Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization

Xueyi Bu[1], Brian Plancher[2]

***Abstract*— There has been a growing interest in parallel strategies for solving trajectory optimization problems. One key step in many algorithmic approaches to trajectory optimization is the solution of moderately-large and sparse linear systems. Iterative methods are particularly well-suited for parallel solves of such systems. However, fast and stable convergence of iterative methods is reliant on the application of a high-quality preconditioner that reduces the spread and increase the clustering of the eigenvalues of the target matrix. To improve the performance of these approaches, we present a new parallel-friendly symmetric stair preconditioner. We prove that our preconditioner has advantageous theoretical properties when used in conjunction with iterative methods for trajectory optimization such as a more clustered eigenvalue spectrum. Numerical experiments with typical trajectory optimization problems reveal that as compared to the best alternative parallel preconditioner from the literature, our symmetric stair preconditioner provides up to a 34% reduction in condition number and up to a 25% reduction in the number of resulting linear system solver iterations.**

I. Introduction

Trajectory optimization algorithms [1] are a powerful, and state-of-the-art set of tools for synthesizing dynamic motions for complex robots [2], [3], [4], [5], [6], [7]. There has been historical interest in parallel strategies [8] for solving trajectory optimization problems and several more recent efforts have shown that significant computational benefits are possible by exploiting parallelism in different stages of the algorithm on heterogeneous hardware platforms including multi-core CPUs [9], [10], [11], [12], GPUs [13], [14], [15], [16], [17], [18], and FPGAs [19], [20], [21], [22]. This shift toward parallelism is growing increasingly important with the impending end of Moore's Law and the end of Dennard Scaling, which have led to a utilization wall that limits the performance a single CPU chip can deliver [23], [24].

One key step in many algorithmic approaches to trajectory optimization is the solution of moderately-large and sparse linear systems. These systems can be solved by direct factorization or through iterative fixed point approaches. While many state-of-the-art solvers leverage factorization-based approaches [25], [26], iterative methods, like the Preconditioned Conjugate Gradient (PCG) algorithm [27], are particularly well-suited for parallelism, as they are computationally dominated by matrix-vector products and vector reductions [28], [29]. Furthermore, in the context of very-large-scale linear systems GPU implementations of PCG, and other iterative methods, have already been shown to outperform both factorization and iterative approaches on the CPU [30], [31]. However, fast and stable convergence of such iterative methods is reliant on the use of a high-quality symmetric preconditioner that reduces the spread and increase the clustering of the eigenvalues of the target matrix [28], [32], [33], [34].

To improve the performance of these approaches, we present a new parallel-friendly symmetric stair preconditioner. We prove that our preconditioner has advantageous theoretical properties when used in conjunction with iterative methods for trajectory optimization such as a more clustered eigenvalue spectrum than previous parallel preconditioners. Numerical experiments with typical trajectory optimization problems reveal that as compared to the best alternative parallel preconditioner from the literature, our symmetric stair preconditioner provides up to a 34% reduction in condition number and up to a 25% reduction in the number of PCG iterations needed for convergence.

II. Background and Related Work

A. Direct Trajectory Optimization

Trajectory optimization [1], also known as numerical optimal control, solves an (often) nonlinear optimization problem to compute a robot's optimal path through an environment as a series of states ($\mathrm{x} \in \mathbb{R}^n$) and controls ($\mathrm{u} \in \mathbb{R}^m$). These problems assume a discrete-time dynamical system,

$$x_{k+1} = f(x_k, u_k, h), \quad x_0 = x_s, \tag{1}$$

with a timestep h, and minimize an additive cost function,

$$J(X, U) = \ell_f(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k). \tag{2}$$

One approach for solving such problems is through the use of *direct methods*. At each iteration of a direct method, a quadratic approximation of the nonlinear problem defined by Equations 1 and 2 is formed around a nominal trajectory (X, U), resulting in a quadratic program (QP). The resulting KKT system [35], [36] is then solved to update the nominal trajectory. This process is repeated until convergence.

One approach to solving the KKT system is through two step process using the symmetric positive definite *Schur*

This material is based upon work supported by the National Science Foundation (under Award 2246022). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations.

[1]Xueyi Bu is with the School of Engineering and Applied Science, Columbia University, New York, NY. xb2133@columbia.edu

[2]Brian Plancher is with Barnard College, Columbia University, New York, NY. bplancher@barnard.edu

Complement, S.[1] This process first solves for the optimal Lagrange multipliers, λ^*, and then recovers the state and control update, $\delta z^* = [\delta x^*, \delta u^*]$ as follows (where c and C are the constraint value and Jacobian, and g and G are the cost Jacobian and Hessian respectively):

$$\begin{aligned} S &= -CG^{-1}C^T & \gamma &= c - CG^{-1}g \\ S\lambda^* &= \gamma & \delta z^* &= G^{-1}(g - C^T\lambda^*) \end{aligned} \tag{3}$$

If we denote the dynamics Jacobians as $A_k = f_{x_k}$ and $B_k = f_{u_k}$, and cost Jacobians and Hessians as $q_k = J_{x_k}$, $r_k = J_{u_k}$, $Q_k = J_{x_k x_k}$, and $R_k = J_{u_k u_k}$, we can define:

$$\begin{aligned} \theta_k &= -A_k Q_k^{-1} A_k^T - B_k R_k^{-1} B_k^T - Q_{k+1}^{-1} \\ \phi_k &= A_k Q_k^{-1} \\ \zeta_k &= A_k Q_k^{-1} q_k + B_k R_k^{-1} r_k - Q_{k+1}^{-1} q_{k+1}. \end{aligned} \tag{4}$$

The Schur complement then takes the following block-tridiagonal form:

$$\begin{aligned} S &= \begin{pmatrix} -Q_0^{-1} & \phi_0^T & & & \\ \phi_0 & \theta_0 & \phi_1^T & & \\ & & \ddots & & \\ & & \phi_{N-2} & \theta_{N-2} & \phi_{N-1}^T \\ & & & \phi_{N-1} & \theta_{N-1} \end{pmatrix} \\ \gamma &= c + \begin{pmatrix} -Q_0^{-1} q_0 & \zeta_0 & \zeta_1 & \dots & \zeta_{N-1} \end{pmatrix}^T. \end{aligned} \tag{5}$$

For most trajectory optimization problems Q and R are chosen to be symmetric positive definite, or made symmetric positive definite by construction (e.g., in the case of quadratic penalty methods), and C^T has full column rank. Furthermore, by construction θ is both invertible and symmetric. Similarly, although A_k and B_k will vary depending on the integrator in use, for most popular explicit and semi-implicit integrators they take the form

$$A = I + hM \quad B = hN, \tag{6}$$

where M and N are functions of the system dynamics f. Additional conditions, such as Lipschitz continuity, are usually imposed on f to ensure the local existence and uniqueness of the solution to the dynamical system [37]. As a result, $\|M\|$ is bounded; thus, for sufficiently small h, $h\|M\| < 1$ and $A = I + hM$, and thus ϕ, is invertible. Finally, given all of the above, the dominant computational step in this approach is solving for λ^* in Equation 3.

B. Parallel Iterative Solvers

There has been a significant amount of prior work developing general purpose parallel iterative solvers, many of which target the GPU [28], [38], [39], [40], [41], [42], [43]. Iterative methods solve the problem $S\lambda^* = \gamma$ for a given S and γ by iterative refining an estimate for λ up to some tolerance ϵ. The most popular of these methods is the Conjugate Gradient (CG) method, which is applicable to systems where S is positive definite. The convergence rate of CG is directly related to the spread of the Eigenvalues of $S \in \mathbb{R}^{n \times n}$, converging faster when they are clustered. We also note that clustered, and moderate in magnitude, eigenvalues also have the added benefit of avoiding round-off and overflow errors caused by iterative floating point math.

To improve the performance of CG, a symmetric preconditioning matrix $\Phi \approx S$ is often applied to instead solve the equivalent problem $\Phi^{-1}S\lambda^* = \Phi^{-1}\gamma$ [44]. The resulting preconditioned CG (PCG) algorithm leverages matrix-vector products with S and Φ^{-1}, as well as vector reductions, both parallel friendly operations. Finally, we note that PCG requires a symmetric preconditioner [44].

C. Parallel Preconditioners

There is a large literature on preconditioning [28], [45], [46], [47], [48], [49] and there are many different preconditioners that are optimized for computation on vector or parallel processors. The most popular of these are the Jacobi and Block-Jacobi preconditioners. These set:

$$\Phi = \text{diag}(S) \qquad \text{or} \qquad \Phi = \text{block-diag}(S). \tag{7}$$

Previous GPU based iterative solvers mainly leveraged these preconditioners [42], [43]. For block banded matrices, alternating and overlapping block preconditioners have also been used in previous work. These methods compute Φ^{-1} as a sum of the inverse of block-diagonal matrices that compose S [28], [50]. Finally, Polynomial splitting preconditioners have also found widespread usage [28], [34]. These follow the pattern $S = \Psi - P$ and compute a preconditioner where:

$$S^{-1} \approx \Phi^{-1} = (I + \Psi^{-1}P + (\Psi^{-1}P)^2 \dots)\Psi^{-1}. \tag{8}$$

We note that these preconditioners are only valid where:

$$\rho(\Psi^{-1}P) < 1, \tag{9}$$

which is called a convergent splitting and will guarantee convergence when used with the CG algorithm [28], [32], [33], [34]. Also, increasing the degree of the polynomial computes a better approximation of S and improves the convergence rate of the resulting PCG algorithm. However, this requires more computation to compute the preconditioner and also often creates a preconditioner with a larger bandwidth, requiring more memory. For block-banded matrices, like S in our problem, the values in the true inverse decay exponentially as one moves away from the diagonal [51], this creates a tradeoff between the accuracy and both the memory and computational complexity of the preconditioner.

III. Stair Preconditioners

One polynomial splitting for (symmetric) block tridiagonal matrices that has been shown to outperform Jacobi and Block-Jacobi methods is the stair based splitting [52], [53]. This splitting comes in two types, left (type 1) and right (type 2), depending upon the direction of the stair. For a 3x3 symmetric block tridiagonal S:

$$S = \begin{bmatrix} D_1 & O_1 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & O_2^T & D_3 \end{bmatrix} \tag{10}$$

[1] While S is actually symmetric negative definite, one can instead negate it and solve for $-\lambda^*$, enabling us to simplify our proofs by treating it as symmetric positive definite in the remainder of this work.

The stair splittings are:

$$\Psi_l = \begin{bmatrix} D_1 & 0 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & 0 & D_3 \end{bmatrix} \quad P_l = -\begin{bmatrix} 0 & O_1 & 0 \\ 0 & 0 & 0 \\ 0 & O_2^T & 0 \end{bmatrix}$$
$$\Psi_r = \begin{bmatrix} D_1 & O_1 & 0 \\ 0 & D_2 & 0 \\ 0 & O_2^T & D_3 \end{bmatrix} \quad P_r = -\begin{bmatrix} 0 & 0 & 0 \\ O_1^T & 0 & O_2 \\ 0 & 0 & 0 \end{bmatrix}. \tag{11}$$

The stair is invertible if and only if its diagonal blocks, D, are invertible. If so, its inverse has the same stair shaped sparsity pattern drawing from the same or neighboring block-rows of S, enabling efficient parallel computation [29], [54]:

$$\Psi^{-1} = D^{-1}(2D - \Psi)D^{-1}$$
$$\Psi_l^{-1} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ -D_2^{-1}O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1}O_2 D_3^{-1} \\ 0 & 0 & D_3^{-1} \end{bmatrix} \tag{12}$$
$$\Psi_r^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1 D_2^{-1} & 0 \\ 0 & D_2^{-1} & 0 \\ 0 & -D_3^{-1}O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix}.$$

While, the left and right stair preconditioners are not symmetric, and thus cannot be used in the context of PCG, the *additive polynomial preconditioner* can instead be used:

$$\Phi_{add}^{-1} = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1}). \tag{13}$$

However, as we shall see later on, this will result in a declustered spectrum that could negatively impact the performance of PCG. In the remainder of this section we prove a few lemmas and theorems about the eigenpairs of Ψ_l^{-1}, Ψ_r^{-1}, and $\Phi_{add}^{-1}S$ which we will need for later proofs.[2]

A. Eigenpairs of Ψ_l and Ψ_l

We first prove a few lemmas about the eigenpairs of the left and right stair preconditioners.[3]

Lemma 3.1: Given the stair-splittings of a symmetric block tridiagonal matrix $S = \Psi_l - P_l = \Psi_r - P_r$ for a $n \times n$ block S, where each block is $m \times m$, $\Psi_l^{-1}P_l$ and $\Psi_r^{-1}P_r$ have the same spectrum.

Proof: If (λ, v) is an eigenpair of $P_r\Psi_r^{-1}$ and $\lambda \neq 0$:

$$\Psi_r^{-1}P_r(\Psi_r^{-1}v) = \Psi_r^{-1}(\lambda v), \tag{14}$$

and $(\lambda, \Psi_r^{-1}v)$ is an eigenpair of $\Psi_r^{-1}P_r$. Since P_r is not invertible, 0 is an eigenvalue for both $P_r\Psi_r^{-1}$ and $\Psi_r^{-1}P_r$. Also as P_r and Ψ_r^{-1} are equal dimensioned square matrices then by Theorem 1.3.22 of [55], $P_r\Psi_r^{-1}$ and $\Psi_r^{-1}P_r$ have the same spectrum. Finally, $(\Psi_l^{-1}P_l)^T = P_r\Psi_r^{-1}$ and every matrix has the same spectrum as its transpose. ∎

Lemma 3.2: For $v^T = (v_1^T, v_2^T, \dots, v_i^T, \dots, v_n^T) \in \mathbb{R}^{nm}$ and $v_i \in \mathbb{R}^m$ for $i = 1, 2, \dots, n$, we denote $v_e^T = (0, v_2^T, \dots, 0, v_{2j}^T, \dots)$ and $v_o^T = (v_1^T, 0, \dots, v_{2j+1}^T, 0, \dots)$ such that $v = v_e + v_o$.

If $(\lambda, v = v_e + v_o)$ is an eigenpair of $\Psi_l^{-1}P_l$ and $\lambda \neq 0$, then $(\lambda, u = v_e + \lambda v_o)$ is an eigenpair of $\Psi_r^{-1}P_r$.

If $(\lambda, u = u_e + u_o)$ is an eigenpair of $\Psi_r^{-1}P_r$ and $\lambda \neq 0$, then $(\lambda, v = \lambda u_e + u_o)$ is an eigenpair of $\Psi_l^{-1}P_l$.

Proof: $\Psi_l^{-1}P_l v_o = 0$ for all v_o because of the zero columns of $\Psi_l^{-1}P_l$ (see Equation 45 in the appendix). Similarly, $P_r v_e = 0$ and $\Psi_r^{-1}P_r v_e = 0$ (see Equation 46 in the appendix).

Suppose $(\lambda, v = v_e + v_o)$ is an eigenpair of $\Psi_l^{-1}P_l$, then

$$\Psi_l^{-1}P_l v_e = \Psi_l^{-1}P_l v = \lambda(v_e + v_o). \tag{15}$$

If $\lambda \neq 0$, we split Equation 15 into two parts,

$$(\Psi_l^{-1} - D^{-1})P_l v = (\Psi_l^{-1} - D^{-1})P_l v_e = \lambda v_e \tag{16}$$
$$D^{-1}P_l v = D^{-1}P_l v_e = \lambda v_o \tag{17}$$

where D is the diagonal blocks of S (see Equations 47 and 48 in the appendix for more details).

Since $P_r = \Psi_r - S = D - \Psi_l$, then:

$$D^{-1}P_r = I - D^{-1}\Psi_l = (\Psi_l^{-1} - D^{-1})\Psi_l. \tag{18}$$

This means that:

$$\begin{aligned} D^{-1}P_r v &= (\Psi_l^{-1} - D^{-1})\Psi_l v \\ &= (\Psi_l^{-1} - D^{-1})\Psi_l(\frac{1}{\lambda}\Psi_l^{-1}P_l v) \\ &= \frac{1}{\lambda}(\Psi_l^{-1} - D^{-1})P_l v \\ &= v_e \end{aligned} \tag{19}$$

Using the inverse of the stair matrix (Equation 12):

$$\Psi_r^{-1} - D^{-1} = D^{-1}(D - \Psi_r)D^{-1} = D^{-1}P_l D^{-1} \tag{20}$$
$$\begin{aligned} (\Psi_r^{-1} - D^{-1})P_r v &= D^{-1}P_l D^{-1}P_r v \\ &= D^{-1}P_l v_e \\ &= \lambda v_o \end{aligned} \tag{21}$$

Combining Equations 19 and 21 as well as make use of the fact $P_r v_e = 0$, we obtain the following result:

$$\begin{aligned} \Psi_r^{-1}P_r(v_e + \lambda v_o) &= \Psi_r^{-1}P_r(\lambda v) \\ &= \lambda(D^{-1}P_r v + (\Psi_r^{-1} - D^{-1})P_r v) \\ &= \lambda(v_e + \lambda v_o) \end{aligned} \tag{22}$$

The second statement follows from Lemma 4.1 or can be proved with the same strategy as above. ∎

Finally, we note that the eigenvectors with eigenvalues at 0 can be divided into two types. The first type of these are the eigenvectors v_o and u_e which are formed because of the zero columns of $\Psi_l^{-1}P_l$ and $\Psi_r^{-1}P_r$ (see Equations 45 and 46 in the appendix). The second type of these are the eigenvectors v_e and u_o such that $P_l v_e = 0$ and $P_r u_o = 0$.

If n is even, then $\Psi_l^{-1}P_l$ and $\Psi_r^{-1}P_r$ have exactly $m\frac{n}{2}$ eigenpairs of the first type. If n is odd, $\Psi_l^{-1}P_l$ will have

[2] We note that throughout this paper we analyze the spectrum of $\Psi^{-1}S$ even though the resulting matrix is not guaranteed to be symmetric positive definite. This is possible because through the Cholesky factorization, $\Psi = LL^T$, we can instead from the symmetric positive definite $L^{-1}SL^{-T}$, and the matrices $L^{-1}SL^{-T}$ and $\Psi^{-1}S$ are similar.

[3] For more details on stair preconditioners please see [52] and [53].

$m\frac{n+1}{2}$ while $\Psi_r^{-1}P_r$ will have m less such eigenpair and, instead, have m more eigenpairs in the form of $(0, u_o)$ since, in this case, $P_r u_o = 0$, which has non-zero solutions even when all Os are invertible:

$$\begin{cases} O_1^T u_1 + O_2 u_3 = 0 \\ O_3^T u_3 + O_4 u_5 = 0 \\ \dots \\ O_{n-2}^T u_{n-2} + O_{n-1} u_n = 0 \end{cases} \tag{23}$$

In the case of most integrators, as discussed before, Os are invertible, and $P_l v_e = 0$ if and only if $v_e = 0$. Hence, there are exactly $m\lceil\frac{n}{2}\rceil$ eigenpairs with eigenvalues at 0 and $m\lfloor\frac{n}{2}\rfloor$ eigenpairs with non-zero eigenvalues.

B. Eigenpairs of Φ_{add}

We next construct the mn eigenpairs of $\Phi_{add}^{-1}S$ from those of $\Psi_l^{-1}P_l$ and $\Psi_r^{-1}P_r$.

Theorem 3.3: If $(0, v_e)$ is an eigenpair of $\Psi_l^{-1}P_l$ then $(1, v_e)$ is an eigenpair of $\Phi_{add}^{-1}S$ and if $(0, u_o)$ is an eigenpair of $\Psi_r^{-1}P_r$ then $(1, u_o)$ is an eigenpair of $\Phi_{add}^{-1}S$.

Proof: Let $(0, v_e)$ be an eigenpair of $\Psi_l^{-1}P_l$, then:

$$\Psi_l^{-1}Sv_e = \Psi_l^{-1}(\Psi_l - P_l)v_e = v_e - 0v_e = v_e \tag{24}$$

$$\Psi_r^{-1}Sv_e = \Psi_r^{-1}(\Psi_r - P_r)v_e = v_e - 0 = v_e. \tag{25}$$

$$\Phi_{add}^{-1}Sv_e = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})Sv_e = \frac{1}{2}(v_e + v_e) = v_e. \tag{26}$$

Let $(0, u_o)$ be an eigenpair of $\Psi_r^{-1}P_r$, then

$$\Psi_l^{-1}Su_o = \Psi_l^{-1}(\Psi_l - P_l)u_o = u_o - 0 = u_o \tag{27}$$

$$\Psi_r^{-1}Su_o = \Psi_r^{-1}(\Psi_r - P_r)u_o = u_o - 0u_o = u_o. \tag{28}$$

$$\Phi_{add}^{-1}Su_o = u_o. \tag{29}$$

■

Theorem 3.4: Let $(\lambda, v = v_e + v_o)$ be an eigenpair of $\Psi_l^{-1}P_l$. If $\lambda \neq 0$, then $(1 - \frac{1}{2}(\lambda \pm \sqrt{\lambda}), v_e \pm \sqrt{\lambda}v_o)$ are eigenpairs of $\Phi_{add}^{-1}S$.

Proof: Let $(\lambda, v = v_e + v_o)$ be an eigenpair of $\Psi_l^{-1}P_l$: Since $\Psi_l^{-1}P_l v_o = 0$,

$$\Psi_l^{-1}P_l v_e = \lambda(v_e + v_o). \tag{30}$$

Similarly, Equation 22 and $\Psi_r^{-1}P_r v_e = 0$ suggests:

$$\Psi_r^{-1}P_r v_o = v_e + \lambda v_o. \tag{31}$$

We can therefore make an educated guess that the eigenvalues of $\Phi_{add}^{-1}S = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})S = \frac{1}{2}(\Psi_l^{-1}S + \Psi_r^{-1}S)$ and should be of the form $u = av_e + bv_o$. Following from Equations 30 and 31, we can therefore say that:

$$\begin{aligned} \Phi_{add}^{-1}Su &= \frac{1}{2}(\Psi_l^{-1}S(av_e + bv_o) + \Psi_r^{-1}S(av_e + bv_o)) \\ &= u - \frac{1}{2}\left[b\Psi_l^{-1}P_l v_e + a\Psi_r^{-1}P_r v_o\right] \\ &= u - \frac{1}{2}\left[b\lambda(v_e + v_o) + a(v_e + \lambda v_o)\right] \\ &= u - \frac{1}{2}\left[(a + \lambda b)v_e + \lambda(a + b)v_o\right] \end{aligned} \tag{32}$$

Given our assumed form $u = av_e + bv_o$, Equation 32 should simplify to: $\beta a v_e + \beta b v_o$. Thus, the eigenpair (λ, u) must satisfy the linear system:

$$\frac{1}{2}\begin{pmatrix} 1 & \lambda \\ \lambda & \lambda \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = (1 - \beta)\begin{pmatrix} a \\ b \end{pmatrix} \tag{33}$$

This means we can conclude that:

$$\beta = 1 - \frac{1}{2}(\lambda \pm \sqrt{\lambda}), \qquad b = \pm\sqrt{\lambda}a \tag{34}$$

■

Theorem 3.5: If S is symmetric block tridiagonal and positive definite, then $\Phi_{add}^{-1}S$ has real positive eigenvalues.

Proof: If S is positive definite, then its diagonal block matrix, D, is positive definite and thus:

$$\begin{aligned} v^T S v &> 0 \\ v^T(\Psi_l + \Psi_r - D)v &> 0 \\ v^T(\Psi_l + \Psi_r)v &> 0 \end{aligned} \tag{35}$$

Since $\Psi_l = \Psi_r^T$ we can also note that Ψ_l, Ψ_r, Ψ_l^{-1}, and Ψ_r^{-1} are all positive definite as:

$$v^T(\Psi_l + \Psi_r)v = 2v^T\Psi_l v = 2v^T\Psi_r v \tag{36}$$

As $(\Psi_l^{-1})^T = \Psi_r^{-1}$, $\Phi_{add}^{-1} = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})$ is symmetric and positive definite. $\Phi_{add}^{-1}S$ is then similar to the symmetric positive definite matrix $\frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})^{1/2}S(\Psi_l^{-1} + \Psi_r^{-1})^{1/2}$ and must also have positive real eigenvalues. We note that this is a special case of Theorem 3.4 of [34] and Theorem 2.2 and Corollary 2.3 of [46]. ■

IV. The Symmetric Stair Precontioner

In this section, we introduce a new symmetric parallel preconditioner for symmetric positive definite block tridiagonal matrices which improves upon existing stair based preconditoners, the *symmetric stair preconditioner*:

$$\Phi_{sym}^{-1} = \Psi_l^{-1} + \Psi_r^{-1} - D^{-1}. \tag{37}$$

This new preconditioner can also simply be thought of as taking Ψ_l^{-1} or Ψ_r^{-1} and copying the off diagonal blocks across the diagonal and takes the form:

$$\Phi_{sym}^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1D_2^{-1} & 0 \\ -D_2^{-1}O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1}O_2D_3^{-1} \\ 0 & -D_3^{-1}O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix} \tag{38}$$

In the remainder of this section, we will prove that $\Phi_{sym}^{-1}S$ is admissible for use with PCG, and that $\Phi_{sym}^{-1}S$ has a more clustered spectrum than $\Phi_{add}^{-1}S$.

We begin by showing that $\Phi_{sym}^{-1}S$ and $\Psi_l^{-1}S$ (or $\Psi_r^{-1}S$) share the same eigenvalues. And, that the multiplicities of the eigenvalues of $\Phi_{sym}^{-1}S$ that are less than one are doubled as compared to $\Psi_l^{-1}S$ (or $\Psi_r^{-1}S$).

Theorem 4.1: If $(0, v_e)$ is an eigenpair of $\Psi_l^{-1}P_l$ then $(1, v_e)$ is an eigenpair of $\Phi_{sym}^{-1}S$. If $(0, u_o)$ is an eigenpair of $\Psi_r^{-1}P_r$ then $(1, u_o)$ are eigenpairs of $\Phi_{sym}^{-1}S$.

Proof: Let $(0, v_e)$ be an eigenpair of $\Psi_l^{-1}P_l$, then by Equations 16, 25 and 48 in the appendix:

$$\begin{aligned}\Phi_{sym}^{-1}Sv_e &= (\Psi_r^{-1} + (\Psi_l^{-1} - D^{-1}))Sv_e \\ &= v_e + (\Psi_l^{-1} - D^{-1})(\Psi_l - P_l)v_e \\ &= v_e + (\Psi_l^{-1} - D^{-1})\Psi_l v_e - (\Psi_l^{-1} - D^{-1})P_l v_e \\ &= v_e + 0 - 0v_e = v_e\end{aligned} \tag{39}$$

Similarly, let $(0, u_o)$ be an eigenpair of $\Psi_r^{-1}P_r$, then:

$$\begin{aligned}\Phi_{sym}^{-1}Su_o &= (\Psi_l^{-1} + (\Psi_r^{-1} - D^{-1}))Su_o \\ &= u_o + (\Psi_r^{-1} - D^{-1})(\Psi_r - P_r)u_o \\ &= u_o + (\Psi_l^{-1} - D^{-1})\Psi_r u_o - (\Psi_r^{-1} - D^{-1})P_r u_o \\ &= u_o + 0 - 0u_o = u_o\end{aligned} \tag{40}$$

■

Theorem 4.2: If $(\lambda, v = v_e + v_o)$ is an eigenpair of $\Psi_l^{-1}P_l$, then $(1-\lambda, v_o)$ and $(1-\lambda, v_e)$ are eigenpairs of $\Phi_{sym}^{-1}S$.

Proof: Using $P_r v_e = 0$, Equations 16 and 48:

$$\begin{aligned}\Phi_{sym}^{-1}Sv_e &= (\Psi_r^{-1} + (\Psi_l^{-1} - D^{-1}))Sv_e \\ &= \Psi_r^{-1}(\Psi_r - P_r)v_e + (\Psi_l^{-1} - D^{-1})(\Psi_l - P_l)v_e \\ &= v_e + (\Psi_l^{-1} - D^{-1})\Psi_l v_e - (\Psi_l^{-1} - D^{-1})P_l v_e \\ &= v_e + 0 - \lambda v_e \\ &= (1-\lambda)v_e.\end{aligned} \tag{41}$$

Hence, $(1-\lambda, v_e)$ is an eigenpair of $\Phi_{sym}^{-1}S$. The proof for $(1-\lambda, v_o)$ follows the same structure. ■

Finally, we will prove that $\Phi_{sym}^{-1}S$ is admissible for PCG, and that $\Phi_{sym}^{-1}S$ has a smaller condition number than $\Phi_{add}^{-1}S$.

Theorem 4.3: If S is symmetric block tridiagonal and positive definite, then:

1) $\Phi_{sym}^{-1}S$ has real eigenvalues.
2) $\Phi_{sym}^{-1}P$ has non-negative real eigenvalues and $\rho(\Phi_{sym}^{-1}P) < 1$.
3) $\Phi_{sym}^{-1}S$ is positive definite and $\lambda(\Phi_{sym}^{-1}S) \in (0, 1]$.

Proof: We prove each part in order:

1) As S is positive definite, then $\Phi_{sym}^{-1}S$ is similar to the symmetric matrix $S^{1/2}\Phi_{sym}^{-1}S^{1/2}$. Therefore, the eigenvalues of $\Phi_{sym}^{-1}S$ are real.
2) As $\lambda \in \mathbb{R}$ from point one, then as we again derive λ from Ψ, by Equation 34 and Theorem 3.5, $1 - \frac{1}{2}(\lambda \pm \sqrt{\lambda})$ will be real and positive if and only if $0 \leq \lambda < 1$. Thus, $\Phi_{sym}^{-1}P$ has non-negative real eigenvalues and $\rho(\Phi_{sym}^{-1}P) < 1$. This also shows that the stair splitting is convergent for a symmetric positive definite S.
3) From points one and two and Theorem 4.1 and Theorem 4.2, $1 - \lambda \in (0, 1]$ and $\Phi_{sym}^{-1}S$ is positive definite.

■

As a result of Theorem 3.4, Theorem 3.5, and Theorem 4.3, we have the following relationships between the maximum (Equation 42) and minimum (Equation 43) eigenvalues of the two preconditioners, where $\lambda_{\max}^{+}(\Psi_x^{-1}P)$ and $\lambda_{\min}^{+}(\Psi_x^{-1}P)$ denotes the largest and smallest non-zero eigenvalues of Ψ_x. As we can see from Equations 42 and 43, the symmetric stair preconditioner results in a smaller condition number as the eigenvalues are of the range:

$$\begin{aligned}\lambda(\Phi_{sym}^{-1}S) &\in (0, 1], \\ \lambda(\Phi_{add}^{-1}S) &\in \left(0, \frac{9}{8}\right].\end{aligned} \tag{44}$$

V. Numerical Results

In this section we present a numerical evaluation of our symmetric stair preconditioner and compare it to the additive stair preconditioner and other parallel preconditioners from the literature on representative trajectory optimization tasks. We evaluate the spread of the eigenvalues both in absolute terms and with regard to the the relative condition number after preconditioning, as well as the resulting number of iterations of PCG needed for convergence. We use the canonical pendulum and cart pole swing up problems as well as a problem to compute a motion across its workspace for a Kuka LBR IIWA-14 manipulator. Source code accompanying this evaluation, including all hyperparameter values used in these experiments, can be found at `github.com/a2r-lab/SymStair`.

Figure 1 shows the distribution of the eigenvalues of the additive and symmetric stair preconditioners along with the line for 0.0 highlighted in black, 1.0 in green, and $\frac{9}{8} = 1.125$ in purple. We see that across all problems our numerical results match the theoretical results in Equation 44.

This theoretical result directly leads to a better relative condition number for our preconditioner as compared to alternative parallel preconditioners from the literature. As shown in Figure 2, when normalizing the condition number to the Jacobi preconditioner equaling 1 for each system, to enable comparison across the three problems which have large differences in the absolute condition number, we see that the symmetric stair preconditioner is the most perfor-

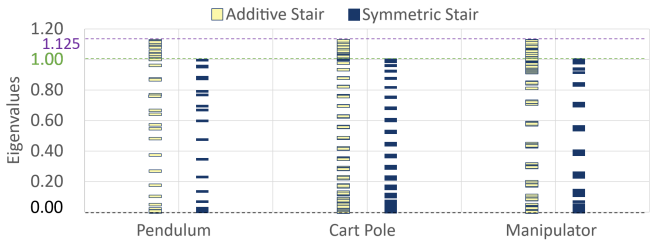

Fig. 1. Distribution of the Eigenvalues of the additive and symmetric stair preconditioners matching the theoretical results in Equation 44.

$$\lambda_{\max}(\Phi_{sym}^{-1}S) = 1 - \lambda_{\min}(\Psi^{-1}P) \leq 1 < 1 - \frac{1}{2}\left(\lambda^{+}(\Psi^{-1}P) - \sqrt{\lambda^{+}(\Psi^{-1}P)}\right) \leq \lambda_{\max}(\Phi_{add}^{-1}S) \leq \frac{9}{8} \tag{42}$$

$$\lambda_{\min}(\Phi_{sym}^{-1}S) = 1 - \lambda_{\max}^{+}(\Psi^{-1}P) > 1 - \frac{1}{2}\left(\lambda_{\max}^{+}(\Psi^{-1}P) + \sqrt{\lambda_{\max}^{+}(\Psi^{-1}P)}\right) = \lambda_{\min}(\Phi_{add}^{-1}S) > 0 \tag{43}$$

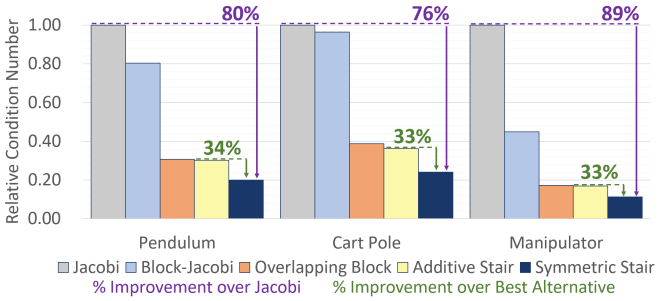

Fig. 2. The relative condition number resulting from different preconditioners, normalized to the value of the Jacobi preconditioner, showing the improved performance of the symmetric stair preconditioner.

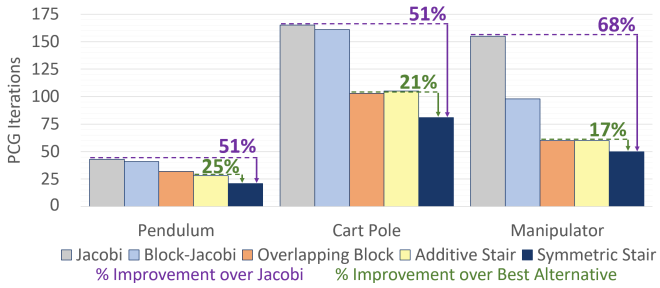

Fig. 3. The number of PCG iterations required for convergence to the same exit tolerance across different preconditioners and problems, again showing the improved performance of the symmetric stair preconditioner.

mant. In fact, it is not only able to reduce the relative condition number by 76-89% as compared to the standard Jacobi preconditioner, but also outperforms the best parallel preconditioner from the literature, the additive stair preconditioner, by 33-34%.

This improvement in numerical conditioning also drastically reduces the number of PCG iterations needed for the problem to converge, enabling faster linear system solves. This is crucial for real-time trajectory optimization as each trajectory optimization solve requires the solution of many linear systems. In particular, as shown in Figure 3, when solving the first KKT system for our three target trajectory optimization problems, PCG using the symmetric stair preconditioner requires 51-68% fewer iterations than the Jacobi preconditioner, and 17-25% fewer iterations than the next best parallel preconditioner to converge to a solution under the same exit tolerance.

VI. Conclusion and Future Work

In this work we present a new parallel-friendly symmetric stair preconditioner. Through both proofs and numerical experiments, we show that our preconditioner has advantageous theoretical and practical properties that enable fast iterative linear system solves. Across three benchmark tasks, our preconditoner provides up to a 34% reduction in condition number and 25% reduction in PCG iterations.

In future work we hope to explore the theoretical and practical performance tradeoffs of higher order polynomial preconditioners on heterogeneous hardware, as well as extend our analysis to provide bounds on the number of PCG iterations under various exit tolerances and preconditioners.

VII. Appendix

Some useful results revealing the sparsity patterns for the 6×6 block cases are listed in Equations 45- 48.

$$\Psi_l^{-1} P_l = \begin{pmatrix} 0 & -D_1^{-1}O_1 & 0 & 0 & 0 & 0 \\ 0 & D_2^{-1}O_2D_3^{-1}O_2^T + D_2^{-1}O_1^T D_1^{-1}O_1 & 0 & D_2^{-1}O_2D_3^{-1}O_3 & 0 & 0 \\ 0 & -D_3^{-1}O_2^T & 0 & -D_3^{-1}O_3 & 0 & 0 \\ 0 & D_4^{-1}O_3^T D_3^{-1}O_2^T & 0 & D_4^{-1}O_4D_5^{-1}O_4^T + D_4^{-1}O_3^T D_3^{-1}O_3 & 0 & D_4^{-1}O_4D_5^{-1}O_5 \\ 0 & 0 & 0 & -D_5^{-1}O_4^T & 0 & -D_5^{-1}O_5 \\ 0 & 0 & 0 & D_6^{-1}O_5^T D_5^{-1}O_4^T & 0 & D_6^{-1}O_5^T D_5^{-1}O_5 \end{pmatrix} \tag{45}$$

$$\Psi_r^{-1} P_r = \begin{pmatrix} D_1^{-1}O_1D_2^{-1}O_1^T & 0 & D_1^{-1}O_1D_2^{-1}O_2 & 0 & 0 & 0 \\ -D_2^{-1}O_1^T & 0 & -D_2^{-1}O_2 & 0 & 0 & 0 \\ D_3^{-1}O_2^T D_2^{-1}O_1^T & 0 & D_3^{-1}O_3D_4^{-1}O_3^T + D_3^{-1}O_2^T D_2^{-1}O_2 & 0 & D_3^{-1}O_3D_4^{-1}O_4 & 0 \\ 0 & 0 & -D_4^{-1}O_3^T & 0 & -D_4^{-1}O_4 & 0 \\ 0 & 0 & D_5^{-1}O_4^T D_4^{-1}O_3^T & 0 & D_5^{-1}O_5D_6^{-1}O_5^T + D_5^{-1}O_4^T D_4^{-1}O_4 & 0 \\ 0 & 0 & 0 & 0 & -D_6^{-1}O_5^T & 0 \end{pmatrix} \tag{46}$$

$$(\Psi_l^{-1} - D^{-1}) P_l = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & D_2^{-1}O_2D_3^{-1}O_2^T + D_2^{-1}O_1^T D_1^{-1}O_1 & 0 & D_2^{-1}O_2D_3^{-1}O_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & D_4^{-1}O_3^T D_3^{-1}O_2^T & 0 & D_4^{-1}O_4D_5^{-1}O_4^T + D_4^{-1}O_3^T D_3^{-1}O_3 & 0 & D_4^{-1}O_4D_5^{-1}O_5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & D_6^{-1}O_5^T D_5^{-1}O_4^T & 0 & D_6^{-1}O_5^T D_5^{-1}O_5 \end{pmatrix} \tag{47}$$

$$D^{-1} P_l = \begin{pmatrix} 0 & -D_1^{-1}O_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -D_3^{-1}O_2^T & 0 & -D_3^{-1}O_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -D_5^{-1}O_4^T & 0 & -D_5^{-1}O_5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (\Psi_l^{-1} - D^{-1}) \Psi_l = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -D_2^{-1}O_1^T & 0 & -D_2^{-1}O_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -D_4^{-1}O_3^T & 0 & -D_4^{-1}O_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -D_6^{-1}O_5^T & 0 \end{pmatrix} \tag{48}$$

REFERENCES

[1] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, ser. Advances in Design and Control. Society for Industrial and Applied Mathematics (SIAM), vol. 3.

[2] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[3] W. Xi and C. D. Remy, "Optimal Gaits and Motions for Legged Robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[4] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1366–1373.

[5] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast Online Trajectory Optimization for the Bipedal Robot Cassie," in *Proceedings of Robotics: Science and Systems*.

[6] T. Howell, B. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2019, pp. 7674 – 7679.

[7] S. Kuindersma, "Taskable agility: Making useful dynamic behavior easier to create," Princeton Robotics Seminar, April 2023.

[8] J. T. Betts and W. P. Huffman, "Trajectory optimization on a parallel processor," vol. 14, no. 2, pp. 431–439.

[9] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl, "A Block Based ALADIN Scheme for Highly Parallelizable Direct Optimal Control," in *Proceedings of the American Control Conference*.

[10] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A family of iterative gauss-newton shooting methods for nonlinear optimal control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[11] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics*.

[12] A. Astudillo, J. Gillis, G. Pipeleers, W. Decré, and J. Swevers, "Speed-up of nonlinear model predictive control for robot manipulators using task and data parallelism," in *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*, 2022, pp. 201–206.

[13] T. Antony and M. J. Grant, "Rapid Indirect Trajectory Optimization on Highly Parallel Computing Architectures," vol. 54, no. 5, pp. 1081–1091.

[14] B. Plancher and S. Kuindersma, "A Performance Analysis of Parallel Differential Dynamic Programming on a GPU," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

[15] ——, "Realtime model predictive control using parallel ddp on a gpu," in *Toward Online Optimal Control of Dynamic Robots Workshop at the 2019 International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, May. 2019.

[16] Z. Pan, B. Ren, and D. Manocha, "Gpu-based contact-aware trajectory optimization using a smooth force model," in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '19. New York, NY, USA: ACM, 2019, pp. 4:1–4:12.

[17] B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, and V. J. Reddi, "Accelerating robot dynamics gradients on a cpu, gpu, and fpga," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2335–2342, 2021.

[18] B. Plancher, S. M. Neuman, R. Ghosal, S. Kuindersma, and V. J. Reddi, "GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6253–6260. [Online]. Available: https://ieeexplore.ieee.org/document/9812384/

[19] J. Sacks, D. Mahajan, R. C. Lawson, and H. Esmaeilzadeh, "Robox: an end-to-end solution to accelerate autonomous control in robotics," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*. IEEE Press, 2018, pp. 479–490.

[20] S. M. Neuman, B. Plancher, T. Bourgeat, T. Tambe, S. Devadas, and V. J. Reddi, "Robomorphic computing: A design methodology for domain-specific accelerators parameterized by robot morphology," ser. ASPLOS 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 674–686. [Online]. Available: https://doi.org/10.1145/3445814.3446746

[21] S. M. Neuman, R. Ghosal, T. Bourgeat, B. Plancher, and V. J. Reddi, "Roboshape: Using topology patterns to scalably and flexibly deploy accelerators across robots," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589104

[22] Y. Yang, X. Chen, and Y. Han, "Rbdcore: Robot rigid body dynamics accelerator with multifunctional pipelines," *arXiv preprint arXiv:2307.02274*, 2023.

[23] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. ACM, pp. 365–376.

[24] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor, "Conservation Cores: Reducing the Energy of Mature Computations," in *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XV. ACM, pp. 205–218.

[25] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[26] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[27] S. C. Eisenstat, "Efficient implementation of a class of preconditioned conjugate gradient methods," *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 1, pp. 1–4, 1981.

[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Apr. 2003.

[29] B. Plancher, "GPU Acceleration for Real-Time, Whole-Body, Nonlinear Model Predictive Controla," Ph.D. dissertation, Harvard University, United States – Massachusetts, 2022.

[30] M. Schubiger, G. Banjac, and J. Lygeros, "GPU acceleration of ADMM for large-scale quadratic programming," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 55–67, Oct. 2020.

[31] R. Helfenstein and J. Koko, "Parallel preconditioned conjugate gradient algorithm on gpu," *Journal of Computational and Applied Mathematics*, vol. 236, no. 15, pp. 3584–3590, 2012.

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

[33] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," USA, Tech. Rep., 1994.

[34] H.-B. Li, T.-Z. Huang, Y. Zhang, X.-P. Liu, and H. Li, "On some new approximate factorization methods for block tridiagonal matrices suitable for vector and parallel processors," *Mathematics and Computers in Simulation*, vol. 79, no. 7, pp. 2135–2147, Mar. 2009.

[35] W. Karush, "Minima of functions of several variables with inequalities as side conditions," Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939.

[36] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, 1951, pp. 481–492.

[37] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[38] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. ACM, pp. 917–924. [Online]. Available: http://doi.acm.org/10.1145/1201775.882364

[39] H. Liu, J.-H. Seo, R. Mittal, and H. H. Huang, "Gpu-accelerated scalable solver for banded linear systems," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2013, pp. 1–8.

[40] H. Anzt, M. Gates, J. Dongarra, M. Kreutzer, G. Wellein, and M. Köhler, "Preconditioned krylov solvers on gpus," *Parallel Computing*, 05 2017.

[41] H. Anzt, M. Kreutzer, E. Ponce, G. D. Peterson, G. Wellein, and J. Dongarra, "Optimization and performance evaluation of the idr iterative krylov solver on gpus," *The International Journal of High Performance Computing Applications*, vol. 32, no. 2, pp. 220–230, 2018.

[42] G. Flegar *et al.*, "Sparse linear system solvers on gpus: Parallel preconditioning, workload balancing, and communication reduction," Ph.D. dissertation, Universitat Jaume I, 2019.

[43] M. Schubiger, G. Banjac, and J. Lygeros, "Gpu acceleration of

admm for large-scale quadratic programming," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 55–67, 2020.

[44] J. W. Pearson and J. Pestana, "Preconditioners for krylov subspace methods: An overview," *GAMM-Mitteilungen*, vol. 43, no. 4, p. e202000015, 2020.

[45] L. Adams, "m-Step Preconditioned Conjugate Gradient Methods," *Society for Industrial and Applied Mathematics. SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 2, p. 12, Apr. 1985.

[46] L. M. Adams and E. G. Ong, "Additive polynomial preconditioners for parallel computers," *Parallel Computing*, vol. 9, no. 3, pp. 333–345, Feb. 1989.

[47] P. Concus, G. H. Golub, and D. P. O'Leary, "Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method," *Computing*, vol. 19, no. 4, pp. 321–339, Dec. 1978.

[48] K. Muzhinji, "Optimal Block Preconditioner for an Efficient Numerical Solution of the Elliptic Optimal Control Problems Using Gmres Solver," *International Journal of Numerical Analysis and Modeling*, vol. 20, no. 1, pp. 47–66, June 2023.

[49] M. Benzi and A. J. Wathen, "Some Preconditioning Techniques for Saddle Point Problems," in *Model Order Reduction: Theory, Research Aspects and Applications*, ser. Mathematics in Industry, W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, Eds. Berlin, Heidelberg: Springer, 2008, pp. 195–211.

[50] E. Galligani and V. Ruggiero, "A polynomial preconditioner for block tridiagonal matrices," *PARALLEL ALGORITHM AND APPLICATIONS*, vol. 3, no. 3-4, pp. 227–237, 1994.

[51] S. Demko, W. F. Moss, and P. W. Smith, "Decay rates for inverses of band matrices," *Mathematics of computation*, vol. 43, no. 168, pp. 491–499, 1984.

[52] H.-B. Li, T.-Z. Huang, Y. Zhang, X.-P. Liu, and H. Li, "On some new approximate factorization methods for block tridiagonal matrices suitable for vector and parallel processors," *Mathematics and Computers in Simulation*, vol. 79, no. 7, pp. 2135–2147, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378475408003881

[53] H.-B. Li, T.-Z. Huang, Y. Zhang, X.-P. Liu, and T.-X. Gu, "Chebyshev-type methods and preconditioning techniques," *Applied Mathematics and Computation*, vol. 218, no. 2, pp. 260–270, 2011.

[54] H. Lu, "Stair Matrices and Their Generalizations with Applications to Iterative Methods I: A Generalization of the Successive Overrelaxation Method," vol. 37, no. 1, pp. 1–17. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S0036142998343294

[55] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.