A CLASS OF GADI METHODS FOR TIME-DEPENDENT LINEAR SYSTEMS WITH MULTITASK KERNEL-LEARNING PARAMETER PREDICTION *

KAI JIANG†, JUAN ZHANG†, AND QI ZHOU†

Abstract. This paper develops a class of general alternating-direction implicit (GADI) iteration methods for solving time-dependent linear systems (TDLS), including linear differential systems and linear matrix systems. We present a GADI Kronecker product splitting (GADI-KP) method and prove the convergence with weak restrictions. The generalized Kronecker product splitting method and Kronecker product splitting method can be unified in the GADI-KP framework. Then, we use the framework to design an effective preconditioner of Krylov subspace methods for solving TDLS. The GADI-KP method is sensitive to the splitting parameters. Different from traditional theoretical estimate methods, we propose multitask kernel learning Gaussian process regression (GPR) method to predict the relative optimal splitting parameters. This method has solved the multi-parameter optimization in GADI framework and kernel selection in GPR method. Finally, we apply our approach to solve a two-dimensional diffusion equation, a two-dimensional convection-diffusion equation, and a differential Sylvester matrix equation. Numerical experiments illustrate that the GADI-KP framework and its preconditioning form have advantage over efficiency and superiority compared with the existing results.

Key words. Time-dependent linear systems, general alternating direction implicit framework, Kronecker product splitting, convergence, preconditioning, multitask kernel learning Gaussian process regression.

AMS subject classifications. 15A24, 65F10

1. Introduction.

1.1. Background. In this paper, we consider the time-dependent linear system (TDLS) of this form

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

where \( x(t) : [0, T] \rightarrow V, \) \( V \) is \( \mathbb{R}^n \) or \( \mathbb{C}^n \) \((n \in \mathbb{N})\), \( x(0) = x_0 \in V \) is an initial value, and \( L \) is a linear operator. TDLS appears in many branches of science and engineering, such as dynamical systems, quantum mechanics, linear Hamiltonian dynamics, linear kinetic equations, semi-discretization of partial differential equations, differential matrix equation, etc \([18, 13, 28, 24, 19, 2]\). Among them, we focus on linear differential systems and linear matrix systems. If \( L \) is a differential operator on space, TDLS becomes the linear differential system

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

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. The differential matrix equation, such as the Sylvester
matrix equation, could be the following form

\[
\begin{align*}
\dot{X}(t) &= AX(t) + X(t)B + EF^T, \\
X(t_0) &= X_0,
\end{align*}
\]

where \( X(t) \in \mathbb{R}^{n \times n} \) for each \( t \in [0, T] \), \( A, B \in \mathbb{R}^{n \times n} \) are large and sparse, and \( E, F \in \mathbb{R}^{n \times s} \) are full rank matrices with \( s \ll n \). The initial condition is \( X_0 = Z_0 \tilde{Z}^T \), where \( Z_0, \tilde{Z}_0 \in \mathbb{R}^{n \times s} \).

The TDLS (1.1) can be discretized by appropriate approaches, such as linear multistep schemes, Runge-Kutta methods, general linear methods, block implicit methods, boundary value methods (BVMs) [31, 17, 10, 16, 22, 8, 7]. By the time discretization scheme, TDLS (1.1) becomes a linear algebraic equation, \( Qx = b \), where \( Q \) is a block sparse matrix. This linear system is usually large. Directly solving it may be expensive. Alternatively, iteration methods become a better choice. Alternating direction implicit (ADI) methods are a class of useful iteration schemes, which were initially designed to solving partial differential equations [25, 15]. Afterward, the idea of the ADI methods has been extended to more branches, such as including numerical algebra, optimization [29, 23]. Among these ADI methods, a large class of Hermitian and skew-Hermitian splitting (HSS) type methods have been developed to solve linear systems [3, 4, 30, 5]. More recently, a general ADI (GADI) framework has been presented to put most existing ADI methods into a unified framework [21]. In this paper, we are concerned with the development of GADI methods for TDLS.

From the theoretical viewpoint, the convergence of exist GADI methods requires that one splitting matrix is positive definite and the other is positive semi-definite [21]. Similarly, the convergence of some HSS methods needs that the matrix \( Q \) of system is positive definite [3, 4]. For the TDLS, these assumptions may be difficult to meet. Therefore, it requires to design new GADI methods with more reasonable convergence assumptions.

Further, as splitting schemes, ADI methods require to split the matrix into different parts with splitting parameters. And the convergence and efficiency of ADI methods are very sensitive to splitting parameters. Therefore, choosing the optimal splitting parameters is critical. A common approach is using theoretical analysis to estimate the bound of splitting parameters in a case-by-case way [3, 11]. However, in practice, this method has certain limitation for solving large-scale systems and the efficiency heavily depends on the theoretical bound. Recently, Ref. [21] has presented a data-driven approach, the Gaussian process regression (GPR) method by choosing appropriate kernel, to predict relatively optimal splitting parameters. The GPR has a lot of advantages such as being easy to implement, high precision, and high generalization capability. The GPR method can efficiently predict one splitting parameter one time [21]. However, there may be two or more splitting parameters in GADI methods. And these splitting parameters may have some complicated connections. Independently predicting each splitting parameter would inevitably affect the predicted accuracy of original GPR method. Therefore, it requires to improve the GRP method to predict multi-parameters simultaneously. Another critical component that determines the GPR’s availability is the kernel function [33, 35]. The original GRP method chooses kernel function by the problem’s properties [21], which might produce artificial error. Meanwhile the chosen kernel in a kind of problem may be difficult to extend to other problems. Therefore, how to automatically learn kernel functions from problems is still a challenge in predicting splitting parameters.
1.2. Our work. In this work, we mainly concern developing new and efficient GADI methods to solve TDLS (1.1). Our contributions are summarized as follows:

- Inspired by the idea of GADI framework [21] and Kronecker product splitting iteration method [11, 12], we propose a GADI Kronecker product splitting (GADI-KP) method for solving TDLS (1.1). We design a fast algorithm to implement the GADI-KP method using the properties of Kronecker product.
- We prove that the GADI-KP method converges to the unique solution when all eigenvalues of one splitting matrix have positive real parts and those of another have nonnegative real parts, which does not require the positive definiteness.
- We present a multitask kernel learning GPR method to predict multiple splitting parameters simultaneously and to automatic learn kernel function.
- Based on the GADI-KP method, we provide an effective preconditioner to accelerate Krylov subspace methods in solving TDLS (1.1). The spectrum of the preconditioner matrix has a tight cluster.
- We apply our developed methods to some standard TDLSs, including two-dimensional (2D) diffusion and convection-diffusion equations, and the differential Sylvester matrix equation. Numerical results demonstrate the efficiency of our methods.

1.3. Organization. The rest of this paper is organized as follows. In section 2, we describe the discrete process of TDLS by BVM in detail, and give two concrete applications of linear differential systems and linear matrix systems. In section 3, we propose the GADI-KP method for TDLS, and prove the convergence property. We present a preconditioning strategy based on the GADI-KP method to speed up Krylov subspace methods. Additionally, to improve the GPR method, we propose the multitask kernel learning GPR method. In section 4, we show the efficiency and superiority of our methods by solid numerical experiments. In section 5, we carry out the summary of whole paper and future work prospects.

2. Discretization of TDLS. In this section, we describe the discrete process of TDLS (1.1) by BVM, and apply it to linear differential systems and linear matrix systems.

2.1. Notation. Throughout the paper, the sets of \( m \times n \) complex and real matrices are denoted by \( \mathbb{C}^{m \times n} \) and \( \mathbb{R}^{n \times n} \), respectively. If \( A \in \mathbb{C}^{n \times n} \), let \( A^T \) and \( A^{-1} \) denote the transpose and inverse of \( A \). \( \sigma(A) \) and \( \rho(A) \) denote the spectral set and spectral radius of \( A \). \( I_n \) is an \( n \)-order identity matrix. \( \| \cdot \|_2 \) represents the \( L_2 \)-norm of a vector. Let \( A = (a_{ij}) \in \mathbb{C}^{m \times n} \), \( B \in \mathbb{C}^{p \times q} \), the Kronecker product of \( A \) and \( B \) is defined by

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

We recall some properties of Kronecker products. More context about the Kronecker product can refer to [20].

**Lemma 2.1.** Let \( A, C \in \mathbb{R}^{m \times n} \), \( B, D \in \mathbb{R}^{n \times n} \), then

1. \( (A + C) \otimes (B + D) = A \otimes B + A \otimes D + C \otimes B + C \otimes D \);
2. \( (A \otimes B)(C \otimes D) = (AC) \otimes (BD) \);

This manuscript is for review purposes only.
(3) \((A \otimes B)^{-1} = A^{-1} \otimes B^{-1}\);
(4) \((A \otimes B)x = \text{vec}(BXAT), \text{vec}(X) = x\);
(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, \cdots, x_m] \in \mathbb{R}^{n \times m} \iff \text{vec}(X) = [x_1^T, x_2^T, \cdots, x_m^T]^T \in \mathbb{R}^{nm}.

2.2. Temporal Discretization. We give a brief description of BVM \([8, 9]\). For \(i = k_1, \ldots, m - k_2 (k = k_1 + k_2)\), the \(k\)-step BVM is

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

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

\[
\sum_{j=0}^{k} \alpha_{j} x_j = \tau \sum_{j=0}^{k} \beta_{j} L x_j, \quad i = 1, \cdots, k_1 - 1,
\]

\[
\sum_{j=0}^{k} \alpha_{j} x_{m-k+j} = \tau \sum_{j=0}^{k} \beta_{j} L x_{m-k+j}, \quad i = m - k_2 + 1, \cdots, m,
\]

where the coefficients \(\alpha_j^i\) and \(\beta_j^i\) are chosen such that the truncation errors over all node are consistent.

By applying the BVM, the TDLS (1.1) is discretized as

\[
Qx := (A - \tau B L)x = b,
\]

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

\[
A = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
\alpha_0^1 & \alpha_1^1 & \cdots & \alpha_k^1 \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^{k_1} & \alpha_1^{k_1} & \cdots & \alpha_k^{k_1} \\
\alpha_0 & \alpha_1 & \cdots & \alpha_k \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^{m-k_2+1} & \alpha_1^{m-k_2+1} & \cdots & \alpha_k^{m-k_2+1} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_0^m & \alpha_1^m & \cdots & \alpha_k^m
\end{bmatrix},
\]
2.3. Applications. In this work, we consider two types of TDLSs, the linear differential system (1.2), and the linear matrix system (1.3).

2.3.1. Linear differential system. By appropriately spatial discretization, such as finite difference and finite element methods, we can obtain the ordinary differential equation (ODE) system

\[
\begin{align*}
M\dot{U}(t) &= -KU(t) + F(t), \quad t \in [0, T], \\
U(0) &= \Psi, 
\end{align*}
\]

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. Then we apply BVM to (2.4), we can arrive at a linear system

\[
Qu = (A \otimes M + \tau B \otimes K)u = b,
\]

where \( u = [\Psi, U_1, \cdots, U_m]^T, \quad b = \tau(B \otimes I_{n^d})f + e_1 \otimes \Psi, \quad f = [F_0, F_1, \cdots, F_m]^T. \)

2.3.2. Linear matrix system. Now we consider the time-dependent linear matrix system (1.3). The equivalent ODE system of (1.3) is

\[
\begin{align*}
M\dot{x}(t) &= -Kx(t) + f(t), \quad t \in [0, T], \\
x(0) &= vec(X_0),
\end{align*}
\]

where \( M \) is an \( n^2 \times n^2 \) identity matrix, \( K = -(I_n \otimes A + B^T \otimes I_n), \quad x(t) = vec(X(t)) \) and \( f(t) = vec(E^T). \) Then we can also apply the BVM to (2.6) to obtain a similar linear system as (2.5).

3. Our methods. In this section, we propose GADI-KP method, a new splitting iteration method, to solve TDLS (1.1). Compared with existing ADI methods, this method can converge to the unique solution under a mild condition. More significantly, to improve the GPR method in multi-parameter prediction, we introduce the multitask kernel learning GPR method. We also present a preconditioning strategy based on the GADI-KP method to accelerate the Krylov subspace methods.
3.1. GADI-KP method. In this subsection, we propose the GADI-KP method and give a fast algorithm of GADI-KP method by using the properties of Kronecker product. The generalized Kronecker product splitting (GKPS) method [11, 12] is

\begin{align}
(3.1a) & \quad ((A + \alpha B) \otimes M)u^{(k+\frac{1}{2})} = b - (B \otimes (\tau K - \alpha M))u^{(k)}, \\
(3.1b) & \quad (B \otimes (\tau K + \beta M))u^{(k+1)} = b - ((A - \beta B) \otimes M)u^{(k+\frac{1}{2})}.
\end{align}

Inspired by the GADI framework [21], we add a viscosity term to the right side of (3.1b)

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

Then we obtain the GADI-KP scheme

\begin{align}
(3.2) & \quad \begin{cases}
(A + \alpha B) \otimes Mu^{(k+\frac{1}{2})} = b - B \otimes (\tau K - \alpha M)u^{(k)}, \\
B \otimes (\tau K + \beta M)u^{(k+1)} = B \otimes \left(\tau K + \left(\frac{\alpha + \beta}{2} \omega - \alpha\right)M\right)u^{(k)} \\
+ \frac{\alpha + \beta}{2} (2 - \omega) B \otimes Mu^{(k+\frac{1}{2})},
\end{cases}
\end{align}

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

Eliminating the intermediate vector \( u^{(k+\frac{1}{2})} \), we can write the GADI-KP 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 of the GADI-KP method is

\begin{align}
(3.3) & \quad T(\alpha, \beta, \omega) = [(A + \alpha B) \otimes (\tau K + \beta M)]^{-1} \\
& \cdot \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],
\end{align}

and

\begin{align}
(3.4) & \quad P(\alpha, \beta, \omega) = \frac{2}{(\alpha + \beta)(2 - \omega)}(A + \alpha B) \otimes (\tau K + \beta M).
\end{align}

It is easy to see that there is a unique splitting

\[
Q = P(\alpha, \beta, \omega) - R(\alpha, \beta, \omega),
\]
where
\[ R(\alpha, \beta, \omega) = \frac{2}{(\alpha + \beta)(2 - \omega)} \cdot \left[ (A - \beta B) \otimes (\tau K - \alpha M) + (A + \alpha B) \otimes \frac{\alpha + \beta}{2} \omega M + \frac{\alpha + \beta}{2} \omega B \otimes (\tau K - \alpha M) \right]. \]

Notice that
\[ T(\alpha, \beta, \omega) = P(\alpha, \beta, \omega)^{-1} R(\alpha, \beta, \omega) = I - P(\alpha, \beta, \omega)^{-1} Q. \]

Therefore, the GADI-KP method (3.2) reduces to
\[ u^{(k+1)} = u^{(k)} + P(\alpha, \beta, \omega)^{-1} r^{(k)}, \quad r^{(k)} = b - Q u^{(k)}. \]

Using the properties of Kronecker product, we can obtain a fast implementation of the GADI-KP method (see Algorithm 3.1).

Algorithm 3.1 GADI-KP method for solving linear system (2.5).

\begin{algorithm}
\begin{algorithmic}[1]
\Require Initial guess $u^{(0)} \in \mathbb{R}^{n \times m}$, splitting parameters $\alpha, \beta > 0$ and $0 \leq \omega < 2$;
\For {$k = 0, 1, 2, \ldots$ until \{\text{$u^{(k)$}}\} converges \Do
\State $r^{(k)} = \frac{(\alpha + \beta)}{(2 - \omega)} (b - Q u^{(k)})$;
\State $\text{vec}([r^{(k)}_1, r^{(k)}_2, \ldots, r^{(k)}_m]) = r^{(k)}$;
\For {$i = 1, 2, \ldots, m$ \Do
\State $(\tau K + \beta M) u^{(k)}_i \approx r^{(k)}_i$ (use GMRES);
\EndFor
\State $\nu^{(k)} = \text{vec} (\nu^{(k)}_1, \nu^{(k)}_2, \ldots, \nu^{(k)}_m) \cdot (A + \alpha B)^{-T}$;
\State $u^{(k+1)} = u^{(k)} + \nu^{(k)}$;
\EndFor
\end{algorithmic}
\end{algorithm}

Remark 3.1. For each iteration of the GADI-KP method, the computational complexity by directly solver is $O \left( (n^d m)^3 \right)$. Base on Algorithm 3.1, the computational complexity can be reduced to $O \left( (n^d m)^2 + n^d m^2 + n^d m \right)$.

3.2. Convergence analysis of GADI-KP. In this subsection, we discuss the convergence of GADI-KP method.

Theorem 3.2. Assume that all eigenvalues of $B^{-1} A$ have positive real parts and all the eigenvalues of $M^{-1} K$ have nonnegative real parts. 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 GADI-KP method (3.2) is convergent to the unique solution $u^*$ of the linear system (2.5). The spectral radius $\rho(T(\alpha, \beta, \omega))$ of iteration matrix $T(\alpha, \beta, \omega)$ of GADI-KP method satisfies
\[ \rho(T(\alpha, \beta, \omega)) \leq \frac{1}{2} |(2 - \omega) \varphi(\alpha, \beta) + \omega| < 1, \]
where
\[ \varphi(\alpha, \beta) := \max_{\lambda \in \sigma(B^{-1} A + \frac{\alpha + \beta}{2} I_m)} \left| \frac{\lambda - (\alpha + \beta)/2}{\lambda + (\alpha + \beta)/2} \right|. \]
Proof. Let $\lambda$ and $\mu$ be the eigenvalues of the matrices $B^{-1}A$ and $\tau M^{-1}K$, respectively. Then by the property of the Kronecker product, the eigenvalues of the iteration matrix $T(\alpha, \beta, \omega)$ (3.3) have the following form:

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

Let $\alpha = \hat{\alpha} + \hat{\beta}$ and $\beta = \hat{\alpha} - \hat{\beta}$, then

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

which is equivalent to

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

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

Notice that

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

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

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

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

$$p = \frac{1}{2}[(2 - \omega)q + \omega].$$

Let $q = a + bi$ ($i = \sqrt{-1}$), then

$$|p| = \frac{1}{2} |(2 - \omega)(a + bi) + \omega|$$

$$= \frac{1}{2} \sqrt{(2 - \omega)^2 a^2 + \omega^2 + 2(2 - \omega)a \omega + (2 - \omega)^2 b^2}$$

$$\leq \frac{1}{2} \sqrt{(2 - \omega)^2 (a^2 + b^2) + \omega^2 + 2(2 - \omega) \omega \sqrt{a^2 + b^2}}$$

$$= \frac{1}{2} \sqrt{(2 - \omega)^2 |q|^2 + \omega^2 + 2(2 - \omega) \omega |q|}$$

$$= \frac{1}{2} [(2 - \omega)|q| + \omega].$$

If $0 < \beta \leq \tau \min_{\xi \in \sigma(M^{-1}K)} Re(\xi)$ and $\alpha > 0$, we obtain

$$\frac{\mu - \hat{\alpha}}{\mu + \hat{\alpha}} \leq 1, \quad \mu \in \sigma(\tau M^{-1}K - \hat{\beta}I_m).$$
When \(0 \leq \omega < 2\), \(Re(\lambda) > 0\), we have

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

This implies that the GADI-KP method (3.2) converges to the unique solution \(u^*\) of linear system (2.5).

3.3. Krylov subspace acceleration with GADI-KP preconditioner. In this subsection, we give a preconditioning strategy based on the GADI-KP method to accelerate Krylov subspace methods.

The GADI-KP method can be used as a preconditioner to accelerate Krylov subspace methods such as GMRES [27]. From (3.5), the linear system (2.5) is equivalent to

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

where \(c = P(\alpha, \beta, \omega)^{-1}b\) and the matrix \(P(\alpha, \beta, \omega)\) is a preconditioner. This equivalent system can be solved by GMRES. Algorithm 3.2 gives the solving process for (2.5) by putting the matrix \(P(\alpha, \beta, \omega)\) as a preconditioner in GMRES. Notice that using the GADI-KP preconditioner within GMRES requires solving a linear system \(P(\alpha, \beta, \omega)v = r\) at each iteration. We can use Algorithm 3.1 to economically solve the above preconditioned linear system.

**Algorithm 3.2** GMRES-GADI-KP method for solving linear system (2.5).

**Require:** Initial guess \(u^{(0)} \in \mathbb{R}^{n \times m}\), splitting parameters \(\alpha, \beta > 0\) and \(0 \leq \omega < 2\);

1. Calculate preconditioner \(P(\alpha, \beta, \omega)\) according to (3.4);
2. for \(k = 1, 2, \ldots\) 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 \(j = 1, 2, \ldots, k\) do
5. \(\tilde{w}_j = Qv_j\);
6. \(w_j = P(\alpha, \omega)^{-1}\tilde{w}_j\);
7. for \(i = 1, 2, \ldots, j\) do
8. \(h_{ij} = (w_j, v_i)\);
9. end for
10. \(\tilde{v}_{j+1} = w_j - \sum_{i=1}^j h_{i,j}v_i\);
11. \(h_{j+1,j} = \|\tilde{v}_{j+1}\|_2\);
12. \(v_{j+1} = \tilde{v}_{j+1}/h_{j+1,j}\);
13. end for
14. \(u_k = u_0 + V_k y_k\), where \(y_k\) minimizes \(\|r_0\|_2 e_1 - H_k y\|_2\);
15. end for

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

![Eigenvalue Distribution](image)

Fig. 1: The eigenvalue distribution of the matrix $Q$ 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).

3.4. Multitask kernel learning GPR. In this subsection, we propose a multitask kernel learning GPR method which improves both simultaneous prediction of multiple splitting parameters and data-driven kernel selection.

3.4.1. Multitask GPR. GADI-type methods have two or more splitting parameters. As we know, the splitting parameters heavily affect the efficiency of GADI approaches. Now we propose a practical multitask GPR method to simultaneously predict these splitting parameters. In the multitask method, we are interested in learning $M$ related functions $f_l$ ($l = 1, \ldots, M$), from training data $(x_{li}, y_{li})(i = 1, \ldots, n_l)$, with $x_{li} \in \mathbb{R}^d$, $y_{li} \in \mathbb{R}$ and $n_1 + \cdots + n_M = N$. Consider the following noised model to avoid singularity in numerical computation

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

where $y_{li}$, $(x_{li})$ denotes the $i$-th output (input) of the $l$-th task, $\epsilon_l$ is the white noise of the $l$-th task.
Let \( y = \text{vec}(Y^T) = [y_{1,1}, \ldots, y_{1,n_1}, \ldots, y_{M,1}, \ldots, y_{M,n_M}]^T \) be the output vector, and latent function \( f = \text{vec}(F^T) \) be the input vector. The multitask regression problem can be presented as the Gaussian process prior over the latent function

\[
f \sim \mathcal{GP}(0, K^t \otimes K^x),
\]

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 (3.9) becomes

\[
y \sim \mathcal{N}(0, K^t \otimes K^x + D \otimes I_N),
\]

where \( D \in \mathbb{R}^{M \times M} \) is a diagonal matrix with \( D_{ll} = \sigma_l^2 \). The prediction distribution for the \( l \)-th task \( \mathbf{y}_l | \mathbf{y} \) on a new point \( \mathbf{x}_* \) is

\[
\mathbf{y}_l | \mathbf{y} \sim \mathcal{N}(\mu_l, \Sigma_l),
\]

where

\[
\mu_l = (k^t_l \otimes k^x_{x,x,*})^T \Sigma^{-1} \mathbf{y},
\]

\[
\Sigma_l = k^t_l k^x_{x,x,*} - (k_l^t \otimes k^x_{x,x,*})^T \Sigma^{-1} (k^t_l \otimes k^x_{x,x,*}),
\]

\( \Sigma = K^t \otimes K^x + D \otimes I_N \). \( k^t_l \) and \( k^x_l \) are the \( l \)-th diagonal element and the \( l \)-th column of \( K^t \), respectively. \( k^x_{x,x,*} \) is a covariances vector between test point \( \mathbf{x}_* \) and training point \( \mathbf{x} \), 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. In order to achieve the hyperparameter estimation, we apply the L-BFGS method to maximize marginal likelihood function \( L \) in logarithmic form

\[
L = \log p(\mathbf{y}|\mathbf{x}, \theta_x, \theta_t)
= -\frac{N}{2} \log |K^t| - \frac{M}{2} \log |K^x| - \frac{1}{2} \text{trace } [(K^t)^{-1} F^T (K^x)^{-1} F]
- \frac{N}{2} \sum_{l=1}^{M} \log \sigma_l^2 - \frac{1}{2} \text{trace } [(Y - F) D^{-1} (Y - F)^T] - \frac{MN}{2} \log 2\pi.
\]

### 3.4.2. Kernel learning approach

Choosing an appropriate kernel function determines whether GRP method succeeds. The original GRP method directly selects kernel function from the feature of problems [21]. It might result in artificial error and could be hardly extended to other problems. For example, Figure 3 shows that the GPR method with manual kernel function will produce a bad 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 \( K = \{k_\xi(x, x') | \xi = 1, \ldots, N\} \) 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

\[
k(x, x') = \sum_{\xi=1}^{N} c_\xi k_\xi(x, x').
\]
For the $N$ training multitask, 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 [34, 32].

Remark 3.3. In GADI methods, the training data comes from the smaller systems [21]. It is a small data set. Learning kernel function by (deep) neural network from small data is difficult. Therefore, we predetermine the kernel library via a priori knowledge instead of directly learning kernel function from data.

4. Numerical experiments. In this section, we present three numerical examples: 2D diffusion and convection-diffusion equations, and the differential Sylvester matrix equation, to show the performance of our proposed GADI-KP, GMRES with GADI-KP preconditioner (GMRES-GADIKP) methods. As a comparison, we also give the experimental data of KPS, GKPS, GMRES, and GMRES with GKPS preconditioner (GMRES-GKPS) methods. All computations are carried out using Py-Charm 2020.2 on a Mac laptop with 2.3 GHz Quad Intel Core i5. All tests are started with the zero vector. All iteration methods are terminated if the relative residual error satisfies $RES = \|r^{(k)}\|_2/\|r^{(0)}\|_2 \leq 10^{-6}$, where $r^{(k)} = b - Qu^{(k)}$ is the $k$-step residual.

In the multitask kernel learning GPR method, the kernel library $\mathcal{K}$ contains linear, Gaussian, periodic kernels, and their multiplicative combinations. Concretely, three basic kernel functions are

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

The output variance $\sigma_f^2$ determines the average distance of the function away from its mean. The offset $c$ determines the $x$-coordinate of the point that all the lines in the posterior go through. The lengthscale $\iota$ determines the length of the ‘wiggles’. The period $p$ simply determines the distance between repetitions of the function. The multiplicative combination kernels are $k_U = k_l k_l$, $k_I = k_l k_g$, $k_P = k_l k_p$, and $k_{gp} = k_g k_p$. As a sequence, the kernel library is $\mathcal{K} = \{k_l, k_g, k_p, k_U, k_I, k_P, k_{gp}\}$.

4.1. 2D diffusion equation. First, we consider the 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. Using centered difference scheme on an $n \times n$ system with mesh size $h = 1/(n + 1)$ on the unit square, we obtain an $n^2$-dimensional
ODE (2.4), 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. $T_n = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{n \times n}$. Then we discretize the ODE system by the fifth-order generalized Adams method (GAM-5) in the BVM framework [8] using uniform time grid on $[0,1]$ with time step size $\tau = 1/(m-1)$. The full discretize linear system as (2.5) is

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -1 & \cdots & -1 & 1 \end{bmatrix}, \quad B = \frac{1}{720} \begin{bmatrix} 0 & 251 & 646 & -264 & 106 & -19 \\ -19 & 346 & 456 & -74 & 11 \\ -19 & -74 & 456 & 346 & -19 \\ -19 & 106 & -264 & 646 & 251 \end{bmatrix}.$$  

4.1.1. **Predicting optimal parameters.** We use the multitask kernel learning GPR approach to predict splitting parameters $(\alpha, \beta, \omega)$ of GADI-KP type methods. Table 1 gives the training, test, and retrained data sets of $(\alpha, \beta, \omega)$. $(\alpha, \beta, \omega)$ in the training data set of the multitask kernel learning GPR approach is produced by traversing parameters as common splitting methods done but for small scale linear systems, $m$ from 10 to 128 with different step size $\Delta m$. The traversing parameters $\alpha$ and $\beta$ are obtained by traversing interval $[0,5)$ with a step size of 0.01. The traversing parameter $\omega$ is obtained by traversing interval $[0,2)$ with a step size of 0.01. For test data set $(\alpha^*, \beta^*, \omega^*)$, 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 retrained data set $(\alpha^*, \beta^*, \omega^*)$, let $m$ be from 128 to 500 with $\Delta m = 30$.

| 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 multitask kernel learning method considers a linear combination of $k_g$, $k_p$, $k_{gp}$ in kernel library $K$. 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 2 shows the optimal parameter regression curve of $(\alpha, \beta, \omega)$ against $m$ for the GADI-KP method. The optimized hyperparameters are

$$C = \begin{bmatrix} 1.07 & 0.02 & 0.13 \\ 0.93 & 0.01 & 0.06 \\ 0.31 & 0.66 & 0.47 \end{bmatrix}, \quad K' = \begin{bmatrix} 15.91 & 0.00 & 0.00 \\ 0.00 & 1.07 & 0.01 \\ 0.00 & 0.01 & 1.22 \end{bmatrix}, \quad \iota = 139.60, \quad p = 8.03.$$  

Notice that, three kernels $k_g(x, x')$, $k_p(x, x')$ and $k_{gp}(x, x')$ in the combination all play important roles. As an example, Figure 3 demonstrates that when only periodic kernel is considered, the GRP method falls to predict accurate splitting parameter $\omega$.

4.1.2. **Comparison.** In this subsection, we compare our methods with the KPS, GKPS, GMRES, GMRES-GKPS methods for solving 2D diffusion equation (4.1). The optimal parameters of KPS, GKPS, and GMRES-GKPS methods are obtained.
Fig. 2: Regression curves of \((\alpha, \beta, \omega)\) against \(m\) (fixed \(n = 16\)) when solving 2D diffusion equation (4.1) with the GADI-KP method. Up: no retraining; Bottom: retraining.

Fig. 3: Regression curve of \(\omega\) against \(m\) (fixed \(n = 16\)) when solving 2D diffusion equation (4.1) with the GADI-KP method. Here we only use the periodic kernel in the GRP method.

by traversing approach, and the optimal parameters of GADI-KP scheme are obtained by multitask kernel learning GRP method.

Table 2 compares the numerical results of KPS, GKPS, GADI-KP, GMRES, GMRES-GKPS, and GMRES-GADIKP methods with different discrete resolution. “IT” and “CPU” denote the required iterations and the CPU time (in seconds), respectively. Numerical results show that the GADI-KP method has the best convergence performance over the KPS and GKPS methods. And the GADI-KP method as a preconditioner can better accelerate the GMRES methods. As the dimension of the system increases, the advantages of our approaches expand significantly.

It is worth noting that, in order to make the KPS, GKPS, and GMRES-GKPS methods reach the most effective situation, it is necessary to traverse the parameters to select optimal parameters. In detail, we obtain a relatively optimal parameter by traversing interval \((0, 5]\) with a step size of 0.01. We give Table 3 to show the
traversal CPU of different methods when $h = \tau = 1/64$. As the dimension of the system becomes larger, the traversal time increases dramatically, making the computer unaffordable. Once again, it shows the important role of multitask kernel learning GPR method for the parameter selection of iteration methods with parameters.

Table 2: Results of solving 2D diffusion equation (4.1) with different methods.

| Method       | $\tau$ | $h = 1/16$ | $h = 1/32$ | $h = 1/64$ |
|--------------|--------|------------|------------|------------|
| KPS          | IT     | 33         | 36         | 37         |
|              | CPU    | 0.67       | 1.58       | 5.56       |
| GKPS         | IT     | 19         | 19         | 19         |
|              | CPU    | 0.38       | 0.83       | 2.89       |
| GADI-KP      | IT     | 15         | 15         | 15         |
|              | CPU    | 0.29       | 0.66       | 2.25       |
| GMRES        | IT     | 138        | 336        | 813        |
|              | CPU    | 0.32       | 2.55       | 40.09      |
| GMRES-GKPS   | IT     | 12         | 12         | 12         |
|              | CPU    | 0.18       | 0.42       | 2.31       |
| GMRES-GADI-KP| IT     | 11         | 11         | 12         |
|              | CPU    | 0.16       | 0.39       | 2.28       |

Table 3: Traversal CPU of different methods in solving (4.1) when $h = \tau = 1/64$.

| Method       | KPS | GKPS | GADI-KP | GMRES | GMRES-GKPS | GMRES-GADI-KP |
|--------------|-----|------|---------|-------|------------|--------------|
| Traversal CPU| 18025.83 | 5848.68 | 0 | 0 | 9160.98 | 0 |

4.2. 2D convection-diffusion equation. The second example is 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 $f(x, y, t)$. $f$ is determined by choosing the exact solution $u_e = (1, 1, ..., 1)^T$. Using centered difference scheme on an $n \times n$ uniform grid with mesh size $h = 1/(n + 1)$ on the unit square, we obtain an $n^2 \times n^2$ dimensional ODE (2.4), in which $K = I_n \otimes P_n + Q_n \otimes I_n$ and $M$ is an $n^2 \times n^2$ identity matrix. $P_n = \text{tridiag}(-1, 2, -1)/h^2$, $Q_n = \text{tridiag}(-1/2h - 1/h^2, 2/h^2, 1/2h - 1/h^2) \in \mathbb{R}^{n \times n}$. Then we apply GAM-5 on uniform time grid with a time step size $\tau = 1/(m-1)$ to discretize the ODE system. Finally, we obtain a linear system of the form as (2.5).

4.2.1. Predicting optimal parameters. We use the multitask kernel learning GPR approach to predict the splitting parameter $(\alpha, \beta, \omega)$ of GADI-KP type methods. Table 4 gives the training, test, and retrained data sets of $(\alpha, \beta, \omega)$. $(\alpha, \beta, \omega)$ in the training data set of the multitask kernel learning GPR approach is produced by traversing parameters as the GADI-KP method done but for small scale linear systems, $m$ from 10 to 128 with different step size $\Delta m$. The traversing parameters $\alpha$ and $\beta$ are obtained by traversing interval $[0, 5]$ with a step size of 0.01. The traversing parameter $\omega$ is obtained by traversing interval $[0, 2]$ with a step size of 0.01. For test data set $(\alpha^*, \beta^*, \omega^*)$, let $m$ from 1 to 500 with $\Delta m = 1$. And for retrained data set $(\alpha^*, \beta^*, \omega^*)$, we let $m$ from 128 to 500 with $\Delta m = 30$. This manuscript is for review purposes only.
Table 4: Training, test, and retrained data sets \((\alpha, \beta, \omega)\) in the multitask kernel learning GRP approach for GADI-KP method when solving 2D convection-diffusion equation (4.2) \((n = 16)\).

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

In this case, the multitask kernel learning method considers a linear combination of \(k_g, k_p, k_{gp}\) in kernel library \(K\). Concretely, for the \(l\)-th task,

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

**Figure 4** shows the optimal parameter regression curve of \((\alpha, \beta, \omega)\) against \(m\) for the GADI-KP method. The optimized hyperparameters are

\[
C = \begin{bmatrix}
1.21 & 0.05 & 0.04 \\
1.17 & 0.03 & 0.02 \\
0.03 & 0.34 & 0.67
\end{bmatrix}, \quad K^t = \begin{bmatrix}
0.21 & 0.41 & 8.38 \\
0.41 & 0.33 & 7.65 \\
8.38 & 7.65 & 1.01
\end{bmatrix}, \quad \iota = 99.54, \quad p = 7.83.
\]

From **Figure 4**, it can be seen that adding predicted points into the training set which forms the retrained set can shrink the confidence interval. This improves the prediction accuracy and strengthens the generalization ability of the regression model.

![Regression curves](image)

**Fig. 4**: Regression curves of \((\alpha, \beta, \omega)\) against \(m\) (fixed \(n = 16\)) when solving 2D convection-diffusion equation (4.2) with the GADI-KP method. Up: no retraining; Bottom: retraining.

### 4.2.2. Comparison

In this subsection, we compare our methods with the KPS, GKPS, GMRES, GMRES-GKPS, GMRES-GADI methods solving the 2D convection-diffusion equation (4.2). The optimal parameters of KPS, GKPS, and GMRES-GKPS methods are obtained by traversing approach, and the optimal parameters of GADI-KP method are obtained by multitask kernel learning GRP method.
Table 5 compares the numerical results of KPS, GKPS, GADI-KP, GMRES, GMRES-GKPS, and GMRES-GADIKP methods with different discrete resolution. “IT” and “CPU” denote the required iterations and the CPU time (in seconds), respectively. Numerical results show that the GADI-KP method has the best convergence performance over the KPS and GKPS methods. In this experiment, we find that the GKPS method does not improve the performance of the KPS method, i.e., the parameter $\beta$ does not affect the performance of the GKPS method, but the parameter $\omega$ has a great influence. And the GADI-KP method as a preconditioner can better accelerate the GMRES methods. Similarly, as the dimension of the system increases, the advantages of our approaches expand significantly. We also give Table 6 to show the traversal CPU of different methods when $h = \tau = 1/64$.

Table 5: Results of solving 2D convection-diffusion equation (4.2) with different methods.

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

Table 6: Traversal CPU of different methods in solving (4.2) when $h = \tau = 1/64$.

| Method    | KPS/GKPS | GADI-KP | GMRES | GMRES-GKPS | GMRES-GADIKP |
|-----------|----------|---------|-------|------------|--------------|
| Traversal CPU | 23276.32 | 0       | 0     | 12324.91   | 0            |

4.3. Differential Sylvester matrix equation. Finally, we consider the differential Sylvester matrix equation (1.3). Differential Sylvester equations originate from many specific problems, such as dynamical systems, filter design theory, model reduction problems, differential equations, and robust control problems [1, 14]. Here, we focus on the matrices $A$ and $B$ obtained from the centered finite difference discretization of the operator $\mathcal{L}(u) = \Delta u + f_1(x,y)\frac{\partial u}{\partial x} + f_2(x,y)\frac{\partial u}{\partial y} + f(x,y)u$ on the unit square $[0,1]^2$ with homogeneous Dirichlet boundary condition. The number of inner grid points in each direction is $n_0$, and the dimension of the matrices $A$ and $B$ is $n = n_0^2$. For this experiment, we extract the matrices $A$ and $B$ from the Lyapack package [26] using `fdm_2d_matrix(n_0, x, y, 0)`. The matrices $\mathcal{E}, \mathcal{F} \in \mathbb{R}^{n \times 2}$ are given by random values uniformly distributed on $[0,1]$. The initial condition is $X_0 = 0$. We discrete the differential Sylvester matrix equation (1.3) in time by the GAM-5 with uniform time grid on $[0,1]$ (time step size $\tau = 0.1$), and obtain a similar linear system of the form as (2.5).

4.3.1. Predicting optimal parameters. We use the multitask kernel learning GPR method to predict the splitting parameter $(\alpha, \beta, \omega)$ of GADI-KP scheme. Table 7
gives the training, test and, retrained data sets of \((\alpha, \beta, \omega)\). \((\alpha, \beta, \omega)\) in the training data set of the multitask kernel learning GPR approach is produced by traversing parameters as the GADI-KP method done but for small scale linear systems, \(n\) from 16 to 225 with different step size. For test data set \((\alpha^*, \beta^*, \omega^*)\), let \(n\) from 1 to 1200 with \(\Delta n = 1\). And for retrained data set \((\alpha^*, \beta^*, \omega^*)\), let \(n\) from 250 to 1200 with \(\Delta n = 50\).

Table 7: Training, test, and retrained data sets \((\alpha, \beta, \omega)\) in multitask kernel learning GRP approach for GADI-KP method when solving differential Sylvester matrix equation (1.3) \((\tau = 0.1)\)

| Training set | Test set | Retrained set |
|--------------|----------|---------------|
| \(n = \gamma\), \(\gamma \in \Gamma\), \(\Gamma = \{\gamma|\gamma \in \mathbb{Z}, 4 \leq \gamma \leq 15\}\) | \(n : 1 \sim 1200, \Delta n = 1\) | \(n : 250 \sim 1200, \Delta n = 50\) |

In this case, the multitask kernel learning method considers a linear combination of \(k_g, k_l, k_{gl}\) in kernel library \(K\). Concretely, for the \(l\)-th task,

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

Figure 5 shows the optimal parameter regression curve of \((\alpha, \beta, \omega)\) against \(m\) for the GADI-KP method. The optimized hyperparameters are

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

Fig. 5: Regression curves of \((\alpha, \beta, \omega)\) against \(n\) (fixed \(\tau = 0.1\)) when solving differential Sylvester matrix equation (1.3) with the GADI-KP method.

4.3.2. Comparison. In this subsection, we compare our methods with the KPS, GKPS, and GMRES methods for solving differential Sylvester matrix equation (1.3). The optimal parameters of KPS and GKPS methods are obtained by traversing, and the optimal parameters of GADI-KP method are obtained by multitask kernel learning GRP method. Table 8 shows corresponding numerical results, where “IT” and “CPU” denote the required iterations and CPU time (in seconds), respectively. Numerical results show that our approach has significant advantages for solving linear matrix equations. When the system is large enough, GADI-KP method can speed up calculations by tens to hundreds of times over other methods.

5. Conclusion and future work. This paper is concerned with the generalization of classical ADI iteration methods for solving the large sparse asymmetric
nonpositive definite TDLS. Our proposed GADI-KP method relaxes the assumptions required for convergence on the basis of existing ADI methods. We present the multitask kernel learning GPR method to address the problems of multi-parameter optimization in GADI framework and kernel selection in GPR method. To demonstrate the efficiency of our approaches, we apply our approaches to solve a 2D diffusion equation, a 2D convection-diffusion equation, and a differential Sylvester matrix equation. Experimental results show the efficiency of our proposed methods.

There are still lots of works to study the proposed methods. For instance, the first one is to give the convergence rate analysis of the GADI framework. The second interesting work is to use machine learning to train an iterative format and splitting parameters that are consistent with concrete problems. The third challenge work is to develop our methods to solve nonlinear systems.

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

[25] 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).

[26] 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.

[27] 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.

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

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

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

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

[32] 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.

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

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