sin(5.25\pi t) xy(1-x)(1-y)$$

and $f$ is correspondingly determined. First, using centered difference scheme on an $n \times n$ uniform grid with mesh size $h = 1/(n+1)$ in each unit square, we obtain an $n^2$-dimensional ODE (3.5), in which $K = (I_n \otimes T_n + T_n \otimes I_n)/h^2$ and $M$ is an $n^2 \times n^2$ identity
matrix. \( T_n = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{n \times n} \). Further, we discretize this ODE by the fifth-order generalized Adams method (GAM-5) in BVM framework [8] using uniform time grid on \([0, 1]\) with time step size \( \tau = 1/(m-1) \). Then we obtain the full discretize linear system as (3.8), in which the coefficient matrices are

\[
A = \begin{bmatrix}
1 & -1 & 1 & & & \\
-1 & 1 & -1 & 1 & & \\
& & \ddots & \ddots & \ddots & \\
& & & -1 & 1 & -1 \\
& & & & -1 & 1 \\
& & & & & -1 & 1
\end{bmatrix},
\]

\[
B = \frac{1}{720} \begin{bmatrix}
0 & 251 & 646 & -264 & 106 & -19 \\
-19 & 346 & 456 & -74 & 11 & \ddots \\
& \ddots & \ddots & \ddots & \ddots & \ddots \\
& & -19 & 346 & 456 & -74 & 11 \\
& & 11 & -74 & 456 & 346 & -19 \\
& & -19 & -264 & 646 & 251
\end{bmatrix}.
\]

### 4.1.1 Predicting relatively optimal parameters

We use MTKL approach to predict the splitting parameters \((\alpha, \beta, \omega)\) of MSKP method. Table 1 gives the training, test, and retrained data sets of \((\alpha, \beta, \omega)\). Concretely, \((\alpha, \beta, \omega)\) in the training data set is produced by traversing interval \([0,5]\) with a step size of 0.01. \( \alpha \) and \( \beta \) are obtained by traversing interval \([0,2]\) with the same step size. For the test data set, let \( m \) from 1 to 500 with \( \Delta m = 1 \). To predict the parameter more accurately and improve the generation ability, we put the predicted data into the training set to form the retrained data set. In this experiment for the retrained data set, we let \( m \) from 128 to 500 with \( \Delta m = 30 \).

**Table 1**: Training, test, and retrained data sets \((\alpha, \beta, \omega)\) in MTKL approach for solving the 2D diffusion equation (4.1) with MSKP method \((n = 16)\).

| Training set | Test set | Retrained set |
|--------------|----------|---------------|
| \( m: 10 \sim 32, \ \Delta m = 2 \) | \( m: 1 \sim 500, \ \Delta m = 1 \) | \( m: 128 \sim 500, \ \Delta m = 30 \) |
| \( m: 36 \sim 80, \ \Delta m = 4 \) | \( m: 1 \sim 500, \ \Delta m = 1 \) | \( m: 128 \sim 500, \ \Delta m = 30 \) |
| \( m: 88 \sim 128, \ \Delta m = 8 \) | \( m: 1 \sim 500, \ \Delta m = 1 \) | \( m: 128 \sim 500, \ \Delta m = 30 \) |

In this case, the MTKL method considers a linear combination of \( k_g, k_p, k_{gp} \) in kernel
Figure 2: Regression curves of \((\alpha, \beta, \omega)\) against \(m\) (fixed \(n = 16\)) when solving the 2D diffusion equation (4.1) with MSKP method. Up: no retraining; Bottom: retraining.

Figure 2 shows the relative optimal parameter regression curve of \((\alpha, \beta, \omega)\) against \(m\), and the retrained data set can shrink the confidence interval. The optimized hyperparameters are

\[
C = \begin{bmatrix}
1.07 & 0.02 & 0.13 \\
0.93 & 0.01 & 0.06 \\
0.31 & 0.66 & 0.47
\end{bmatrix}, \quad K_t = \begin{bmatrix}
15.91 & 0.00 & 0.00 \\
0.00 & 1.07 & 0.01 \\
0.00 & 0.01 & 1.22
\end{bmatrix},
\]

and \(t = 139.60, \ p = 8.03\). Note that, three kernels \(k_g(x, x'), k_p(x, x'),\) and \(k_{gp}(x, x')\) in the combination all play important roles. For comparison, Figure 3 demonstrates that when only periodic kernel is considered, the GRP method [21] fails to predict accurate splitting parameter \(\omega\).

4.1.2 Comparison

In this subsection, we compare our methods with KPS, GKPS, GMRES, and GMRES-GKPS methods for solving the 2D diffusion equation (4.1). Table 2 compares the numerical results with different discrete resolution. MSKP method can achieve a convergence efficiency two to three times faster than KPS method, and is superior to GKPS method in terms of iteration steps and computational cost. Additionally, MSKP method as a pre-conditioner can better accelerate GMRES methods compared with GKPS method. As the dimension of system increases, the advantages of our approaches becomes significantly.
Figure 3: Regression curve of $\omega$ against $m$ (fixed $n = 16$) when solving the 2D diffusion equation (4.1) with MSKP method. Here we use GRP method with only periodic kernel.

Table 3 shows the traversal CPU time of different methods when $h = \tau = 1/64$. Due to the MTKL approach, MSKP and GMRES-MSKP methods do not consume traversal time in selecting relatively optimal parameters. However, the traversal time of KPS, GKPS, and GMRES-GKPS methods increases dramatically as the dimension of system becomes larger, resulting in an unaffordable computational cost.

Table 2: Results of solving the 2D diffusion equation (4.1).

| Method   | $\tau$ | $h$ | KPS-IT | KPS-CPU | GKPS-IT | GKPS-CPU | MSKP-IT | MSKP-CPU | GMRES-IT | GMRES-CPU | GMRES-GKPS-IT | GMRES-GKPS-CPU | GMRES-MSKP-IT | GMRES-MSKP-CPU |
|----------|--------|-----|--------|---------|---------|----------|---------|---------|----------|----------|----------------|----------------|----------------|----------------|
|          |        |     |    1/16 |    1/32 |    1/64 |    1/16 |    1/32 |    1/64 |    1/16 |    1/32 |    1/64 |                |                |                |                |
|          |        |     |        |         |         |          |         |         |          |         |                |                |                |                |
| KPS      |        |     | 33     | 36      | 37      | 43       | 51      | 54      | 45       | 74       | 86               |                |                |                |
|          |        |     | 0.67   | 1.58    | 5.56    | 1.28     | 3.93    | 14.56   | 1.94     | 8.20     | 37.79           |                |                |                |
| GKPS     |        |     | 19     | 19      | 19      | 22       | 22      | 22      | 36       | 36       | 36               |                |                |                |
|          |        |     | 0.38   | 0.83    | 2.89    | 0.64     | 1.71    | 5.94    | 1.55     | 4.02     | 15.98           |                |                |                |
| MSKP     |        |     | 15     | 15      | 15      | 18       | 19      | 20      | 23       | 26       | 27               |                |                |                |
|          |        |     | 0.29   | 0.66    | 2.25    | 0.52     | 1.47    | 5.40    | 0.99     | 2.86     | 11.91           |                |                |                |
| GMRES    |        |     | 138    | 336     | 813     | 167      | 403     | 966     | 214      | 511      | 1227            |                |                |                |
|          |        |     | 0.32   | 2.55    | 40.09   | 0.51     | 11.81   | 152.48  | 1.18     | 30.31    | 611.77          |                |                |                |
| GMRES-GKPS |      |     | 12     | 12      | 12      | 19       | 19      | 19      | 33       | 33       | 33               |                |                |                |
|          |        |     | 0.18   | 0.42    | 2.31    | 0.57     | 1.37    | 7.26    | 2.15     | 4.29     | 25.03           |                |                |                |
| GMRES-MSKP |     |     | 11     | 11      | 12      | 16       | 17      | 18      | 21       | 24       | 25               |                |                |                |
|          |        |     | 0.16   | 0.39    | 2.28    | 0.48     | 1.22    | 6.83    | 1.37     | 3.12     | 18.94           |                |                |                |

Table 3: Traversal CPU in solving the 2D diffusion equation (4.1) when $h = \tau = 1/64$. 

| Method   | KPS     | GKPS   | MSKP   | GMRES  | GMRES-GKPS | GMRES-MSKP |
|----------|---------|--------|--------|--------|------------|------------|
| Traversal CPU | 18025.83 | 5848.68 | 0      | 0      | 9160.98    | 0          |
4.2 2D convection-diffusion equation

The second example is a 2D convection diffusion equation

\[ u_t + u_x = u_{xx} + u_{yy} + f, \quad (x,y) \in [0,1]^2, \quad t \in [0,1], \]  

with homogeneous Dirichlet boundary condition. \( u \in \mathbb{R}^{mn^2} \) is the unknown vector of discretizing \( u(x,y,t) \). \( f \in \mathbb{R}^{mn^2} \) is the vector of discretizing \( f(x,y,t) \), which determined by choosing the exact solution \( u_e = (1,1,\cdots,1)^T \). Using centered difference scheme on an \( n \times n \) uniform grid with mesh size \( h = 1/(n+1) \) in each unit square, we obtain an \( n^2 \)-dimensional ODE (3.5), in which \( K = I_n \otimes P_n + Q_n \otimes I_n \) and \( M \) is an \( n^2 \times n^2 \) identity matrix.

\[ P_n = \text{tridiag}\left(-1,2,-1\right)/h^2 \text{ and } Q_n = \text{tridiag}\left(-1/2h-1/h^2,2/h^2,1/2h-1/h^2\right) \in \mathbb{R}^{n \times n}. \]

Then we apply GAM-5 on uniform time grid with a time step size \( \tau = 1/(m-1) \) to discrete this ODE. Finally, we obtain a linear system of the form as (3.8).

4.2.1 Predicting relatively optimal parameters

We use MTKL approach to predict the splitting parameter \((\alpha, \beta, \omega)\) of MSKP method. Table 4 gives the training, test, and retrained data sets of \((\alpha, \beta, \omega)\). Again, \((\alpha, \beta, \omega)\) in the training data set is generated by traversing method, \( m \) from 10 to 128 with different step size \( \Delta m \). The traversing interval and step size of \( \alpha, \beta, \) and \( \omega \) are the same as the first example. For the test data set, let \( m \) from 1 to 500 with \( \Delta m = 1 \). And for the retrained data set, let \( m \) from 128 to 500 with \( \Delta m = 30 \).

Table 4: Training, test, and retrained data sets \((\alpha, \beta, \omega)\) in MTKL approach for solving the 2D convection-diffusion equation (4.2) with MSKP method \((n = 16)\).

| Training set                  | Test set              | Retrained set             |
|-------------------------------|-----------------------|---------------------------|
| \( m: 10 \sim 32, \Delta m = 2 \) | \( m: 1 \sim 500, \Delta m = 1 \) | \( m: 128 \sim 500, \Delta m = 30 \) |
| \( m: 36 \sim 80, \Delta m = 4 \) |                       |                           |
| \( m: 88 \sim 128, \Delta m = 8 \) |                       |                           |

In this case, the MTKL method uses a linear combination of \( k_g, k_p, k_{gp} \) to form the kernel library \( \mathcal{K} \). Concretely, for the \( l \)-th task,

\[ k_l(x,x') = c_{l1}k_g(x,x') + c_{l2}k_p(x,x') + c_{l3}k_{gp}(x,x'), \quad l = 1,2,3. \]

Figure 4 shows the relative optimal parameter regression curve of \((\alpha, \beta, \omega)\) against \( m \). The optimized hyperparameters are

\[ C = \begin{bmatrix} 1.21 & 0.05 & 0.04 \\ 1.17 & 0.03 & 0.02 \\ 0.03 & 0.34 & 0.67 \end{bmatrix}, \quad K' = \begin{bmatrix} 0.21 & 0.41 & 8.38 \\ 0.41 & 0.33 & 7.65 \\ 8.38 & 7.65 & 1.01 \end{bmatrix}, \]

and \( t = 99.54, \quad p = 7.83 \). From Figure 4, it can be seen that using the retrained data set can shrink the confidence interval. This improves the prediction accuracy and strengthens the generalization ability of the regression model.
Figure 4: Regression curves of \((\alpha, \beta, \omega)\) against \(m\) (fixed \(n = 16\)) when solving 2D convection-diffusion equation (4.2) with MSKP method. Up: no retraining; Bottom: retraining.

4.2.2 Comparison

In this subsection, we compare our methods with KPS, GKPS, GMRES, and GMRES-GKPS methods for solving the 2D convection-diffusion equation (4.2). Table 5 shows MSKP method has a better convergence performance than KPS and GKPS methods. In this experiment, \(\beta\) in GKPS does not affect the performance, while \(\omega\) in MSKP still has a great influence. Moreover, MSKP method as a preconditioner can accelerate GMRES method by tens of times. As Table 6 presents, MTKL method demonstrates an immense advantage in predicting the relatively optimal splitting parameters with large-scale systems.

4.3 Differential Sylvester matrix equation

Finally, we consider a differential Sylvester matrix equation (3.6), in which \(A\) and \(B\) obtained from the centered finite difference discretization of the operator \(L(u) = \Delta u + f_1(x,y)\frac{\partial u}{\partial x} + f_2(x,y)\frac{\partial u}{\partial y} + f(x,y)u\) on the unit square \([0,1]^2\) with homogeneous Dirichlet boundary condition. The number of inner grid points in each direction is \(n_0\), then the dimension of \(A\) and \(B\) is \(n = n_0^2\). For this experiment, we extract the \(A\) and \(B\) from the Lyapack package [26] using the command `fdm_2d_matrix(n_0, x, y, 0)`. \(E, F \in \mathbb{R}^{n \times 2}\) are obtained by the random values uniformly distributed on \([0,1]\). The initial condition is \(X_0 = 0\). We discrete the differential Sylvester matrix equation (3.6) in temporal direction by GAM-5 with uniform time grid on \([0,1]\) (time step size \(\tau = 0.1\)), and obtain a similar linear system of the
Table 5: Results of solving the 2D convection-diffusion equation (4.2).

| Method         | τ  | h  | IT  | CPU   | IT  | CPU   | IT  | CPU   |
|----------------|----|----|-----|-------|-----|-------|-----|-------|
| KPS / GKPS     | 1/16 | 1/16 | 58  | 0.87  | 101 | 2.56  | 178 | 11.21 |
|                | 1/32 | 1/32 | 59  | 1.54  | 98  | 4.95  | 169 | 22.69 |
|                | 1/64 | 1/64 | 56  | 2.69  | 103 | 10.01 | 168 | 46.74 |
| MSKP           | 1/16 | 1/16 | 43  | 0.66  | 68  | 1.73  | 108 | 6.81  |
|                | 1/32 | 1/32 | 44  | 1.15  | 70  | 3.48  | 111 | 14.87 |
|                | 1/64 | 1/64 | 45  | 2.16  | 73  | 7.16  | 115 | 31.93 |
| GMRES          | 1/16 | 1/16 | 150 | 0.41  | 333 | 3.22  | 731 | 33.66 |
|                | 1/32 | 1/32 | 158 | 0.59  | 350 | 6.80  | 764 | 82.85 |
|                | 1/64 | 1/64 | 197 | 1.26  | 433 | 11.43 | 944 | 216.34 |
| GMRES-GKPS     | 1/16 | 1/16 | 25  | 0.35  | 25  | 0.67  | 26  | 3.04  |
|                | 1/32 | 1/32 | 30  | 0.84  | 34  | 2.18  | 45  | 12.44 |
|                | 1/64 | 1/64 | 46  | 2.69  | 51  | 6.58  | 60  | 30.90 |
| GMRES-MSKP     | 1/16 | 1/16 | 22  | 0.31  | 24  | 0.64  | 24  | 2.76  |
|                | 1/32 | 1/32 | 27  | 0.75  | 31  | 1.98  | 33  | 9.18  |
|                | 1/64 | 1/64 | 32  | 1.87  | 37  | 4.77  | 40  | 21.08 |

Table 6: Traversal CPU in solving the 2D convection-diffusion equation (4.2) when \( h = \tau = 1/64 \).

| Method         | KPS/GKPS | MSKP | GMRES | GMRES-GKPS | GMRES-MSKP |
|----------------|----------|------|-------|------------|------------|
| Traversal CPU  | 2327/6,52 | 0    | 0     | 12324.91   | 0          |

form as (3.8).

4.3.1 Predicting relatively optimal parameters

We use MTKL method to predict the relatively optimal splitting parameters \((\alpha, \beta, \omega)\) of MSKP scheme. Table 7 gives the training, test, and retrained data sets of \((\alpha, \beta, \omega)\) in the training data set is obtained by a traversing way for small scale linear systems, \( n \) from 16 to 225 with different step size. For the test data set, let \( n \) from 1 to 1200 with \( \Delta n = 1 \). And for the retrained data set, let \( n \) from 250 to 1200 with \( \Delta n = 50 \).

Table 7: Training, test, and retrained data sets \((\alpha, \beta, \omega)\) in MTKL approach for solving the differential Sylvester matrix equation (3.6) with MSKP method (\( \tau = 0.1 \)).

| Training set     | Test set | Retrained set |
|------------------|----------|---------------|
| \( n = \gamma \) | \( n : 1 \sim 1200, \Delta n = 1 \) | \( n : 250 \sim 1200, \Delta n = 50 \) |

In this case, the kernel library \( \mathcal{K} \) in MTKL method is a linear combination of \( k_g, k_l, k_{gl} \).

For the \( l \)-th task,

\[
k_l(x, x') = c_{1l}k_g(x, x') + c_{2l}k_l(x, x') + c_{3l}k_{gl}(x, x'), \quad l = 1, 2, 3.
\]

Figure 5 shows the relative optimal parameter regression curve of \((\alpha, \beta, \omega)\) against \( m \). The optimized hyperparameters are

\[
C = \begin{bmatrix} 0.89 & 0.05 & 0.24 \\ 0.99 & 0.11 & 0.07 \\ 0.32 & 0.26 & 0.79 \end{bmatrix}, \quad K^t = \begin{bmatrix} 1.23 & 0.03 & 0.01 \\ 0.03 & 1.14 & 0.12 \\ 0.01 & 0.12 & 0.97 \end{bmatrix}, \quad t = 1.15.
\]
4.3.2 Comparison

We compare our proposed methods with KPS, GKPS, and GMRES approaches for solving the differential Sylvester matrix equation (3.6). Table 8 shows corresponding numerical results, which present that our approach has a significant advantage for solving linear matrix systems. When the scale of system is larger than one hundred thousand, MSKP method as a preconditioner can speed up tens to hundreds times over GMRES.

Table 8: Results of solving the differential Sylvester matrix equation (3.6) with \( \tau = 0.1 \).

| \( n^2 \) | KPS IT | KPS CPU | GKPS IT | GKPS CPU | MSKP IT | MSKP CPU | GMRES IT | GMRES CPU |
|---|---|---|---|---|---|---|---|---|
| 256 | 19 | 0.29 | 9 | 0.14 | 4 | 0.06 | 70 | 0.11 |
| 1296 | 27 | 0.62 | 9 | 0.21 | 4 | 0.09 | 114 | 0.56 |
| 4096 | 40 | 2.47 | 9 | 0.53 | 4 | 0.23 | 160 | 2.35 |
| 20736 | 68 | 28.76 | 9 | 3.74 | 4 | 1.65 | 256 | 13.56 |
| 65536 | 101 | 146.21 | 9 | 12.98 | 4 | 5.75 | 362 | 193.77 |
| 390625 | 173 | 972.26 | 9 | 106.77 | 4 | 45.62 | 491 | 2597.43 |
| 1048576 | 259 | 6011.62 | 9 | 619.08 | 4 | 247.67 | > 30000 | |

5 Conclusion and future work

This paper has developed a new parameter selection method for matrix splitting iteration methods. Concretely, we present the MTKL method to address the problems of multi-parameter optimization and kernel selection. Moreover, we propose a new matrix splitting iteration scheme, MSKP method, to solving the TDLSs (1.1). We give its convergence analysis and preconditioning strategy. To demonstrate the efficiency of our approaches, we solve a 2D diffusion equation, a 2D convection-diffusion equation, and a differential Sylvester matrix equation. Numerical results illustrate MTKL method has a huge advantage in predicting the relatively optimal splitting parameters, especially
for large-scale systems. Further, the MSKP method as a preconditioner can effectively accelerate GMRES. Especially for solving the differential Sylvester matrix equation, the speedup ratio can reach tens to hundreds of times when the scale of the system is larger than one hundred thousand.

There are still lots of works worthy of our further study. For instance, the first one is to give the convergence rate analysis of ADI methods. The second interesting work is to use machine learning to train iteration schemes and splitting parameters that are consistent with concrete problems. The third challenge work is to develop our methods for solving nonlinear systems.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China (12171412), Natural Science Foundation for Distinguished Young Scholars of Hunan Province (2021JJ10037), Hunan Youth Science and Technology Innovation Talents Project (2021RC3110), the Key Project of Education Department of Hunan Province (21A0116), Hunan Provincial Innovation Foundation for Postgraduate (CX20220647).

References

[1] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank, Matrix Riccati equations in control and systems theory, Birkhäuser, 2012.

[2] F. Amato, G. De Tommasi, and A. Pironti, Necessary and sufficient conditions for finite-time stability of impulsive dynamical linear systems, Automatica, 49 (2013), pp. 2546–2550.

[3] Z.-Z. Bai, G. H. Golub, and M. K. Ng, Hermitian and skew-hermitian splitting methods for non-hermitian positive definite linear systems, SIAM Journal on Matrix Analysis and Applications, 24 (2003), pp. 603–626.

[4] Z. Z. Bai, G. H. Golub, and M. K. Ng, On successive-overrelaxation acceleration of the hermitian and skew-hermitian splitting iterations, Numerical Linear Algebra with Applications, 14 (2007), pp. 319–335.

[5] P. Benner, R.-C. Li, and N. Truhar, On the adi method for sylvester equations, Journal of Computational and Applied Mathematics, 233 (2009), pp. 1035–1045.

[6] E. V. Bonilla, K. Chai, and C. Williams, Multi-task gaussian process prediction, Advances in neural information processing systems, 20 (2007).

[7] W. Briley and H. McDonald, On the structure and use of linearized block implicit schemes, Journal of Computational Physics, 34 (1980), pp. 54–73.
[8] L. Brugnano and D. Trigiante, *Solving differential equations by multistep initial and boundary value methods*, CRC Press, 1998.

[9] K. Burrage, *Parallel and sequential methods for ordinary differential equations*, Clarendon Press, 1995.

[10] J. C. Butcher, *Numerical methods for ordinary differential equations*, John Wiley & Sons, 2016.

[11] H. Chen, *A splitting preconditioner for the iterative solution of implicit runge-kutta and boundary value methods*, BIT Numerical Mathematics, 54 (2014), pp. 607–621.

[12] H. Chen, *Generalized kronecker product splitting iteration for the solution of implicit runge-kutta and boundary value methods*, Numerical Linear Algebra with Applications, 22 (2015), pp. 357–370.

[13] M. Cirant and A. Goffi, *On the existence and uniqueness of solutions to time-dependent fractional mfg*, SIAM Journal on Mathematical Analysis, 51 (2019), pp. 913–954.

[14] M. J. Corless and A. Frazho, *Linear systems and control: an operator perspective*, CRC Press, 2003.

[15] J. Douglas and H. H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Transactions of the American Mathematical Society, 82 (1956), pp. 421–439.

[16] M. J. Gander, *Optimized schwarz methods*, SIAM Journal on Numerical Analysis, 44 (2006), pp. 699–731.

[17] D. F. Griffiths and D. J. Higham, *Numerical methods for ordinary differential equations: initial value problems*, vol. 5, Springer, 2010.

[18] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time dependent problems and difference methods*, vol. 24, John Wiley & Sons, 1995.

[19] C. D. Hauck and R. G. McClarren, *A collision-based hybrid method for time-dependent, linear, kinetic transport equations*, Multiscale Modeling & Simulation, 11 (2013), pp. 1197–1227.

[20] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1994.

[21] K. Jiang, X. Su, and J. Zhang, *A general alternating-direction implicit framework with gaussian process regression parameter prediction for large sparse linear systems*, SIAM Journal on Scientific Computing, 44 (2022), pp. A1960–A1988.
[22] H. Lee, J. Lee, and D. Sheen, *Laplace transform method for parabolic problems with time-dependent coefficients*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 112–125.

[23] P.-L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[24] R. McLachlan and M. Perlmutter, *Conformal hamiltonian systems*, Journal of Geometry and Physics, 39 (2001), pp. 276–300.

[25] D. W. Peaceman and H. H. Rachford, Jr, *The numerical solution of parabolic and elliptic differential equations*, Journal of the Society for Industrial and Applied Mathematics, 3 (1955), pp. 28–41.

[26] T. Penzl et al., *A matlab toolbox for large lyapunov and riccati equations, model reduction problems, and linear–quadratic optimal control problems*, Software available at https://www.tu-chemnitz.de/sfb393/lyapack, (2000).

[27] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[28] B. I. Schneider, L. A. Collins, and S. Hu, *Parallel solver for the time-dependent linear and nonlinear schrödinger equation*, Physical Review E, 73 (2006), p. 036708.

[29] R. S. Varga, *Iterative analysis*, Springer, 1962.

[30] X. Wang, W.-W. Li, and L.-Z. Mao, *On positive-definite and skew-hermitian splitting iteration methods for continuous sylvester equation ax+ xb= c*, Computers & Mathematics with Applications, 66 (2013), pp. 2352–2361.

[31] G. Wanner and E. Hairer, *Solving ordinary differential equations II*, vol. 375, Springer Berlin Heidelberg, 1996.

[32] A. Wilson and R. Adams, *Gaussian process kernels for pattern discovery and extrapolation*, in International conference on machine learning, PMLR, 2013, pp. 1067–1075.

[33] A. Wilson and H. Nickisch, *Kernel interpolation for scalable structured gaussian processes (kiss-gp)*, in International conference on machine learning, PMLR, 2015, pp. 1775–1784.

[34] A. G. Wilson, *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*, PhD thesis, Citeseer, 2014.

[35] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, *Deep kernel learning*, in Artificial intelligence and statistics, PMLR, 2016, pp. 370–378.