University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Zafar, A;Cantoni, M;Farokhi, F

Title:
Structured preconditioning of conjugate gradients for path-graph network optimal control problems

Date:
2021-01-01

Citation:
Zafar, A., Cantoni, M. & Farokhi, F. (2021). Structured preconditioning of conjugate gradients for path-graph network optimal control problems. IEEE Transactions on Automatic Control, PP (8), pp.1-7. https://doi.org/10.1109/TAC.2021.3107246.

Persistent Link:
https://hdl.handle.net/11343/282484

Structured preconditioning of conjugate gradients for path-graph network optimal control problems

Armaghan Zafar, Michael Cantoni, and Farhad Farokhi

***Abstract*—A structured preconditioned conjugate gradient (PCG) based solver is developed for implementing the Newton updates in second-order methods for a class of constrained network optimal control problems. Of specific interest are problems with discrete-time dynamics arising from the path-graph interconnection of N heterogeneous sub-systems. The arithmetic complexity of each PCG step is $O(NT)$, where T is the length of the time horizon. The proposed preconditioning involves a fixed number of block Jacobi iterations per PCG step. A decreasing analytic bound on the effective conditioning is given in terms of this number. The associated computations are decomposable across the spatial and temporal dimensions of the optimal control problem, into sub-problems of size independent of N and T. Numerical results are provided for two example systems.**

***Index Terms*—Optimal control of networks; Structured second-order solver; System chains.**

I. Introduction

CONSIDER the path-graph interconnection of N heterogeneous sub-systems with dynamics given by

$$x_{j,t+1} = A_{j,t}x_{j,t} + B_{j,t}u_{j,t} + E_{j,t}x_{j-1,t} + F_{j,t}x_{j+1,t}, \quad (1)$$

where $x_{j,t} \in \mathbb{R}^{n_j}$ and $u_{j,t} \in \mathbb{R}^{m_j}$ are the state and input of sub-system $j \in \mathcal{N} = \{1, 2, ..., N\}$ at time $t \in \mathcal{T} = \{0, 1, ..., T\}$, respectively. The initial conditions are given by $x_{j,0} = \boldsymbol{\xi}_j \in \mathbb{R}^{n_j}$ for $j \in \mathcal{N}$ and the spatial boundary conditions are given by $x_{0,t} = \boldsymbol{\chi}_t \in \mathbb{R}^{n_0}$ and $x_{N+1,t} = \boldsymbol{\zeta}_t \in \mathbb{R}^{n_{N+1}}$ for $t \in \mathcal{T}$. The constrained finite-horizon linear-quadratic (LQ) optimal control problem of interest is the following:

$$\min_{\substack{(x_{j,t})_{(j,t)\in(\{0,N+1\}\cup\mathcal{N})\times\mathcal{T}} \\ (u_{j,t})_{(j,t)\in\mathcal{N}\times\mathcal{T}}}} \frac{1}{2}\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} \ell_{j,t}(x_{j,t}, u_{j,t}) \quad (2a)$$

subject to

$$(1) \quad \text{for } (j,t) \in \mathcal{N} \times (\mathcal{T}\backslash\{T\}), \quad (2b)$$

$$x_{0,t} = \boldsymbol{\chi}_t, \ x_{N+1,t} = \boldsymbol{\zeta}_t \quad \text{for } t \in \mathcal{T}, \quad (2c)$$

$$x_{j,0} = \boldsymbol{\xi}_j \quad \text{for } j \in \mathcal{N}, \quad (2d)$$

$$C_{j,t}x_{j,t} + D_{j,t}u_{j,t} \leq \boldsymbol{\kappa}_{j,t} \quad \text{for } (j,t) \in \mathcal{N} \times \mathcal{T}, \quad (2e)$$

where $\ell_{j,t}(x,u) = x'Q_{j,t}x + 2x'S_{j,t}u + u'R_{j,t}u + q'_{j,t}x + r'_{j,t}u$, $C_{j,t} \in \mathbb{R}^{\nu_j \times n_j}$, $D_{j,t} \in \mathbb{R}^{\nu_j \times m_j}$ and $\boldsymbol{\kappa}_{j,t} \in \mathbb{R}^{\nu_j}$. For $j \in \mathcal{N}$ and $t \in \mathcal{T}\backslash\{T\}$, it is assumed that $Q_{j,t} = Q'_{j,t} \succeq 0$, $R_{j,t} = R'_{j,t} \succ 0$, and $Q_{j,t} - S'_{j,t}R_{j,t}^{-1}S_{j,t} \succeq 0$. Moreover, for every $j \in \mathcal{N}$, $Q_{j,T} \succeq 0$, but $S_{j,T} = 0$, $R_{j,T} = 0$, and $D_{j,T} = 0$, so that $u_{j,T}$ plays no role (i.e., it can be removed as a decision variable). Under these assumptions the problem (2) is a convex quadratic program with $O(NT)$ decision variables and $O(NT)$ constraints.

While the cost (2a) and inequality constraints (2e) are separable across the sub-systems and time horizon, there is coupling in the equality constraint (2b). Specifically, there is spatial coupling between states of adjacent sub-systems, and inter-temporal coupling. While it is possible to also accommodate corresponding spatial coupling in the inequality constraints (2e), this is not considered explicitly here to reduce the notational burden. Path-graph network dynamics of this kind are relevant in the operation of irrigation channels [1], vehicle platoons [2], supply chains [3], and radial power networks [4]. The structure also arises from the discretization of one-dimensional partial differential equations [5].

This note is about the computation of second-order search directions for solving the quadratic program (2). Specifically, a preconditioned conjugate gradient (PCG) based linear system solver (e.g., see [6]) is developed for implementing the Newton updates in second-order methods, such as the interior-point method [7]. The main innovation pertains to $O(NT)$ arithmetic complexity of each PCG step, and decomposability of the preconditioning computations across both the temporal and spatial dimensions into sub-problems of sizes that are independent of N and T. These computations are amenable to implementation as $\lceil N/2 \rceil$ parallel threads, each comprising a sequence of $2T$ (possibly dense but small) sub-problems, with localized data exchange that aligns with the path-graph structure of the system dynamics (1).

Structure in second-order methods for optimal control problems is studied in [8], [9], where the so-called Riccati-factorization approach was originally developed, and more recently in [10]–[14]. These papers all focus on the structure associated with the localized coupling in the temporal dimension of optimal control problems. Following this approach for problem (2) results in solvers with $O(TN^3)$ arithmetic complexity for each of the moderate number of Newton iterations needed for second-order methods to converge (typically $10-20$ steps). The computations are decomposable across the temporal dimension, but not the spatial dimension. The resulting sub-problems, of size $O(N)$, are amenable to implementation as parallel threads in a tree type communication network, leading to $O(\log(T)N^3)$ time complexity [14].

In [15], the aforementioned approach is pursued in the special case of (2) with directed spatial coupling, by interchanging the role of the time and space indexes to develop a linear system solver for implementing Newton updates with arithmetic complexity $O(NT^3)$. The computations are

Supported in part by the Australian Research Council (LP160100666).

The authors are with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville VIC 3010, Australia. (emails: armaghanzafar.az@gmail; {cantoni; ffarokhi}@unimelb.edu.au)

decomposable across the spatial dimension of the problem, but not the temporal dimension. Again, parallel processing can lead to $O(\log(N)T^3)$ time complexity.

In [16], the chordal property of a graph based representation of the temporal structure of optimal control problems is exploited to devise a distributed solver. At the finer resolution of the optimal control problem formulation in (2), however, the corresponding spatio-temporal graph is not chordal.

All approaches discussed above correspond to structured direct methods for solving the linear system of equations associated with each Newton update. In particular, all are related, in some way, to block-LU factorization for a permutation of variables that yields a block tri-diagonal structure in the linear system of equations to be solved. With direct methods, it appears to be difficult to leverage both the spatial and the temporal structure in (2).

The proposed PCG method is of the kind often used for large sparse problems [6]. For problem (2), the size of the linear system to solve at each Newton update is $O(NT)$. It is well-known that preconditioning (the P in PCG) can significantly reduce the number of steps required to terminate, which is bounded by the size of the problem [6, Thm. 10.7]; i.e., $O(NT)$. In this note, it is proposed to use a fixed number of block Jacobi iterations for preconditioning. In principle, this fixed number can be selected to achieve preconditioning specifications, in that a decreasing analytic bound on the conditioning of the outcome is provided. For the numerical examples presented, it is observed that as few as two Jacobi iterations can result in a much smaller number of PCG steps than the worst-case bound described above. Importantly, the preconditioning steps are decomposable across both the spatial and temporal dimension of (2). The size of the resulting $O(NT)$ parallelizable sub-problems is independent of N and T. As such, the arithmetic complexity of PCG steps is $O(NT)$. In the case of $O(NT)$ PCG steps, the arithmetic complexity of each Newton update becomes $O(N^2T^2)$. For $T \approx N$, the method is therefore no worse than the structured direct methods discussed, and potentially much better.

For separable-in-cost quadratic programs like (2), methods based on dual decomposition [17], and operator splitting, such as ADMM [18], [19] and FAMA [20], can also lead to parallelizable computations. For the structure in (2), the dual decomposition technique of [21] leads to local computations for each sub-system. Similarly, the ADMM approach presented in [22], [23], and projected sub-gradient algorithm of [24], also yield decomposable computations. However, these methods often require many thousands of iterations to converge, even for problems of moderate size, leading to high data-exchange overhead in distributed implementations. The issue is exacerbated within the path-graph context of this note, since the algebraic connectivity of the underlying sparsity pattern (which influences the rate of convergence [25], [26]) tends to zero as N grows. This motivates the consideration of the method proposed herein. The challenge is to maintain structure in the computations.

The note is organized as follows. An equivalent re-formulation of problem (2) is presented in Section II. The structure of the linear system of equations to solve for each Newton update is given in Section II-A. PCG methods are reviewed in Section III, and the structured preconditioner based on block Jacobi iterations is developed in Section IV. The proposed PCG algorithm is explored numerically for mass-spring-damper chain and irrigation channel examples in Section V. Concluding remarks are provided in Section VI.

NOTATION

Identity matrices are denoted by I, $\mathrm{blkdiag}(\cdot)$ denotes a matrix with block diagonal elements given by the arguments, which are the only non-zero elements, and $\mathrm{col}(\cdot)$ denotes the concatenation of the input arguments into a column vector. Every block tri-diagonal matrix is parameterized by sequences $\Phi = (\Phi_k)_{k=1}^m \in \prod_{k=1}^m \mathbb{R}^{l_k \times l_k}$ and $\Omega = (\Omega_k)_{k=2}^m \in \prod_{k=2}^m \mathbb{R}^{l_{k-1} \times l_k}$ for appropriate $(l_k)_{k=1}^m \subset \mathbb{N}^m$ and $m \in \mathbb{N}$. Given such sequences Φ and Ω, the corresponding block tri-diagonal matrix is denoted by

$$\mathrm{blktrid}(\Phi, \Omega) = \begin{bmatrix} \Phi_1 & \Omega_2' & & \\ \Omega_2 & \Phi_2 & \ddots & \\ & \ddots & \ddots & \Omega_m' \\ & & \Omega_m & \Phi_m \end{bmatrix} \in \mathbb{R}^{\bar{l} \times \bar{l}},$$

where $\bar{l} = \sum_{k=1}^m l_k$.

II. PROBLEM RE-FORMULATION

Defining $u_j = \mathrm{col}(u_{j,0}, \ldots, u_{j,T-1}) \in \mathbb{R}^{m_j T}$, $x_j = \mathrm{col}(x_{j,0}, \ldots, x_{j,T}) \in \mathbb{R}^{n_j(T+1)}$, and slack variables $\theta_j = \mathrm{col}(\theta_{j,0}, \ldots, \theta_{j,T}) \in \mathbb{R}^{\nu_j(T+1)}$, problem (2) can be reformulated as the following quadratic program:

$$\min_{\substack{(x_j)_{j\in\{0,N+1\}\cup\mathcal{N}} \\ (u_j)_{j\in\mathcal{N}}}} \frac{1}{2} \sum_{j\in\mathcal{N}} \begin{bmatrix} x_j \\ u_j \end{bmatrix}' \begin{bmatrix} Q_j & S_j' \\ S_j & R_j \end{bmatrix} \begin{bmatrix} x_j \\ u_j \end{bmatrix} + \begin{bmatrix} q_j \\ r_j \end{bmatrix}' \begin{bmatrix} x_j \\ u_j \end{bmatrix}, \tag{3a}$$

subject to $x_0 = \boldsymbol{\chi}$, $x_{N+1} = \boldsymbol{\zeta}$, and

$$0 = A_j x_j + B_j u_j + E_j x_{j-1} + F_j x_{j+1} + H_j \boldsymbol{\xi}_j, \quad j \in \mathcal{N}, \tag{3b}$$

$$0 = C_j x_j + D_j u_j + \theta_j - \boldsymbol{\kappa}_j, \quad j \in \mathcal{N}, \tag{3c}$$

$$0 \leq \theta_j, \quad j \in \mathcal{N}, \tag{3d}$$

where

$$\begin{aligned}
Q_j &= \mathrm{blkdiag}(Q_{j,0}, \ldots, Q_{j,T}) \in \mathbb{R}^{n_j(T+1) \times n_j(T+1)}, \\
R_j &= \mathrm{blkdiag}(R_{j,0}, \ldots, R_{j,T-1}) \in \mathbb{R}^{m_j T \times m_j T}, \\
S_j &= [\mathrm{blkdiag}(S_{j,0}, \ldots, S_{j,T-1})\ 0] \in \mathbb{R}^{m_j T \times n_j(T+1)}, \\
C_j &= \mathrm{blkdiag}(C_{j,0}, \ldots, C_{j,T}) \in \mathbb{R}^{\nu_j(T+1) \times n_j(T+1)}, \\
D_j &= [\mathrm{blkdiag}(D_{j,0}, \ldots, D_{j,T-1})'\ 0]' \in \mathbb{R}^{\nu_j(T+1) \times m_j T}, \\
H_j &= [I\ 0\ \cdots\ 0]' \in \mathbb{R}^{n_j(T+1) \times n_j}, \\
q_j &= \mathrm{blkdiag}(q_{j,0}, \ldots, q_{j,T}) \in \mathbb{R}^{n_j(T+1)}, \\
r_j &= \mathrm{blkdiag}(r_{j,0}, \ldots, r_{j,T-1}) \in \mathbb{R}^{m_j T}, \\
\boldsymbol{\kappa}_j &= \mathrm{col}(\boldsymbol{\kappa}_{j,0}, \ldots, \boldsymbol{\kappa}_{j,T}) \in \mathbb{R}^{\nu_j(T+1)}, \\
\boldsymbol{\chi} &= \mathrm{col}(\boldsymbol{\chi}_0, \ldots, \boldsymbol{\chi}_T) \in \mathbb{R}^{n_0(T+1)}, \\
\boldsymbol{\zeta} &= \mathrm{col}(\boldsymbol{\zeta}_0, \ldots, \boldsymbol{\zeta}_T) \in \mathbb{R}^{n_{N+1}(T+1)},
\end{aligned}$$

and

$$A_j = \begin{bmatrix} -I & & & \\ A_{j,0} & -I & & \\ & \ddots & \ddots & \\ & & A_{j,T-1} & -I \end{bmatrix}, \; B_j = \begin{bmatrix} 0 & \cdots & 0 \\ B_{j,0} & \ddots & \vdots \\ & \ddots & 0 \\ & & B_{j,T-1} \end{bmatrix}, \quad (4a)$$

$$E_j = \begin{bmatrix} 0 & & & \\ E_{j,0} & 0 & & \\ & \ddots & \ddots & \\ & & E_{j,T-1} & 0 \end{bmatrix}, \; F_j = \begin{bmatrix} 0 & & & \\ F_{j,0} & 0 & & \\ & \ddots & \ddots & \\ & & F_{j,T-1} & 0 \end{bmatrix}. \quad (4b)$$

Note that $A_j \in \mathbb{R}^{n_j(T+1)\times n_j(T+1)}$, $B_j \in \mathbb{R}^{n_j(T+1)\times m_j T}$, $E_j \in \mathbb{R}^{n_j(T+1)\times n_{j-1}(T+1)}$, and $F_j \in \mathbb{R}^{n_j(T+1)\times n_{j+1}(T+1)}$. The block bi-diagonal structure of the matrices A_j arises from the temporal structure of the system dynamics in the optimal control problem (2).

For the quadratic program (3), the Karush-Kuhn-Tucker (KKT) conditions for optimality are given by

$$Q_1 x_1 + S_1' u_1 + A_1' p_1 + C_1' \lambda_1 + E_2' p_2 + q_1 = 0, \quad (5a)$$

$$Q_j x_j + S_j' u_j + A_j' p_j + C_j' \lambda_j + F_{j-1}' p_{j-1} + E_{j+1}' p_{j+1} + q_j = 0, \quad j \in \mathcal{N}\backslash\{1, N\}, \quad (5b)$$

$$Q_N x_N + S_N' u_N + A_N' p_N + C_N' \lambda_N + F_{N-1}' p_{N-1} + q_N = 0, \quad (5c)$$

$$S_j x_j + R_j' u_j + B_j' p_j + D_j' \lambda_j + r_j = 0, \quad j \in \mathcal{N}, \quad (5d)$$

$$A_1 x_1 + B_1 u_1 + E_1 \boldsymbol{\chi} + F_1 x_2 + H_1 \boldsymbol{\xi}_1 = 0, \quad (5e)$$

$$A_j x_j + B_j u_j + E_j x_{j-1} + F_j x_{j+1} + H_j \boldsymbol{\xi}_j = 0, \quad j \in \mathcal{N}\backslash\{1, N\}, \quad (5f)$$

$$A_N x_N + B_N u_N + E_N x_{N-1} + F_N \boldsymbol{\zeta} + H_N \boldsymbol{\xi}_N = 0, \quad (5g)$$

$$C_j x_j + D_j u_j - \boldsymbol{\kappa}_j + \theta_j = 0, \quad j \in \mathcal{N}, \quad (5h)$$

$$\Lambda_j \Theta_j \mathbf{1} = 0, \quad \text{and} \quad [\lambda_j' \;\; \theta_j']' \geq 0, \quad j \in \mathcal{N}, \quad (5i)$$

where $p_j = \mathrm{col}(p_{j,0}, \ldots, p_{j,T}) \in \mathbb{R}^{n_j(T+1)}$ and $\lambda_j = \mathrm{col}(\lambda_{j,0}, \ldots, \lambda_{j,T}) \in \mathbb{R}^{\nu_j(T+1)}$ are Lagrange multipliers, $\Lambda_j = \mathrm{blkdiag}(\lambda_{j,0}, \ldots, \lambda_{j,T}) \in \mathbb{R}^{\nu_j(T+1)\times\nu_j(T+1)}$, $\Theta_j = \mathrm{blkdiag}(\theta_{j,0}, \ldots, \theta_{j,T}) \in \mathbb{R}^{\nu_j(T+1)\times\nu_j(T+1)}$, and $\mathbf{1}$ denotes a vector of all ones. Since (3) is convex, the KKT conditions are necessary and sufficient for optimality [7].

A. Newton's Method

Various second-order optimization algorithms can be understood in terms of Newton's method for solving the KKT conditions [7]. Typically, only a moderate number of Newton iterations is required for convergence, and this is the main advantage over first-order methods. The benefit comes from the use of second-order information, which can be constructed explicitly for quadratic programs. For the problem (3), the Newton updates in an interior point method take the form of

$$s^{(n+1)} = s^{(n)} + \alpha^{(n)} \delta^{(n)}, \quad (6)$$

where $\alpha^{(n)} > 0$ is a step size, $s^{(n)} = \mathrm{col}(s_1^{(n)}, \ldots, s_N^{(n)})$, $s_j^{(n)} = \mathrm{col}(x_j^{(n)}, \; u_j^{(n)}, \; p_j^{(n)}, \; \lambda_j^{(n)}, \; \theta_j^{(n)})$, and the second-order search direction $\delta^{(n)} = \mathrm{col}(\delta_1^{(n)}, \ldots, \delta_N^{(n)})$ is obtained by solving the linearized KKT conditions, given by

$$\mathrm{blktrid}(\Phi^{(n)}, \Omega)\, \delta^{(n)} = b^{(n)}, \quad (7)$$

with $\Phi^{(n)} = (\Phi_j^{(n)})_{j\in\mathcal{N}}$, $\Omega = (\Omega_j)_{j\in\mathcal{N}\backslash\{1\}}$, $b^{(n)} = \mathrm{col}(b_1^{(n)}, \ldots, b_N^{(n)})$,

$$\Phi_j^{(n)} = \begin{bmatrix} Q_j & S_j' & A_j' & C_j' & 0 \\ S_j & R_j & B_j' & D_j' & 0 \\ A_j & B_j & 0 & 0 & 0 \\ C_j & D_j & 0 & 0 & I \\ 0 & 0 & 0 & \Theta_j^{(n)} & \Lambda_j^{(n)} \end{bmatrix}, \; j \in \mathcal{N}, \quad (8a)$$

$$\Omega_j = \begin{bmatrix} 0 & 0 & F_{j-1}' & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ E_j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; j \in \mathcal{N}\backslash\{1\}, \quad (8b)$$

$$b_1^{(n)} = \mathrm{col}(-q_1, -r_1, -H_1\boldsymbol{\xi}_1 - E_1\boldsymbol{\chi}, \boldsymbol{\kappa}_1, \eta_1^{(n)}) - \Phi_1^{(n)} s_1^{(n)} - \Omega_2' s_2^{(n)}, \quad (8c)$$

$$b_N^{(n)} = \mathrm{col}(-q_N, -r_N, -H_N\boldsymbol{\xi}_N - F_N\boldsymbol{\zeta}, \boldsymbol{\kappa}_N, \eta_N^{(n)}) - \Omega_N s_{N-1}^{(n)} - \Phi_N^{(n)} s_N^{(n)}, \quad (8d)$$

$$b_j^{(n)} = \mathrm{col}(-q_j, -r_j, -H_j\boldsymbol{\xi}_j, \boldsymbol{\kappa}_j, \eta_j^{(n)}) - \Omega_j s_{j-1}^{(n)} - \Phi_j^{(n)} s_j^{(n)} - \Omega_{j+1}' s_{j+1}^{(n)}, \; j \in \mathcal{N}\backslash\{1,N\}, \quad (8e)$$

$$\eta_j^{(n)} = \Theta_j^{(n)}\lambda_j^{(n)} + \Lambda_j^{(n)}\theta_j^{(n)} - \Lambda_j^{(n)}\Theta_j^{(n)}\mathbf{1} + \sigma^{(n)}\mu^{(n)}\mathbf{1}, \; j \in \mathcal{N}. \quad (8f)$$

In (8f), $\mu^{(n)} = \sum_{j=1}^N ((\lambda_j^{(n)})'\theta_j^{(n)}) / \sum_{j=1}^N (\nu_j(T+1))$ is a measure of the duality gap and $\sigma^{(n)} \in (0,1)$ is a centering parameter [7]. The step-size $\alpha^{(n)} > 0$ in (6) is selected online to ensure the components of $\lambda_j^{(n+1)}$ and $\theta_j^{(n+1)}$ remain positive for $j \in \mathcal{N}$. The coefficient matrix $\mathrm{blktrid}(\Phi^{(n)}, \Omega)$ in (7) is non-singular because A_j in (4a) is non-singular for all $j \in \mathcal{N}$ [15, Lem. A.1]. When $Q_j \succ 0$ and $Q_{j,t} - S_{j,t}' R_{j,t}^{-1} S_{j,t} \succ 0$ for $j \in \mathcal{N}$, it can be shown that $\mathrm{blktrid}(\Phi^{(n)}, \Omega)$ is non-singular without recourse to the invertibility of A_j. In this case, it is possible to accommodate more general sub-system dynamics of the form

$$G_{j,t} x_{j,t+1} = A_{j,t} x_{j,t} + B_{j,t} u_{j,t} + E_{j,t} x_{j-1,t} + F_{j,t} x_{j+1,t},$$

with singular matrices $G_{j,t}$ (which would make A_j singular), in place of (1), as may arise from implicit discretization.

B. Structure-Preserving Block Elimination

Λ_j, Θ_j and R_j in (8a) are block diagonal, with block sizes that are independent of N and T. For $j \in \mathcal{N}$, let $\delta_j^{(n)} = \mathrm{col}(\delta_{x_j}^{(n)}, \delta_{u_j}^{(n)}, \delta_{p_j}^{(n)}, \delta_{\lambda_j}^{(n)}, \delta_{\theta_j}^{(n)})$ and $b_j^{(n)} = \mathrm{col}(b_{x_j}^{(n)}, b_{u_j}^{(n)}, b_{p_j}^{(n)}, b_{\lambda_j}^{(n)}, b_{\theta_j}^{(n)})$ be partitions aligned with the structure of $s_j^{(n)}$ noted below (6). Dropping the Newton iteration index (n), the ordered elimination of

$$\delta_{\theta_j} = \Lambda_j^{-1}(b_{\theta_j} - \Theta_j \delta_{\lambda_j}), \quad (9)$$

$$\delta_{\lambda_j} = -(\Theta_j^{-1}\Lambda_j)(b_{\lambda_j} - C_j\delta_{x_j} - D_j\delta_{u_j} - \Lambda_j^{-1} b_{\theta_j}), \quad (10)$$

$$\delta_{u_j} = \hat{R}_j^{-1}(\hat{b}_{u_j} - \hat{S}_j\delta_{x_j} - B_j'\delta_{p_j}), \quad (11)$$

from (7), for $j \in \mathcal{N}$, yields the smaller symmetric system

$$\mathrm{blktrid}(\tilde{\Phi}, \tilde{\Omega})\, \tilde{\delta} = \tilde{b}, \quad (12)$$

where $\tilde{\Phi} = (\tilde{\Phi}_j)_{j\in\mathcal{N}}$, $\tilde{\Omega} = (\tilde{\Omega}_j)_{j\in\mathcal{N}\setminus\{1\}}$, $\tilde{\delta} = \mathrm{col}(\tilde{\delta}_1, \ldots, \tilde{\delta}_N)$, $\tilde{b} = \mathrm{col}(\tilde{b}_1, \ldots, \tilde{b}_N)$, and $\tilde{\delta}_j = \mathrm{col}(\delta_{x_j}, \delta_{p_j}) \in \mathbb{R}^{2n_j(T+1)}$, $\tilde{b}_j = \mathrm{col}(\tilde{b}_{x_j}, \tilde{b}_{p_j}) \in \mathbb{R}^{2n_j(T+1)}$,

$$\tilde{\Phi}_j = \begin{bmatrix} \tilde{Q}_j & \tilde{A}_j' \\ \tilde{A}_j & \tilde{R}_j \end{bmatrix} \in \mathbb{R}^{2n_j(T+1)\times 2n_j(T+1)}, \quad (13a)$$

$$\tilde{\Omega}_j = \begin{bmatrix} 0 & F_{j-1}' \\ E_j & 0 \end{bmatrix} \in \mathbb{R}^{2n_j(T+1)\times 2n_j(T+1)}, \quad (13b)$$

$$\begin{bmatrix} \tilde{Q}_j & \tilde{A}_j' \\ \tilde{A}_j & \tilde{R}_j \end{bmatrix} = \begin{bmatrix} \hat{Q}_j & A_j' \\ A_j & 0 \end{bmatrix} - \begin{bmatrix} \hat{S}_j' \\ B_j \end{bmatrix} \hat{R}_j^{-1} \begin{bmatrix} \hat{S}_j' \\ B_j \end{bmatrix}', \quad (13c)$$

$$\begin{bmatrix} \hat{Q}_j & \hat{S}_j' \\ \hat{S}_j & \hat{R}_j \end{bmatrix} = \begin{bmatrix} Q_j & S_j' \\ S_j & R_j \end{bmatrix} + \begin{bmatrix} C_j' \\ D_j' \end{bmatrix} (\Theta_j^{-1}\Lambda_j) \begin{bmatrix} C_j' \\ D_j' \end{bmatrix}', \quad (13d)$$

$$\begin{bmatrix} \tilde{b}_{x_j} \\ \tilde{b}_{p_j} \end{bmatrix} = \begin{bmatrix} \hat{b}_{x_j} \\ \hat{b}_{p_j} \end{bmatrix} - \begin{bmatrix} \hat{S}_j' \\ B_j \end{bmatrix} \hat{R}_j^{-1} \hat{b}_{u_j}, \quad (13e)$$

$$\begin{bmatrix} \hat{b}_{x_j} \\ \hat{b}_{u_j} \end{bmatrix} = \begin{bmatrix} b_{x_j} \\ b_{u_j} \end{bmatrix} + \begin{bmatrix} C_j' & C_j' \\ D_j' & D_j' \end{bmatrix} \begin{bmatrix} (\Theta_j^{-1}\Lambda_j) b_{\lambda_j} \\ \Theta_j^{-1} b_{\theta_j} \end{bmatrix}, \quad (13f)$$

for $j \in \mathcal{N}$. The structure of (7) is preserved in (12), and in (13a), $\tilde{Q}_j$ and $\tilde{R}_j$ are block diagonal, and $\tilde{A}_j$ is block bi-diagonal. Note that the decomposable computations required to form (13c)–(13f) are performed only once per Newton iteration. The arithmetic complexity is $O(NT(\bar{m}^3 + \bar{\nu}^3))$, where $\bar{m} = \max_j(m_j)$ and $\bar{\nu} = \max_j(\nu_j)$.

A PCG based method is developed for (12) next. The number of steps required depends on the quality of the preconditioner used. Assuming perfect arithmetic, the maximum number of steps of the PCG method is the size of the problem [6, Thm. 10.7], which is $O(NT)$ in (12). As such, the arithmetic complexity of the proposed approach is $O(N^2T^2)$ at worst, since the arithmetic complexity of each PCG step is shown to be $O(NT)$. When $T \approx N$, this is comparable to the previously discussed direct methods. However, good preconditioning can substantially reduce the number of PCG steps. Properties of an effective and structured preconditioner are given in Section IV. This is the main contribution.

III. PCG Solvers

The conjugate gradient (CG) method is used for linear systems with positive-definite coefficient matrix [27]. While non-singular, the block tri-diagonal matrix $\mathrm{blktrid}(\tilde{\Phi}, \tilde{\Omega})$ in (12) has both positive and negative eigenvalues. This indefinite system can be solved using other Krylov methods, like MINRES [28] or GMRES [29]. However, the computations for these are more involved than the CG method, with reduced scope for decomposability of the computations. Multiplying both sides of (12) by $\mathrm{blktrid}(\tilde{\Phi}, \tilde{\Omega})$ from the left yields the positive-definite system of equations

$$\Psi\breve{\delta} = \breve{b}, \quad (14)$$

where $\Psi = (\mathrm{blktrid}(\tilde{\Phi}, \tilde{\Omega}))^2$ and $\breve{b} = \mathrm{blktrid}(\tilde{\Phi}, \tilde{\Omega})\,\tilde{b}$. The positive-definite matrix Ψ is now block penta-diagonal, but (14) in now amenable to the CG method.

Let $e^{(i)} = \tilde{\delta}^{(i)} - \tilde{\delta}^*$ be the error between i-th iterate $\tilde{\delta}^{(i)}$ of the CG method and the exact solution $\tilde{\delta}^*$ of (14). It can be shown that $e^{(i)}$ satisfies the following [30, Thm. 6.29]:

$$\|e^{(i)}\|_\Psi \leq 2\left((\sqrt{\kappa(\Psi)} - 1)/(\sqrt{\kappa(\Psi)} + 1)\right)^i \|e^{(0)}\|_\Psi, \quad (15)$$

Algorithm 1 PCG [6] for (14) with preconditioner P.

1: **initialize** $\tilde{\delta}^{(0)}$, ϵ, $\mathrm{iter}_{\max}$
2: $r^{(0)} = \breve{b} - \Psi\tilde{\delta}^{(0)}$
3: **solve** $Pd^{(0)} = r^{(0)}$
4: $\beta^{(0)} = (d^{(0)})'r^{(0)}$
5: $i = 0$
6: **while** $i < \mathrm{iter}_{\max}$ **do**
7: $\quad y^{(i)} = \Psi d^{(i)}$
8: $\quad \gamma^{(i)} = \beta^{(i)}/((y^{(i)})'d^{(i)})$
9: $\quad \tilde{\delta}^{(i+1)} = \tilde{\delta}^{(i)} + \gamma^{(i)}d^{(i)}$
10: $\quad r^{(i+1)} = r^{(i)} - \gamma^{(i)}y^{(i)}$
11: $\quad$ **if** $\|r^{(i+1)}\|_\infty < \epsilon$ **exit**
12: $\quad$ **solve** $Pq^{(i+1)} = r^{(i+1)}$
13: $\quad \beta^{(i+1)} = (q^{(i+1)})'r^{(i+1)}$
14: $\quad d^{(i+1)} = r^{(i+1)} + \left(\beta^{(i+1)}/\beta^{(i)}\right) d^{(i)}$
15: $\quad i = i + 1$
16: **end while**

where $\|e\|_\Psi = e'\Psi e$, $\kappa(\Psi) = \lambda_{\max}(\Psi)/\lambda_{\min}(\Psi)$ is the condition number, and $\lambda_{\max}(\Psi)$ (resp. $\lambda_{\min}(\Psi)$) is the maximum (resp. minimum) eigenvalue of Ψ. The CG method converges faster for $\kappa(\Psi)$ closer to 1. To improve performance, CG can be applied to the transformed problem

$$P^{-1/2}\Psi P^{-1/2}\breve{\delta} = P^{-1/2}\breve{b}, \quad (16)$$

where $\breve{\delta} = P^{1/2}\tilde{\delta}$ for $P = P' \succ 0$. Algorithm 1 is an efficient implementation of this preconditioned CG (i.e., PCG) method taken from [6]. Lines 7 to 14 constitute one PCG step.

The preconditioner $P = \Psi$ gives $P^{-1/2}\Psi P^{-1/2} = I$. But lines 3 and 12 of Algorithm 1 then become the original problem. Incomplete sparse LU factorization of Ψ can be used for P instead. Such preconditioners are considered in [31], [32]. However, for the resulting preconditioner to be positive definite and effective, it may be necessary to use incomplete LU factors that are denser (i.e., have less structure) than Ψ.

In the next section, a structured preconditioning approach is developed, which involves a fixed number of block Jacobi iterations (e.g., [6]) to *approximately* implement lines 3 and 12 with $P = \Psi$. This approach builds on ideas from [33]–[35].

IV. Block Jacobi Preconditioning

Let $\mathcal{K} = \{1, 2, \ldots, K\}$, with $K = \lceil N/2 \rceil$, i.e., $K = N/2$ when N is even, and $K = (N+1)/2$ otherwise. Also define

$$\Delta_k = \begin{bmatrix} Z_{2k-1} & Y_{2k}' \\ Y_{2k} & Z_{2k} \end{bmatrix}, \ k \in \mathcal{K}\setminus\{K\}, \quad (17a)$$

$$\Delta_K = \begin{cases} \begin{bmatrix} Z_{N-1} & Y_N' \\ Y_N & Z_N \end{bmatrix}, & N \text{ even}, \\ Z_N, & N \text{ odd}, \end{cases} \quad (17b)$$

$$\Upsilon_k = \begin{bmatrix} V_{2k-1} & Y_{2k-1} \\ 0 & V_{2k} \end{bmatrix}, \ k \in \mathcal{K}\setminus\{1, K\}, \quad (17c)$$

$$\Upsilon_K = \begin{cases} \begin{bmatrix} V_{N-1} & Y_{N-1} \\ 0 & V_N \end{bmatrix}, & N \text{ even}, \\ \begin{bmatrix} V_N & Y_N \end{bmatrix}, & N \text{ odd}, \end{cases} \quad (17d)$$

where referring to (13),

$$Z_j = \tilde{\Phi}_j^2 + \tilde{\Omega}_j\tilde{\Omega}_j' + \tilde{\Omega}_{j+1}'\tilde{\Omega}_{j+1}, \tag{18a}$$
$$Y_j = \tilde{\Omega}_j\tilde{\Phi}_{j-1} + \tilde{\Phi}_j\tilde{\Omega}_j, \tag{18b}$$
$$V_j = \tilde{\Omega}_j\tilde{\Omega}_{j-1}, \tag{18c}$$

for $j \in \mathcal{N}$, with $\tilde{\Omega}_{N+1} = 0$. Given this, $\Psi = \mathrm{blktrid}(\mathit{\Delta}, \Upsilon)$, where $\mathit{\Delta} = (\Delta_k)_{k\in\mathcal{K}}$ and $\Upsilon = (\Upsilon_k)_{k\in\mathcal{K}\setminus\{1\}}$. Moreover, lines 3 and 12 with $P = \Psi$ take the form of

$$\Psi\zeta = \mathrm{blktrid}(\mathit{\Delta}, \Upsilon)\,\zeta = \tau. \tag{19}$$

Let $\Delta = \mathrm{blkdiag}(\Delta_1, \ \ldots, \ \Delta_K)$ and $\Sigma = \Delta - \Psi$. The block Jacobi method for solving (19) involves the following:

$$\Delta\zeta^{(l+1)} = \tau + \Sigma\zeta^{(l)}. \tag{20}$$

These iterations converge to the solution of (19) if and only if

$$\varrho(\Delta^{-1}\Sigma) < 1, \tag{21}$$

where $\varrho(\cdot)$ denotes spectral radius [6, Thm. 2.16]. For $\Psi = \mathrm{blktrid}(\mathit{\Delta}, \Upsilon) \succ 0$, and the splitting $\Psi = \Delta - \Sigma$, condition (21) always holds [6, Lem. 4.7 with Thm. 4.18].

The proposal here is to apply a fixed number $L > 0$ of Jacobi iterations (20) to approximately implement lines 3 and 12 in Algorithm 1 with $P = \Psi$. Characteristics of this approach are discussed in the next three sub-sections.

A. Positive definiteness of the preconditioner

Executing a fixed number of block Jacobi steps from zero is equivalent to the use of a positive-definite preconditioner.

Theorem IV.1: Given $L \in \mathbb{N}$ and $\zeta^{(0)} = 0$, the L-th iterate of (20) satisfies $P_L\zeta^{(L)} = \tau$ with $P_L = W_L^{-1}$, where $W_L = \sum_{l=0}^{L-1}(\Delta^{-1}\Sigma)^l\Delta^{-1} \succ 0$.

Proof: Noting that $\Delta \succ 0$ is invertible, it follows from (20) that $\zeta^{(L)} = W_L\tau + (\Delta^{-1}\Sigma)^L\zeta^{(0)} = W_L\tau$. It is established below that W_L is positive definite, and thus, invertible. As such, $P_L\zeta^{(L)} = W_L^{-1}\zeta^{(L)} = \tau$.

Positive definiteness of W_L is a consequence of the known property $\Psi = \mathrm{blktrid}(\Delta, \Upsilon) \succ 0$. With $U = \mathrm{blkdiag}(U_K, ..., U_1)$, and $U_k = (-I)^k$ for $k = 1, \ldots, K$, first note that $2\Delta - \Psi = U'\Psi U \succ 0$. Then note that $W_1 = \Delta^{-1} \succ 0$, and using $(\Delta^{-1}\Sigma)^l\Delta^{-1} = \Delta^{-1}(\Sigma\Delta^{-1})^l$, that

$$\begin{aligned} W_{2M} &= \sum_{l=0}^{M-1}(\Delta^{-1}\Sigma)^l\Delta^{-1}(\Delta+\Sigma)\Delta^{-1}(\Sigma\Delta^{-1})^l \\ &= \sum_{l=1}^{M-1}(\Delta^{-1}\Sigma)^l\Delta^{-1}(2\Delta-\Psi)\Delta^{-1}(\Sigma\Delta^{-1})^l \\ &\quad + \Delta^{-1}(2\Delta-\Psi)\Delta^{-1} \succ 0, \end{aligned}$$

and

$$\begin{aligned} W_{2M+1} &= \sum_{l=0}^{2M-1}(\Delta^{-1}\Sigma)^l\Delta^{-1} + (\Delta^{-1}\Sigma)^{2M}\Delta^{-1} \\ &= \sum_{l=0}^{M-1}(\Delta^{-1}\Sigma)^l\Delta^{-1}(\Delta+\Sigma)\Delta^{-1}(\Sigma\Delta^{-1})^i \\ &\quad + (\Delta^{-1}\Sigma)^M\Delta^{-1}((\Delta^{-1}\Sigma)^M)' \succ 0, \end{aligned}$$

for $M \in \mathbb{N}$. Therefore, $W_L \succ 0$, as claimed. ∎

B. An analytic bound on achieved conditioning

Theorem IV.2: With $P_L = (\sum_{l=0}^{L-1}(\Delta^{-1}\Sigma)^l\Delta^{-1})^{-1}$ for given $L \in \mathbb{N}$,

$$\kappa(P_L^{-1/2}\Psi P_L^{-1/2}) \le \frac{1+(\varrho(\Delta^{-1}\Sigma))^L}{1-(\varrho(\Delta^{-1}\Sigma))^L}. \tag{22}$$

Proof: By Theorem IV.1, $P_L \succ 0$. Using $\Psi = \Delta - \Sigma$,

$$P_L^{-1}\Psi = \sum_{l=0}^{L-1}(\Delta^{-1}\Sigma)^l(I - \Delta^{-1}\Sigma) = I - (\Delta^{-1}\Sigma)^L. \tag{23}$$

Furthermore, $P_L^{-1}\Psi = P_L^{-1/2}(P_L^{-1/2}\Psi P_L^{-1/2})P_L^{1/2}$, whereby $\mathrm{spec}(P_L^{-1/2}\Psi P_L^{-1/2}) = \mathrm{spec}(P_L^{-1}\Psi)$. As noted below (21), $\varrho(\Delta^{-1}\Sigma) < 1$ since $\Psi \succ 0$. So (23) holds because $\lambda_{\max}(I - (\Delta^{-1}\Sigma)^L) \le 1 + (\varrho(\Delta^{-1}\Sigma))^L$ and $\lambda_{\min}(I - (\Delta^{-1}\Sigma)^L) \ge 1 - (\varrho(\Delta^{-1}\Sigma))^L > 0$. ∎

By Theorem IV.2, the number L of block Jacobi iterations can be selected to achieve desired preconditioning.

C. Decomposable computations

Note that explicit construction of the preconditioner P_L is not needed. At each PCG step, L iterations of (20) are performed from $\zeta^{(0)} = 0$. Since Δ is block diagonal, the computations required to implement each Jacobi iteration can be decomposed into $K = \lceil N/2 \rceil$ smaller problems

$$\Delta_k\zeta_k^{(l+1)} = \omega_k, \tag{24}$$

where $\omega_k = \tau_k + \Upsilon_k\zeta_{k-1}^{(l)} + \Upsilon_{k+1}'\zeta_{k+1}^{(l)}$ for $k \in \mathcal{K}$, with $\Upsilon_{K+1} = 0$. Each Δ_k is a block 2×2 matrix, with inner blocks that are structured. To see this structure, consider

$$\Delta_k = \begin{bmatrix} Z_{2k-1} & Y_{2k}' \\ Y_{2k} & Z_{2k} \end{bmatrix}. \tag{25}$$

Note that

$$\begin{aligned} Z_j &= \tilde{\Phi}_j^2 + \tilde{\Omega}_j\tilde{\Omega}_j' + \tilde{\Omega}_{j+1}'\tilde{\Omega}_{j+1} \\ &= \begin{bmatrix} \tilde{Q}_j^2 + \tilde{A}_j'\tilde{A}_j + F_{j-1}F_{j-1}' + E_j'E_j & \tilde{Q}_j\tilde{A}_j' + \tilde{A}_j'\tilde{R}_j \\ \tilde{A}_j\tilde{Q}_j + \tilde{R}_j\tilde{A}_j & \tilde{A}_j\tilde{A}_j' + \tilde{R}_j^2 + F_jF_j' + E_{j+1}'E_{j+1} \end{bmatrix}, \end{aligned} \tag{26a}$$

$$Y_j = \tilde{\Omega}_j\tilde{\Phi}_{j-1} + \tilde{\Phi}_j\tilde{\Omega}_j = \begin{bmatrix} F_{j-1}'\tilde{A}_{j-1} + \tilde{A}_j'E_j & F_{j-1}'\tilde{R}_{j-1} + \tilde{Q}_jF_{j-1}' \\ E_j\tilde{Q}_{j-1} + \tilde{R}_{j-1}E_j & E_j\tilde{A}_{j-1}' + \tilde{A}_jF_{j-1}' \end{bmatrix}. \tag{26b}$$

The diagonal blocks of Z_j are block tri-diagonal, while off-diagonal blocks are block bi-diagonal for $j \in \mathcal{N}$. Similarly, the diagonal blocks of Y_j are block tri-diagonal, and the off-diagonal blocks are block diagonal for $j \in \mathcal{N}\setminus\{1\}$. As such, a permutation of variables exists such that (24) takes the form

$$\mathrm{blktrid}(\Xi_k, \mathit{\Pi}_k)\,\hat{\zeta}_k^{(l+1)} = \hat{\omega}_k, \tag{27}$$

where $\hat{\omega}_k = \hat{\tau}_k + \hat{\Upsilon}_k\hat{\zeta}_{k-1}^{(i)} + \hat{\Upsilon}_{k+1}'\hat{\zeta}_{k+1}^{(i)}$, $\Xi_k = (\Xi_{k,t})_{t\in\mathcal{T}}$, $\mathit{\Pi}_k = (\mathit{\Pi}_{k,t})_{t\in\mathcal{T}\setminus\{T\}}$,

$$\Xi_{k,t} = \begin{bmatrix} \check{Q}_{2k-1,t} & \mathit{\Omega}_{2k-1,t} & \check{E}_{2k,t}' & 0 \\ \mathit{\Omega}_{2k-1,t} & \check{R}_{2k-1,t-1} & 0 & \check{F}_{2k,t-1}' \\ \check{E}_{2k,t} & 0 & \check{Q}_{2k,t} & \mathit{\Omega}_{2k,t} \\ 0 & \check{F}_{2k,t-1} & \mathit{\Omega}_{2k,t} & \check{R}_{2k,t-1} \end{bmatrix}, \tag{28a}$$

	Single Thread	N/2 Parallel Threads	
Alg. 1	**Computations**	**Computations per thread**	**Data xchg. per thread**
Line 7:	$O(NT\bar{n}^2)$	$O(T\bar{n}^2)$	$O(T\bar{n})$
Line 8:	$O(NT\bar{n})$	$O(T\bar{n})$	$O(1)$
Line 9:	$O(NT\bar{n})$	$O(T\bar{n})$	0
Line 10:	$O(NT\bar{n})$	$O(T\bar{n})$	0
Line 11:	$O(NT\bar{n})$	$O(T\bar{n})$	$O(1)$
Line 12:	$O(LT\bar{n}^3)$	$O(LT\bar{n}^3)$	$O(LT\bar{n})$
Line 13:	$O(NT\bar{n})$	$O(T\bar{n})$	$O(1)$
Line 14:	$O(NT\bar{n})$	$O(T\bar{n})$	0

TABLE I
ARITHMETIC COMPLEXITY AND DATA-EXCHANGE OVERHEAD OF ALGORITHM 1 MAIN LOOP: $\bar{n} = \max_j(n_j)$, WHERE n_j IS THE SIZE OF $x_{j,t}$; AND L IS THE FIXED NUMBER OF JACOBI ITERATIONS.

with

$$\check{Q}_{j,t} = \tilde{Q}^2_{j,t} + I + \tilde{A}'_{j,t}\tilde{A}_{j,t} + E'_{j,t}E_{j,t} + F'_{j-1,t}F_{j-1,t}, \quad (28b)$$
$$\check{R}_{j,t} = \tilde{R}^2_{j,t} + I + \tilde{A}_{j,t}\tilde{A}'_{j,t} + E_{j,t}E'_{j,t} + F_{j,t}F'_{j,t}, \quad (28c)$$
$$\Omega_{j,t} = -\tilde{Q}_{j,t} - \tilde{R}_{j,t-1}, \quad (28d)$$
$$\check{E}_{j,t} = \tilde{A}'_{j,t}E_{j,t} + F'_{j-1,t}\tilde{A}_{j-1,t}, \quad (28e)$$
$$\check{F}_{j,t} = \tilde{A}_{j,t}F'_{j-1,t} + E_{j,t}\tilde{A}'_{j-1,t}, \quad (28f)$$

for $j \in \mathcal{N}$, and

$$\Pi_{k,t} = \begin{bmatrix} -\tilde{A}_{2k-1,t} & 0 & -F'_{2k-1,t} & 0 \\ \check{A}_{2k-1,t} & -\tilde{A}_{2k-1,t} & G_{2k-1,t} & -F'_{2k-1,t} \\ -E_{2k,t} & 0 & -\tilde{A}_{2k,t} & 0 \\ X_{2k,t} & -E_{2k,t} & \check{A}_{2k,t} & -\tilde{A}_{2k,t} \end{bmatrix} \quad (29a)$$

with

$$\check{A}_{j,t} = \tilde{A}_{j,t}\tilde{Q}_{j,t} + \tilde{R}_{j,t}\tilde{A}_{j,t}, \quad j \in \mathcal{N} \quad (29b)$$
$$G_{j,t} = F'_{j-1,t}\tilde{R}_{j-1,t} + \tilde{Q}_{j,t}F'_{j-1,t}, \quad j \in \mathcal{N}\backslash\{1\} \quad (29c)$$
$$X_{j,t} = E_{j,t}\tilde{Q}_{j-1,t} + \tilde{R}_{j-1,t}E_{j,t}, \quad j \in \mathcal{N}\backslash\{1\}. \quad (29d)$$

Noting that $\Xi_{k,t}, \Pi_{k,t} \in \mathbb{R}^{\hat{n}_{k,t}\times\hat{n}_{k,t}}$, where $\hat{n}_{k,t} = 2(n_{2k-1} + n_{2k})$ for $k \in \mathcal{K}$ and $t \in \mathcal{T}$, the sub-blocks of $\mathrm{blktrid}(\Xi_k, \Pi_k)$ are of size independent of N and T.

For each $k \in \mathcal{K}$, the block tri-diagonal system (27) can be solved with arithmetic complexity $O(T)$, by structured (backward-forward) recursions that implement an LDL factorization based direct method [36]. So the preconditioning lines 3 and 12 of Algorithm 1 decompose into $K = \lceil N/2 \rceil$ parallel threads, each comprising computations for $2T$ sequential (possibly dense) problems of size independent of N and T. The data exchange per PCG step is localized in line with the path-graph structure of the system dynamics in (1). For such an implementation, Table I shows the arithmetic complexity of each line of Algorithm 1, overall and per-thread, in addition to the scalar-value data-exchange overhead for each thread.

The per PCG step arithmetic complexity is dominated by line 12, i.e., $O(LNT\bar{n}^3)$. With the number L of block Jacobi preconditioning iterations fixed, and the bound $\bar{n}$ on the size of sub-system states fixed, the overall arithmetic complexity of PCG each step is $O(NT)$. The computations in lines 7, 8, and 13 also decompose in a fashion that is consistent with the block partitioning of Ψ in (27), and hence, the aforementioned $\lceil N/2 \rceil$ parallel threads with localized path-graph data exchange. Note that lines 8, 11 and 13 involve sequential computations to form dot-products and to test the stopping condition. These can be carried out by a backward-forward sweep with path-graph data exchange. Lines 7 and 12 involve the exchange of less than $T\bar{n}$ scalars between the neighbouring threads on this path-graph, since the partition of Ψ is block tri-diagonal. The overall scalar data-exchange overhead is $O(LNT\bar{n})$ per PCG step.

V. NUMERICAL RESULTS

Results of numerical experiments are presented for two optimal control problems of the form (2).

Case (I): A one-dimensional mass-spring-damper chain of $N > 0$ masses, taken from [37]. Each sub-system $j \in \mathcal{N}$ has dynamics of the form (1) with $n_j = 2$, $m_j = 1$, and $\nu_j = 6$. The corresponding cost has $Q_{j,t} = \mathrm{diag}(1, 0)$ and $R_{j,t} = 1$ for $t \in \mathcal{T}$. The mass, spring constant, damping coefficient parameters are selected randomly between 0.8 to 1.5 to generate a heterogeneous network. Experiments are done with $N = T$, varied from 10 to 1000. The number of scalar variables in the largest problem is of the order 10^7, and there are a similar number of constraints.

Case (II): An automated irrigation channel with N controlled pools [39]. Each pool has one control input, the water-level reference, and four states, two for the water-level dynamics and two for a PI feedback controller that sets the upstream inflow on the basis of the measured downstream water-level error. The dynamics of each controlled pool can be described by a model of the form (1). For each sub-system $j \in \mathcal{N}$, $n_j = 4$, $m_j = 1$ and $\nu_j = 6$, the matrix $F_{j,t} = 0$, and the cost is set such that $Q_{j,t} = C^\top_{j,t}C_{j,t}$ and $R_{j,t} = 1$. The model data are all borrowed from [40]. Experiments are done for $N = T$, varied from 10 to 500. The number of scalar variables in the largest problem is of the order 10^6, and the number of and inequality constraints is similar.

For both cases, the Newton updates in the interior-point method of Section II-A are computed in four different ways:

- Via the proposed PCG method to solve (19), with $L = 2$ block Jacobi preconditioning steps;
- Via block Jacobi iterations (20) to solve (19);
- Via the direct method from [15], which solves (12) by recursive LDL factorization;
- Via MATLAB's backslash (MA-57 [38]) to solve (12).

Results for backslash are provided as a base line, noting that backslash is able to permute variables in ways that do not respect the spatio-temporal structure of problem (2), unlike the proposed PCG method, which decomposes in accordance with this structure. To gauge overall arithmetic complexity, computation time of a single thread implementation is considered for all methods. The stopping criterion for the infinity norm of the residuals in Algorithm 1, and in the pure block Jacobi iterations based implementation, is set to $\epsilon = 10^{-9}$.

The duality-gap based stopping criterion for the interior point method is set to $\epsilon_{\mathrm{IPM}} = 10^{-6}$. For all experiments, this was achieved in 15 to 20 Newton updates.

Case (I): Mass-spring-damper chain. Fig. 1 shows the maximum/average number of iterations for the PCG method with $L = 2$, and the pure block Jacobi method, across all Newton updates. The pure block Jacobi method consistently involves a large number of iterates (20), in the order of thousands. By contrast, the proposed PCG method consistently involves fewer steps, in the order of hundreds. The corresponding reduction in overall data exchange overhead could be advantageous in a distributed implementation.

Fig. 2 shows the normalized average processor time for a single thread implementation as proxy for the arithmetic complexity per Newton update. Along the line $N = T$, the average time is $O(N^2)$ for the PCG method, compared to $O(N^4)$ for the direct method of [15]. While the Jacobi method is also $O(N^2)$, the time is an order of magnitude greater than the PCG method, which is comparable to MATLAB's backslash, suggesting little loss for the gain in terms of scope for decomposability and distributed implementation.

Fig. 3 shows the maximum condition number of Ψ in (14), and maximum bound for $\kappa(P_L^{-1/2}\Psi P_L^{-1/2})$ in (22) with $L = 2$, across all Newton updates. The approach improves the conditioning by more than two orders of magnitude.

Case (II): Automated irrigation channel. Fig. 4 shows the maximum/average number of iterations for the pure block Jacobi method, and the PCG method with $L = 2$, taken across Newton updates. The total number of iterations of pure Jacobi method increases sharply with the size of the problem due to ill-conditioning along the line $N = T$. By contrast, the number of PCG steps is $O(N^2)$, but remains much smaller than the size of the problem. The total number of variables with $N = T = 100$ is 40800 for system (19), while the maximum number of PCG steps is 45.

Fig. 5 shows the normalized processor time, averaged across all Newton updates. The normalized processor time for all methods is $O(N^4)$, except for the Jacobi method which is worse than that. For the PCG method, this arises because the number of PCG steps grows quadratically along the line $N = T$. However, the total processor time of the PCG method is more than one order of magnitude less than the direct method of [15]. Again, performance of the PCG method is comparable MATLAB's Backslash, while enabling decomposition of the computations in a way that matches the spatio-temporal structure of the problem. For $N = T = 200$, the time required for the direct method [15] sits above $O(N^4)$. This is because, for such large systems, the memory management overhead becomes significant as $O(N)$ dense matrices with $O(N^3)$ non-zero elements are involved. The processor time also reflects this memory overhead, and is thus no longer a good proxy for arithmetic complexity.

Fig. 6 shows the maximum condition number of Ψ in (14), and maximum bound for $\kappa(P_L^{-1/2}\Psi P_L^{-1/2})$ in (22) with $L = 2$, across all Newton updates. For this case, Ψ can be very ill-conditioned with $\kappa(\Psi) > 10^{15}$. For the proposed structured preconditioner, the maximum $\kappa(P_L^{-1/2}\Psi P_L^{-1/2})$ bound remains below 10^3 up to $N = 100$.

VI. Conclusions

A decomposable PCG method is proposed for computing second-order search directions for optimal control problems with path-graph network structure. The proposed algorithm exhibits per PCG step arithmetic complexity that scales linearly with the number of sub-systems N and the length of time horizon T. The computations at each iteration can be distributed across parallel processing agents in a network with path-graph structured information exchange. Future work includes extending the results for tree networks, where structure is manifest in three dimensions.

References

[1] H. A. Nasir, M. Cantoni, Y. Li, and E. Weyer, "Stochastic model predictive control based reference planning for automated open-water channels," *IEEE Transactions on Control Systems Technology*, 2019.

[2] Y. Zheng, S. E. Li, K. Li, F. Borrelli, and J. K. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 899–910, 2016.

[3] E. Perea-López, B. E. Ydstie, and I. E. Grossmann, "A model predictive control strategy for supply chain optimization," *Computers and Chemical Engineering*, vol. 27, no. 8-9, pp. 1201–1218, 2003.

[4] A. Giannitrapani, S. Paoletti, A. Vicino, and D. Zarrilli, "Optimal allocation of energy storage systems for voltage control in LV distribution networks," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2859–2870, 2017.

[5] T. Rees, H. S. Dollar, and A. J. Wathen, "Optimal solvers for PDE-constrained optimization," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 271–298, 2010.

[6] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, ser. Applied Mathematical Sciences, vol. 95, Springer, 2016.

[7] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, 2000.

[8] S. J. Wright, "Interior point methods for optimal control of discrete time systems," *Journal of Optimization Theory and Applications*, vol. 77, no. 1, pp. 161–187, 1993.

[9] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[10] A. G. Wills and W. P. Heath, "Interior-point methods for linear model predictive control," *Technical Report EE03016, University of Newcastle, NSW*, 2003.

[11] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[12] A. Shahzad, E.C. Kerrigan, and G.A. Constantinides, "A stable and efficient method for solving a convex quadratic program with application to optimal control," *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1369–1393, 2012.

[13] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proc. 51st IEEE Conference on Decision and Control (CDC)*, pp. 668–674, 2012.

[14] I. Nielsen and D. Axehill, "Direct parallel computations of second-order search directions for model predictive control," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2845–2860, 2019.

[15] M. Cantoni, F. Farokhi, E. Kerrigan, and I. Shames, "Structured computation of optimal controls for constrained cascade systems," *International Journal of Control*, vol. 93, no. 1, pp. 30–39, 2020.

[16] A. Hansson and S. K. Pakazad, "Exploiting Chordality in Optimization Algorithms for Model Predictive Control," in *Lecture Notes in Mathematics*, 2018, pp. 11–32.

[17] J. N. Bertsekas, Dimitri P and Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends®in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] B. O'Donoghue, G. Stathopoulos, and S. Boyd, "A splitting method for optimal control," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2432–2442, 2013.

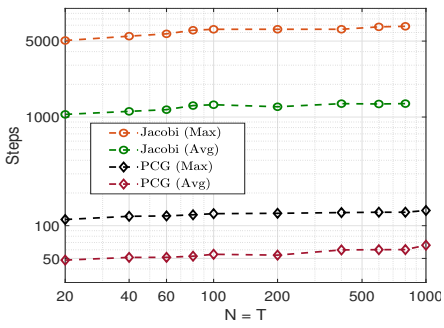


Fig. 1. Case (I): Max/average pure Jacobi and PCG steps per Newton update

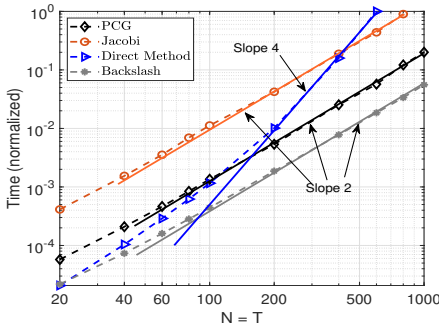


Fig. 2. Case (I): Normalized average time per Newton update

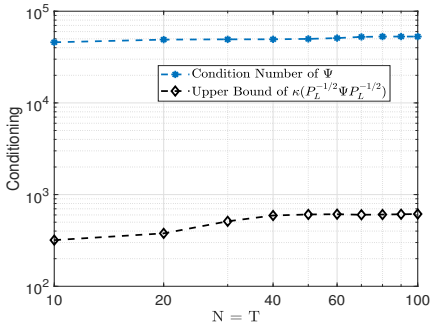


Fig. 3. Case (I): Max $\kappa(\Psi)$ in (14) and max bound for $\kappa(P_L^{-1/2}\Psi P_L^{-1/2})$ in (22) with $L = 2$.

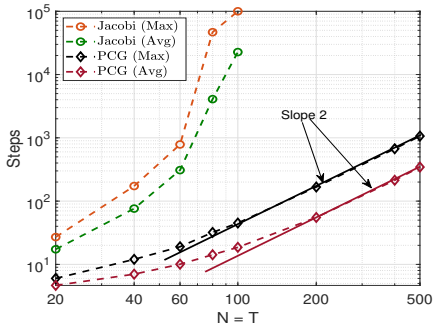


Fig. 4. Case (II): Max/average pure Jacobi and PCG steps per Newton update

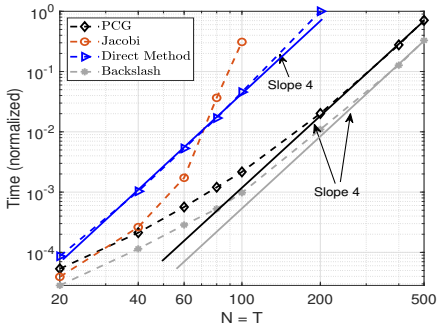


Fig. 5. Case (II): Normalized average time per Newton update

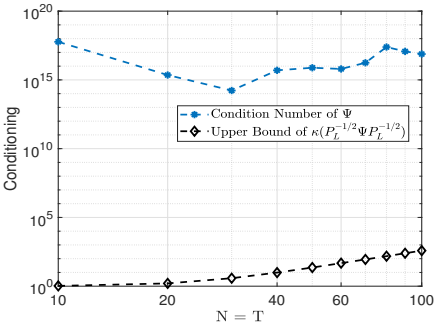


Fig. 6. Case (II): Max $\kappa(\Psi)$ in (14) and max bound for $\kappa(P_L^{-1/2}\Psi P_L^{-1/2})$ in (22) with $L = 2$.

[20] G. Stathopoulos, H. A. Shukla, A. Szuecs, Y. Pu, and C. Jones, "Operator splitting methods in control," *Foundations and Trends in Systems and Control*, vol. 3, pp. 249–362, 2016.

[21] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149–158, 2017.

[22] M. Cantoni, A. Zafar, and F. Farokhi, "Scalable iterations for solving constrained LQ control problems with cascade dynamics," in *Proc. 23rd International Symposium on Mathematical Theory of Networks and Systems* (MTNS), 2018.

[23] F. Rey, P. Hokayem, and J. Lygeros, "ADMM for Exploiting Structure in MPC Problems," *IEEE Transactions on Automatic Control*, 2020.

[24] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[25] G. França and J. Bento, "How is Distributed ADMM Affected by Network Topology?", 2017. [Online]. Available: http://arxiv.org/abs/1710.00889

[26] A. Nedic, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[27] M. M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*. NBS, vol. 49, no. 1, 1952.

[28] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 12, no. 4, pp. 617–629, 1975.

[29] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[30] Y. Saad, *Iterative Methods for Sparse Linear Systems*. *SIAM*, 2003.

[31] M. Benzi and M. Tůma, "A robust incomplete factorization preconditioner for positive definite matrices," *Numerical Linear Algebra with Applications*, vol. 10, no. 5-6, pp. 385–400, 2003.

[32] J. Xia and Z. Xin, "Effective and robust preconditioning of general SPD matrices via structured incomplete factorization," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1298–1322, 2017.

[33] P. Concus, G. H. Golub, and D. P. O'Leary, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in *Sparse Matrix Computations*. Elsevier, 1976, pp. 309–332.

[34] O. G. Johnson, C. A. Micchelli, and G. Paul, "Polynomial preconditioners for conjugate gradient calculations," *SIAM Journal on Numerical Analysis*, vol. 20, no. 2, pp. 362–376, 1983.

[35] L. Adams, "m-Step preconditioned conjugate gradient methods," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 2, pp. 452–463, 1985.

[36] G. Meurant, "A review on the inverse of symmetric tridiagonal and block tridiagonal matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 707–728, 1992.

[37] M. Guo, A. Lang, and M. Cantoni, "Structured moving horizon estimation for linear system chains," in *Proc. 18th European Control Conference (ECC)*, pp. 1830–1835, 2019.

[38] I. S. Duff, "MA57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 118–144, 2004.

[39] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, "Control of large-scale irrigation networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 75–91, 2007.

[40] A. R. Neshastehriz, M. Cantoni, and I. Shames, "Water-level reference planning for automated irrigation channels via robust MPC," in *Proc. 13th European Control Conference (ECC)*, pp. 1331–1336, 2014.