

Article

A Hybrid Preconditioned Iterative Framework for Large-Scale Multibody Dynamics

Di Wang ^{1,2,3,*}, Hui Ren ¹ , Perry Gu ² and Chongchong Song ³¹ Institute of Aerospace Vehicle Dynamics and Control, School of Astronautics, Harbin Institute of Technology, Harbin 150001, China; renhui@hit.edu.cn² Zhejiang Geely Holding Group Co., Ltd., Hangzhou 310051, China; gupengyun@geely.com³ Geely Automobile Research Institute (Ningbo) Co., Ltd., Ningbo 315336, China; songchongchong@geely.com

* Correspondence: di.wang14@geely.com

Abstract

Multibody dynamics (MBD) simulations involving hundreds to thousands of bodies give rise to large-scale, sparse, and structurally indefinite linear systems. Traditional direct solvers incur prohibitive memory and computational costs, while iterative methods suffer from slow convergence due to severe ill-conditioning. This paper proposes HPI-MBD, a hybrid preconditioned iterative framework. It combines an algebraic multigrid (AMG) for global error smoothing with a block Jacobi preconditioner tailored to the kinematic constraint graph. The framework exploits graph topology to construct a block-diagonal Schur complement approximation, incorporates Tikhonov regularisation for redundant constraints, and maintains $\mathcal{O}(n)$ work per iteration, where n is the number of degrees of freedom. A rigorous spectral analysis supports the problem-size independent convergence of the Minimal Residual (MINRES) solver. Evaluated on five benchmark systems with 10^4 to 10^6 degrees of freedom, the HPI-MBD achieves speedups up to $12.7\times$ and memory reductions up to 68% against MA57, with comparable gains against PARDISO. All solutions maintain relative residuals below 10^{-6} . Comparisons against ILU(0)-preconditioned Generalised Minimal Residual (GMRES), Finite Element Tearing and Interconnecting method (FETI-1), and a block-Jacobi-only variant confirm the essential role of AMG. The framework exhibits near-linear scalability and strong parallel efficiency on up to 32 processors, along with robust performance under redundant constraints and varying time step sizes. These results position HPI-MBD as a scalable, memory-efficient alternative for real-time simulation in virtual prototyping, robotics, and biomechanics.

Keywords: multibody dynamics; saddle-point systems; algebraic multigrid; hybrid preconditioning; Krylov subspace methods; real-time simulation

MSC: 65F08; 65F10; 65F50; 65L80



Academic Editor: Jonathan Blackledge

Received: 30 May 2026

Revised: 19 June 2026

Accepted: 22 June 2026

Published: 25 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The accurate simulation of mechanical systems comprising multiple interconnected rigid and flexible bodies lies at the heart of modern computational engineering. Applications span a broad spectrum, including the design of articulated robotic manipulators [1], autonomous vehicle suspensions [2], biomechanical analysis of the human musculoskeletal system [3], and large-scale virtual prototyping of industrial machinery [4]. In all these contexts, the governing equations of MBD are typically expressed as differential-algebraic

equations (DAEs) of index 3. These equations require the repeated solution of large, sparse, and structurally indefinite linear systems at each time step.

The linear system arising from the Newton–Raphson linearisation of the MBD equations reads:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \text{with} \quad \mathbf{A} = \begin{bmatrix} \mathbf{K}_{\text{eff}} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \Delta \mathbf{q} \\ \Delta \lambda \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{r}_q \\ \mathbf{r}_\lambda \end{bmatrix}, \quad (1)$$

where $\mathbf{K}_{\text{eff}} \in \mathbb{R}^{n \times n}$ is the effective stiffness matrix. This tangent operator encapsulates the combined influence of inertia, elasticity, and constraint forces. It forms the $(1, 1)$ -block of the saddle-point structure. The off-diagonal blocks contain the constraint Jacobian $\mathbf{J} = \partial \mathbf{C} / \partial \mathbf{q}$ and its transpose. The vectors $\Delta \mathbf{q}$ and $\Delta \lambda$ denote corrections to the generalised coordinates and Lagrange multipliers, respectively. The vectors $\mathbf{r}_q$ and $\mathbf{r}_\lambda$ are the corresponding residuals. The lower-right zero block arises from the Lagrange multiplier formulation. This zero block renders the overall coefficient matrix $\mathbf{A}$ indefinite, reflecting the interplay between dynamic evolution and exact constraint satisfaction.

In practical simulations, the number of degrees of freedom n may range from 10^4 in moderate engineering models to 10^6 or beyond in high-resolution flexible body analyses. The structural indefiniteness produces a saddle-point problem that is notoriously challenging for both direct and iterative solvers. Traditional direct solvers are accurate, but they often incur prohibitive memory and computational costs, particularly in real-time or high-fidelity settings. Iterative methods offer better scalability, but their convergence is frequently hindered by severe ill-conditioning.

Direct sparse solvers, such as MA57 [5], PARDISO [6], and MUMPS [7], provide robust and numerically stable solutions for MBD linear systems. However, their applicability to large-scale problems is fundamentally constrained by fill-in incurred during factorisation. For three-dimensional systems with $n \sim 10^5$ degrees of freedom, the LU or LDL^T factorisation of $\mathbf{A}$ typically requires $\mathcal{O}(n^{1.5})$ memory and $\mathcal{O}(n^2)$ operations. These costs render direct solvers impractical for real-time simulation, design optimisation, or high-fidelity analyses that involve repeated solves. Iterative Krylov subspace methods offer a memory-efficient alternative. The MINRES method requires $\mathcal{O}(n)$ storage per iteration. The GMRES method can also achieve $\mathcal{O}(n)$ storage per iteration if restarted periodically, although restarting may affect convergence. In principle, these methods scale favourably and are well suited to large sparse problems. Nevertheless, the spectral properties of saddle-point systems in constrained MBD formulations are extremely adverse. Condition numbers ranging from 10^8 to 10^{12} are common, which renders unpreconditioned iterations virtually useless. The convergence of Krylov solvers degrades sharply under such ill-conditioning, and the computational cost quickly becomes prohibitive.

The central challenge lies in the design of an effective preconditioner $\mathbf{P}$. This preconditioner should approximate $\mathbf{A}$ in some sense, so that the preconditioned system $\mathbf{P}^{-1}\mathbf{A}$ exhibits a clustered spectrum and a modest condition number. The application of $\mathbf{P}^{-1}$ must be computationally inexpensive. Ideally, it should require only $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$ operations per iteration. Striking this balance is particularly difficult for indefinite saddle-point problems. The interplay between physical degrees of freedom and Lagrange multipliers introduces both structural and numerical complexities. In addition, redundant constraints and near-singular kinematic configurations further complicate the design of robust preconditioners. Given that these preconditioning operations are performed at every time step, their cumulative computational cost motivates the exploration of heterogeneous computing architectures. The local dense Schur complement factorisations and sparse matrix–vector products are prime candidates for GPU acceleration, as they constitute the dominant runtime components of many preconditioning strategies.

To address these challenges, this paper introduces HPI-MBD, a hybrid preconditioned iterative framework. This framework combines AMG for global error smoothing with a block Jacobi preconditioner tailored to the underlying kinematic constraint graph. The proposed framework has several key features. It exploits the graph topology of the constraint Jacobian to construct a block-diagonal Schur complement approximation, enabling scalable treatment of the Lagrange multiplier variables. It incorporates Tikhonov regularisation to handle redundant constraints without compromising long-term physical accuracy. It maintains $\mathcal{O}(n)$ work per iteration and exhibits problem-size independent convergence in practice, as supported by a rigorous spectral analysis of the idealised preconditioner and confirmed by numerical experiments. Furthermore, its CPU-based implementation achieves near-linear scalability with respect to problem size. It also maintains strong parallel efficiency above 82% on up to 32 Message Passing Interface (MPI) ranks. The computational hotspots are also well suited for future GPU acceleration.

Before detailing the framework, we briefly review related work on saddle-point preconditioning. A framework for tree-coupled saddle-point systems has been proposed [8]. It introduces multigrid-inspired multi-level preconditioners that exploit structured coupling. While effective for problems with a tree-based block structure, this method relies on a tree assumption. In contrast, HPI-MBD handles arbitrary kinematic constraint graphs. A general class of constraint preconditioners derived from matrix splitting of the $(1, 1)$ -block has been developed [9]. This work unifies several existing approaches and provides rigorous convergence analysis. However, those preconditioners target general saddle-point systems, whereas HPI-MBD specialises to the nonlinear, time-varying setting of multibody dynamics. In this setting, the graph-based Schur complement approximation can be updated efficiently as the configuration changes.

Domain decomposition methods remain highly scalable. The Finite Element Tearing and Interconnecting (FETI) methods have been surveyed [10], demonstrating scalability beyond 65,000 cores. A Balancing Domain Decomposition by Constraints (BDDC) preconditioner for cardiac EMI models has been developed [11], handling discontinuous solutions across cell boundaries. For flexible multibody systems, an enhanced BDDC preconditioner with weighted Schur complement averaging has been proposed [12]. It improves robustness to material heterogeneity. While these methods operate at the finite-element subdomain level, HPI-MBD works directly on the kinematic constraint graph, which is a coarser and more irregular partition. It is inherently lighter and better adapted to rapidly changing constraint topologies. For contact mechanics, algebraic multigrid with filtering (AMGF) has been introduced [13]. It augments AMG with a subspace correction that filters near-null components from contact constraints. This strategy is analogous to our use of graph-based clustering and Tikhonov regularisation to handle redundant constraints. Finally, a multigrid-accelerated global eXtended Position-Based Dynamics (XPBD) solver, termed MGPBD, has been proposed [14]. It combines unsmoothed aggregation AMG with preconditioned conjugate gradient within the XPBD framework for deformable bodies. That work focuses on deformable objects without explicit kinematic constraints, whereas HPI-MBD addresses bilateral constraints via a graph-aware Schur complement approximation.

In summary, existing preconditioners either target different problem structures or lack a tailored treatment of kinematic constraints. They also do not systematically address the combined challenges of constraint redundancy, time-step sensitivity, and outer Newton iteration costs. These three issues are particularly critical in practice. Redundant constraints can render the Schur complement singular. Small time steps can severely degrade the conditioning of the effective stiffness matrix. Inexact linear solves may increase the number of outer Newton iterations. HPI-MBD bridges this gap. It synergistically integrates

AMG smoothing with graph-constrained block-Jacobi preconditioning. It also provides a comprehensive assessment of these practical issues.

The remainder of this paper is organised as follows. Section 2 provides essential background on multibody dynamics formulations, presenting the saddle-point structure of the linear systems that arise upon implicit integration and Newton–Raphson linearisation. Section 3 describes the theoretical underpinnings of the HPI-MBD preconditioner. This includes the construction of the AMG hierarchy for the mechanical sub-block, the design of the block Jacobi preconditioner via graph-based constraint clustering, and a spectral analysis of the preconditioned system. Section 4 details the algorithmic implementation, complexity analysis, and parallelisation strategy. Section 5 evaluates the empirical performance on benchmark systems. It compares HPI-MBD against state-of-the-art direct solvers and other preconditioned iterative methods. It also assesses scalability, robustness, and sensitivity to algorithmic parameters. Section 6 concludes the paper by summarising the key findings and their implications for large-scale multibody dynamics simulation.

2. Multibody Dynamics and the Linear System Structure

This section reviews the essential mathematical background for the linear systems arising in multibody dynamics simulations. The equations of motion are first introduced, followed by the saddle-point structure of the linearised systems, and finally the sparsity patterns and graph properties that will be exploited in the preconditioner design of Section 3.

Consider a mechanical system comprising N_b rigid bodies and N_f flexible bodies interconnected by N_c holonomic bilateral constraints. Let $\mathbf{q} \in \mathbb{R}^n$ denote the vector of generalised coordinates, where $n = 6N_b + n_f$ and n_f represents the degrees of freedom arising from finite element discretisations of the flexible components. The governing equations take the form of an index-3 differential-algebraic system:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}_q^T(\mathbf{q})\boldsymbol{\lambda} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t), \quad (2)$$

$$\mathbf{C}(\mathbf{q}) = \mathbf{0}, \quad (3)$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the generalised mass matrix. This matrix is block diagonal in structure and reflects the inertial properties of individual bodies. The function $\mathbf{C} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ encodes kinematic restrictions. The vector $\boldsymbol{\lambda} \in \mathbb{R}^m$ denotes the Lagrange multipliers that enforce these constraints. The term $\mathbf{f}$ aggregates all applied forces, including external loads, gravitational effects, and elastic restoring forces for flexible bodies.

Direct index reduction to index-1 via double differentiation of Equation (3) yields a mathematically convenient system at the velocity level. However, this approach introduces the well-known problem of constraint drift. In practice, researchers control this drift through techniques such as Baumgarte stabilisation [15] or coordinate partitioning methods [16]. These methods maintain constraint satisfaction without full index reduction. The formulations are typically coupled with implicit time integrators, notably the HHT- α and generalised- α methods. These integrators preserve second-order accuracy while providing controllable numerical dissipation.

The linear system (1) arises from the application of an implicit time integrator to the differential-algebraic system (2)–(3). For example, the generalised- α scheme with spectral radius $\rho_\infty \in [0, 1]$ is a common choice. The Newton–Raphson linearisation at iteration k then produces the system. The matrix $\mathbf{A} \in \mathbb{R}^{(n+m) \times (n+m)}$ is symmetric. Under standard regularity assumptions, it possesses inertia $(p, q, 0)$. This means it has p positive eigenvalues, q negative eigenvalues, and no zero eigenvalues. This structural indefiniteness

is a hallmark of constrained multibody dynamics, and it poses fundamental challenges for both direct and iterative solvers.

A spectral analysis of $\mathbf{A}$ reveals important insights for preconditioner design. Eigenvalues associated with the mechanical degrees of freedom cluster near $1/(\beta_m h^2)$. Eigenvalues corresponding to constraint modes appear near zero but with an opposite sign. The spectral condition number scales approximately as $\kappa(\mathbf{A}) \sim \|\mathbf{K}_{\text{eff}}\| \cdot h^2$. For stiff mechanical systems simulated with small time steps ($h \rightarrow 0$), this condition number can readily exceed 10^{10} . In such cases, unpreconditioned iterative solvers become completely ineffective.

The computational tractability of large-scale MBD simulations hinges on exploiting sparsity. The effective stiffness matrix $\mathbf{K}_{\text{eff}}$ inherits its sparsity pattern from two sources. The first source is the block-diagonal structure of the mass matrix, which reflects the inertial decoupling of distinct bodies. The second source is the connectivity patterns of the underlying finite element discretisations for flexible components.

Of particular significance is the sparsity structure of the constraint Jacobian $\mathbf{J}$. This matrix exhibits a block-sparse pattern that directly mirrors the topology of the mechanical system. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the kinematic constraint graph. Here, $\mathcal{V} = \{1, \dots, N_b + N_f\}$ is the set of vertices representing individual bodies (rigid or flexible). The set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains edges that represent kinematic constraints connecting pairs of bodies.

For each constraint connecting bodies i and j , the corresponding row of $\mathbf{J}$ contains non-zero blocks in the columns associated with the degrees of freedom of bodies i and j . This structural property is fundamental to the scalability of multibody dynamics formulations. It means that constraints couple only a small and local set of bodies.

The graph structure $\mathcal{G}$ plays a pivotal role in the design of our block Jacobi preconditioner, which we detail in Section 3.2. Tree-structured graphs are characteristic of open-chain mechanisms such as robotic manipulators. They yield particularly sparse and amenable Jacobian matrices. Closed kinematic chains, by contrast, introduce cycles that demand more sophisticated treatment. In Section 5, we demonstrate that HPI-MBD remains robust even for systems with complex, cycle-rich constraint graphs. This robustness extends to configurations with redundant constraints and near-singular kinematics.

Understanding the interplay between the continuous equations of motion, the discrete saddle-point structure of the linearised systems, and the underlying graph topology provides the essential foundation for the preconditioning strategies developed in the following section.

3. The HPI-MBD Framework

Building on the structural insights from Section 2, we now introduce the HPI-MBD framework. This is a hybrid preconditioned iterative approach for multibody dynamics. The preconditioner combines a multilevel AMG component for the mechanical block with a graph-based block-Jacobi approximation for the constraint block of the saddle-point system (1). It exploits both the spectral properties of the mechanical block $\mathbf{K}_{\text{eff}}$ and the graph topology of the constraint Jacobian $\mathbf{J}$. The core idea consists of two complementary ingredients. The first is an AMG V-cycle applied to the $(1, 1)$ -block $\mathbf{K}_{\text{eff}}$. The second is a block-Jacobi correction that acts on the constraint degrees of freedom via an approximate Schur complement. The resulting block-triangular preconditioner $\mathbf{P}_{\text{BT}}$ yields a preconditioned system with tightly clustered eigenvalues. This clustering enables fast convergence of Krylov subspace methods such as MINRES.

3.1. Algebraic Multigrid for $\mathbf{K}_{\text{eff}}$

To approximate the inverse of the $(1, 1)$ -block $\mathbf{K}_{\text{eff}}$, we construct an AMG hierarchy using classical Ruge–Stüben coarsening [17,18] with a strength-of-connection threshold $\theta = 0.25$. Let $\mathbf{K}_0 = \mathbf{K}_{\text{eff}}$. For each level $\ell = 0, 1, \dots, L-1$, we define a prolongation operator $\mathbf{P}_\ell$ and set the restriction operator as $\mathbf{R}_\ell = \mathbf{P}_\ell^T$. The coarse-grid operator is then obtained by the Galerkin product

$$\mathbf{K}_{\ell+1} = \mathbf{R}_\ell \mathbf{K}_\ell \mathbf{P}_\ell. \quad (4)$$

The coarsening procedure starts by building the strength graph $\mathcal{S}_\ell$ from $\mathbf{K}_\ell$. An off-diagonal entry (i, j) is considered a strong connection if

$$|(\mathbf{K}_\ell)_{ij}| \geq \theta \max_k |(\mathbf{K}_\ell)_{ik}|, \quad \theta = 0.25. \quad (5)$$

Using a maximal independent set heuristic on $\mathcal{S}_\ell$, we partition the unknowns into coarse points C and fine points F . For each fine point i , the interpolation weights are defined by

$$(\mathbf{P}_\ell)_{i,C(i)} = \frac{(\mathbf{K}_\ell)_{i,C(i)}}{\sum_{j \in C(i)} (\mathbf{K}_\ell)_{ij}}, \quad (6)$$

where $C(i)$ denotes the set of coarse points that are strongly connected to i . For sub-blocks originating from three-dimensional continuum finite elements, which correspond to flexible bodies, we replace the classical interpolation with a smoothed aggregation approach [19]. In that case, the prolongator is given by

$$\mathbf{P}_\ell = \left(\mathbf{I} - \omega \mathbf{D}_\ell^{-1} \mathbf{K}_\ell \right) \mathbf{P}_\ell^{(0)}, \quad \omega = \frac{2}{3}, \quad (7)$$

where $\mathbf{P}_\ell^{(0)}$ is an initial aggregation-based prolongator and $\mathbf{D}_\ell = \text{diag}(\mathbf{K}_\ell)$. The choice $\omega = 2/3$ is standard in smoothed aggregation. This variant preserves near-kernel modes more accurately, especially the rigid-body motions of individual element patches. This property is crucial for robust convergence in problems involving near-incompressible materials or slender structural components.

At each multigrid level, we perform a $\nu_1 = 1$ pre-smoothing step and a $\nu_2 = 1$ post-smoothing step. The choice of smoother depends on the type of degrees of freedom. For scalar degrees of freedom, such as those arising from finite element discretisations, we use red-black Gauss–Seidel. For the indefinite case, the smoother is applied only to the positive-definite part of $\mathbf{K}_{\text{eff}}$. This is a standard strategy in AMG for saddle-point problems. Denote by $\mathbf{D}_\ell$, $\mathbf{L}_\ell$, and $\mathbf{U}_\ell$ the diagonal, strictly lower, and strictly upper parts of $\mathbf{K}_\ell$, respectively. The corresponding smoother operator is

$$\mathbf{S}_\ell^{\text{GS}} = (\mathbf{D}_\ell - \mathbf{L}_\ell)^{-1} + (\mathbf{D}_\ell - \mathbf{U}_\ell)^{-1} - \mathbf{D}_\ell^{-1}. \quad (8)$$

For rigid-body degrees of freedom, where the six translational and rotational coordinates of each body are strongly coupled, we use a block-Gauss–Seidel smoother with 6×6 blocks. Let $\mathbf{Z}_b$ be the matrix that selects the six coordinates of body b . The block smoother is then given by

$$\mathbf{S}_\ell^{\text{block}} = \sum_{b=1}^{N_b} \mathbf{Z}_b (\mathbf{Z}_b^T \mathbf{K}_\ell \mathbf{Z}_b)^{-1} \mathbf{Z}_b^T. \quad (9)$$

A single V-cycle defines the action of the approximate inverse $\tilde{\mathbf{K}}_{\text{eff}}^{-1}$ on a vector $\mathbf{b}$. The V-cycle consists of pre-smoothing to reduce high-frequency errors, restriction of the residual to the coarse grid, coarse-grid correction, interpolation back to the fine grid, and post-smoothing to remove high-frequency components introduced by interpolation. For a level ℓ , let $\tilde{\mathbf{K}}_{\ell}^{-1}$ denote the V-cycle operator at that level. The recursion is

$$\tilde{\mathbf{K}}_{\ell}^{-1}\mathbf{b} = \begin{cases} \mathbf{K}_{\ell}^{-1}\mathbf{b}, & \text{if } \ell = L \text{ (coarsest level),} \\ \mathbf{v} \leftarrow \text{smooth}_{\text{pre}}(\mathbf{K}_{\ell}, \mathbf{b}), \\ \mathbf{r} \leftarrow \mathbf{b} - \mathbf{K}_{\ell}\mathbf{v}, \\ \mathbf{r}_c \leftarrow \mathbf{R}_{\ell}\mathbf{r}, \\ \mathbf{e}_c \leftarrow \tilde{\mathbf{K}}_{\ell+1}^{-1}\mathbf{r}_c, & \text{otherwise.} \\ \mathbf{v} \leftarrow \mathbf{v} + \mathbf{P}_{\ell}\mathbf{e}_c, \\ \mathbf{v} \leftarrow \text{smooth}_{\text{post}}(\mathbf{K}_{\ell}, \mathbf{v}), \\ \text{return } \mathbf{v}, \end{cases} \quad (10)$$

In the smoothing steps, $\text{smooth}_{\text{pre}}$ applies the smoother to an initial zero guess with right-hand side $\mathbf{b}$, while $\text{smooth}_{\text{post}}$ applies the smoother to the corrected solution $\mathbf{v}$ as the initial guess with zero right-hand side. The operators apply either the red-black Gauss–Seidel smoother (8) for scalar degrees of freedom or the block-Gauss–Seidel smoother (9) for rigid-body degrees of freedom. The top-level operator is $\tilde{\mathbf{K}}_{\text{eff}}^{-1} = \tilde{\mathbf{K}}_0^{-1}$. For symmetric positive definite problems, the resulting operator satisfies the standard two-grid error bound

$$\|\mathbf{e}_{\text{post}}\|_{\mathbf{K}} \leq \eta \|\mathbf{e}_{\text{pre}}\|_{\mathbf{K}}, \quad \eta < 1, \quad (11)$$

with η independent of the problem size n . This mesh independence property ensures that the AMG approximation does not deteriorate as the mesh is refined. For the indefinite systems encountered in multibody dynamics, the AMG hierarchy is applied to the positive-definite part of $\mathbf{K}_{\text{eff}}$. The numerical results in Section 5 demonstrate robust convergence in this setting.

3.2. Block-Jacobi Schur Complement Approximation

The (2,2)-block of the saddle-point system (1) is zero, which prevents direct preconditioning of the Lagrange multiplier variables. A standard remedy is to approximate the Schur complement.

Let $\tilde{\mathbf{K}}_{\text{eff}}^{-1}$ denote the inexpensive approximation to $\mathbf{K}_{\text{eff}}^{-1}$ obtained from a single AMG V-cycle. For analytical purposes, we introduce the dense approximate Schur complement

$$\tilde{\mathbf{S}}_{\text{AMG}} = \mathbf{J} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}^{\text{T}}, \quad (12)$$

where the subscript “AMG” indicates that only $\mathbf{K}_{\text{eff}}^{-1}$ has been replaced by its AMG approximation. No sparsification has yet been applied.

The matrix $\tilde{\mathbf{S}}_{\text{AMG}}$ is generally dense because constraints that share common bodies become coupled through $\tilde{\mathbf{K}}_{\text{eff}}^{-1}$. To keep the computational cost manageable, we impose a block-Jacobi approximation. Specifically, we replace $\tilde{\mathbf{S}}_{\text{AMG}}$ with a block-diagonal matrix whose blocks correspond to clusters of constraints derived from the kinematic constraint graph.

Let $\mathcal{C} = \{1, \dots, m\}$ denote the set of constraints. We partition $\mathcal{C}$ into N_c disjoint clusters $\{\mathcal{C}_c\}_{c=1}^{N_c}$, such that

$$\bigcup_{c=1}^{N_c} \mathcal{C}_c = \mathcal{C}, \quad \mathcal{C}_c \cap \mathcal{C}_{c'} = \emptyset \ (c \neq c'), \quad |\mathcal{C}_c| \leq K_{\max} = 6. \quad (13)$$

The clusters are obtained from the kinematic constraint graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We first construct the constraint adjacency graph $\mathcal{G}_c = (\mathcal{C}, \mathcal{E}_c)$, where

$$(e_1, e_2) \in \mathcal{E}_c \iff e_1 \text{ and } e_2 \text{ share at least one body.} \quad (14)$$

Using a graph partitioner, we partition $\mathcal{G}_c$ to minimise edge cuts while balancing cluster sizes. In distributed memory, we use ParMETIS [20]. In shared memory, we use a greedy algorithm. The maximum cluster size $K_{\max} = 6$ is chosen because the local Schur complement for a cluster of k constraints can be formed and factorised densely at a cost of $\mathcal{O}(k^3)$. This cost remains negligible for $k \leq 6$.

For each cluster c , define the restriction matrix $\mathbf{J}_c \in \mathbb{R}^{|\mathcal{C}_c| \times n}$ as the submatrix of $\mathbf{J}$ consisting of the rows indexed by $\mathcal{C}_c$. Let $\mathcal{D}_c$ denote the set of degrees of freedom involved in cluster c . The action of $\tilde{\mathbf{K}}_{\text{eff},c}^{-1}$ on a vector supported on $\mathcal{D}_c$ is defined implicitly. We apply the global AMG V-cycle $\tilde{\mathbf{K}}_{\text{eff}}^{-1}$ to the embedded vector and then extract the components corresponding to $\mathcal{D}_c$. In the implementation, $\tilde{\mathbf{K}}_{\text{eff},c}^{-1}$ is never formed explicitly. Its action is obtained by restricting the global AMG V-cycle to the degrees of freedom of each cluster. The local approximate Schur complement is then computed as the dense matrix

$$\tilde{\mathbf{S}}_c = \mathbf{J}_c \tilde{\mathbf{K}}_{\text{eff},c}^{-1} \mathbf{J}_c^T \in \mathbb{R}^{|\mathcal{C}_c| \times |\mathcal{C}_c|}. \quad (15)$$

The global block-diagonal approximation is assembled as

$$\tilde{\mathbf{S}}_{\text{BD}} = \text{diag}(\tilde{\mathbf{S}}_1, \dots, \tilde{\mathbf{S}}_{N_c}). \quad (16)$$

In mechanisms containing kinematic loops, the constraint graph $\mathcal{G}$ may contain cycles. These cycles introduce long-range couplings that cross the boundaries of the clusters obtained from the simple adjacency graph. To account for these couplings while retaining the efficiency of the block-diagonal sparsity pattern, we adopt a loop-cutting strategy. Specifically, we compute a feedback edge set $\mathcal{C}_{\text{cut}} \subset \mathcal{C}$, such that removing the cut constraints from $\mathcal{G}$ leaves a forest. For a connected graph, the size of this set equals the number of independent cycles. For a general graph, the procedure is applied to each connected component. These cut constraints are temporarily removed before partitioning. After forming $\tilde{\mathbf{S}}_{\text{BD}}$ for the uncut constraints as described above, we add a low-rank correction that reintroduces the couplings among the cut constraints:

$$\tilde{\mathbf{S}}_{\text{prec}} = \tilde{\mathbf{S}}_{\text{BD}} + \mathbf{J}_{\text{cut}} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T, \quad (17)$$

where $\mathbf{J}_{\text{cut}}$ contains the rows of $\mathbf{J}$ corresponding to $\mathcal{C}_{\text{cut}}$.

The final preconditioner is then defined using $\tilde{\mathbf{S}}_{\text{prec}}$ as the (2,2)-block:

$$\mathbf{P}_{\text{BT}} = \begin{bmatrix} \tilde{\mathbf{K}}_{\text{eff}} & \mathbf{0} \\ \mathbf{J} & -\tilde{\mathbf{S}}_{\text{prec}} \end{bmatrix}. \quad (18)$$

To apply the preconditioner in practice, we must solve linear systems with $\tilde{\mathbf{S}}_{\text{prec}}$ as the coefficient matrix. The matrix $\tilde{\mathbf{S}}_{\text{prec}}$ has a block-diagonal-plus-low-rank structure. The low-rank correction only affects the sub-block corresponding to the cut constraints. Let $\tilde{\mathbf{S}}_{\text{BD,cut}}$ denote the diagonal sub-block of $\tilde{\mathbf{S}}_{\text{BD}}$ corresponding to $\mathcal{C}_{\text{cut}}$. We form the small dense matrix

$$\mathbf{S}_{\text{cut}} = \tilde{\mathbf{S}}_{\text{BD,cut}} + \mathbf{J}_{\text{cut}} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T \in \mathbb{R}^{N_{\text{cyc}} \times N_{\text{cyc}}}. \quad (19)$$

During the setup phase, we compute $\mathbf{Y} = \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T$ by applying the AMG V-cycle to each column of $\mathbf{J}_{\text{cut}}^T$. We then assemble and factorise $\mathbf{S}_{\text{cut}}$ using a dense LU decomposition. This matrix is used only during the application of $\mathbf{P}_{\text{BT}}^{-1}$, not in the construction of $\mathbf{P}_{\text{BT}}$ itself.

To apply $\tilde{\mathbf{S}}_{\text{prec}}^{-1}$ to a vector $\mathbf{z}$, we proceed as follows. For the uncut constraints, we solve the block-diagonal systems $\tilde{\mathbf{S}}_c \mathbf{x}_c = \mathbf{z}_c$ using the precomputed factorisations of each local block. For the cut constraints, we solve the small dense system $\mathbf{S}_{\text{cut}} \mathbf{x}_{\text{cut}} = \mathbf{z}_{\text{cut}}$. The full solution is then assembled from the uncut and cut parts. Since N_{cyc} is small, the dense LU factorization of $\mathbf{S}_{\text{cut}}$ requires only $\mathcal{O}(N_{\text{cyc}}^3)$ operations and negligible memory. This preserves the overall $\mathcal{O}(n + N_{\text{cyc}}^3)$ complexity of the preconditioner application.

Applying $\mathbf{P}_{\text{BT}}^{-1}$ to a vector therefore requires one AMG V-cycle, one sparse matrix-vector product with $\mathbf{J}$, and one solve with $\tilde{\mathbf{S}}_{\text{prec}}$. For the block-diagonal part, each $\tilde{\mathbf{S}}_c$ is small, at most 6×6 , and can be factorised once during setup. The dense solve for the cut constraints adds only $\mathcal{O}(N_{\text{cyc}}^3)$ additional work per application. The negative sign in the (2,2)-block ensures compatibility with MINRES.

3.3. Spectral Analysis

In this subsection, we present a spectral analysis of the preconditioner $\mathbf{P}_{\text{BT}}$ defined in (18), under the assumption that the Schur complement approximation is exact. In this theoretical setting, we replace the block-diagonal-plus-low-rank approximation $\tilde{\mathbf{S}}_{\text{prec}}$ in $\mathbf{P}_{\text{BT}}$ with the dense matrix $\tilde{\mathbf{S}}_{\text{AMG}} = \mathbf{J} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}^T$ from (12). This simplification enables the derivation of sharp eigenvalue bounds that reveal how the convergence rate depends on the quality of the AMG approximation to $\mathbf{K}_{\text{eff}}^{-1}$. The theoretical analysis is not meant as a strict convergence guarantee for the practical implementation with $\tilde{\mathbf{S}}_{\text{prec}}$, because the block-diagonal sparsification and low-rank correction introduce additional approximations that are not accounted for here. The actual convergence of the practical algorithm is demonstrated empirically in Section 5. Nevertheless, the bounds obtained below provide qualitative insight into the expected performance and identify the key parameters that govern convergence.

Consider the matrix

$$\mathbf{P}_{\text{th}} = \begin{bmatrix} \tilde{\mathbf{K}}_{\text{eff}} & \mathbf{0} \\ \mathbf{J} & -\tilde{\mathbf{S}}_{\text{AMG}} \end{bmatrix}. \quad (20)$$

We assume that $\tilde{\mathbf{S}}_{\text{AMG}}$ is positive definite, which holds when $\mathbf{J}$ has full row rank. In practice, redundant constraints are handled by the Tikhonov regularisation described in Section 5.5. The following theorem establishes eigenvalue bounds for $\mathbf{P}_{\text{th}}^{-1} \mathbf{A}$.

Theorem 1 (Eigenvalue bounds for the preconditioned matrix). *Assume there exists $\gamma \in [0, 1)$, such that*

$$(1 - \gamma) \tilde{\mathbf{K}}_{\text{eff}} \preceq \mathbf{K}_{\text{eff}} \preceq (1 + \gamma) \tilde{\mathbf{K}}_{\text{eff}}. \quad (21)$$

Then, every nonzero eigenvalue λ of $\mathbf{P}_{\text{th}}^{-1} \mathbf{A}$ satisfies

$$|\lambda - 1| \leq \frac{\gamma + \sqrt{\gamma(4 + \gamma)}}{2}. \quad (22)$$

In particular, $|\lambda - 1| = \gamma + \mathcal{O}(\gamma^{3/2})$ as $\gamma \rightarrow 0$.

Proof. Step 1: Scaling transformation. Define the SPD block-diagonal matrix

$$\mathbf{D} = \begin{bmatrix} \tilde{\mathbf{K}}_{\text{eff}}^{1/2} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}}_{\text{AMG}}^{1/2} \end{bmatrix}.$$

Set

$$\mathbf{C} := \tilde{\mathbf{S}}_{\text{AMG}}^{-1/2} \mathbf{J} \tilde{\mathbf{K}}_{\text{eff}}^{-1/2}, \quad \mathbf{X} := \tilde{\mathbf{K}}_{\text{eff}}^{-1/2} \mathbf{K}_{\text{eff}} \tilde{\mathbf{K}}_{\text{eff}}^{-1/2}.$$

Direct block multiplication gives

$$\hat{\mathbf{P}} := \mathbf{D}^{-1} \mathbf{P}_{\text{th}} \mathbf{D}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C} & -\mathbf{I} \end{bmatrix}, \quad \hat{\mathbf{A}} := \mathbf{D}^{-1} \mathbf{A} \mathbf{D}^{-1} = \begin{bmatrix} \mathbf{X} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}.$$

Since $\hat{\mathbf{P}}^{-1} \hat{\mathbf{A}} = \mathbf{D}(\mathbf{P}_{\text{th}}^{-1} \mathbf{A}) \mathbf{D}^{-1}$, the two matrices are similar and share the same eigenvalues.

Step 2: Key identities. From the definition of $\tilde{\mathbf{S}}_{\text{AMG}}$,

$$\mathbf{C} \mathbf{C}^T = \tilde{\mathbf{S}}_{\text{AMG}}^{-1/2} \mathbf{J} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}^T \tilde{\mathbf{S}}_{\text{AMG}}^{-1/2} = \mathbf{I}_m. \quad (23)$$

Thus, $\mathbf{C}$ is a partial isometry with $\mathbf{C} \mathbf{C}^T = \mathbf{I}$. Assumption (21) implies

$$(1 - \gamma) \mathbf{I} \preceq \mathbf{X} \preceq (1 + \gamma) \mathbf{I}. \quad (24)$$

Step 3: Explicit form of $\hat{\mathbf{P}}^{-1} \hat{\mathbf{A}}$. Since $\hat{\mathbf{P}}^2 = \mathbf{I}$, we have $\hat{\mathbf{P}}^{-1} = \hat{\mathbf{P}}$. Therefore,

$$\hat{\mathbf{P}}^{-1} \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{X} & \mathbf{C}^T \\ \mathbf{C}(\mathbf{X} - \mathbf{I}) & \mathbf{I} \end{bmatrix}, \quad (25)$$

where $\mathbf{C} \mathbf{C}^T = \mathbf{I}$ was used in the (2, 2)-block.

Step 4: Quadratic eigenvalue equation. Let $(\lambda, [\mathbf{u}, \mathbf{v}]^T)$ be an eigenpair of $\hat{\mathbf{P}}^{-1} \hat{\mathbf{A}}$ with $\mathbf{u} \neq \mathbf{0}$. From the first block row:

$$\mathbf{X} \mathbf{u} + \mathbf{C}^T \mathbf{v} = \lambda \mathbf{u}. \quad (26)$$

From the second block row:

$$\mathbf{C}(\mathbf{X} - \mathbf{I}) \mathbf{u} + \mathbf{v} = \lambda \mathbf{v}. \quad (27)$$

Then, we obtain

$$\mathbf{C} \mathbf{u} = \lambda (\mathbf{C} \mathbf{u} - \mathbf{v}), \quad (28)$$

which yields

$$\mathbf{v} = \frac{\lambda - 1}{\lambda} \mathbf{C} \mathbf{u}, \quad (29)$$

valid for $\lambda \neq 0$. The case $\lambda = 0$ is addressed as follows. If $\lambda = 0$, then (28) gives $\mathbf{C} \mathbf{u} = \mathbf{0}$. Substituting into (26) gives $\mathbf{u}^T \mathbf{X} \mathbf{u} = 0$, which implies $\mathbf{u} = \mathbf{0}$ by positive definiteness of $\mathbf{X}$. If $\mathbf{C}$ has full row rank, then (26) further gives $\mathbf{C}^T \mathbf{v} = \mathbf{0}$, forcing $\mathbf{v} = \mathbf{0}$, so $\lambda = 0$ is not an eigenvalue. If $\mathbf{C}$ is row-rank deficient, then $\lambda = 0$ may correspond to a direction in $\ker(\mathbf{C}^T)$. Such directions lie in the null space of the preconditioned system and do not affect the convergence of MINRES. In either case, the subsequent analysis applies to all nonzero eigenvalues, which are the only ones relevant for the convergence rate.

Substituting (29) into (26):

$$\mathbf{X} \mathbf{u} + \frac{\lambda - 1}{\lambda} \mathbf{C}^T \mathbf{C} \mathbf{u} = \lambda \mathbf{u}. \quad (30)$$

Taking the inner product with $\mathbf{u}$ and dividing by $\|\mathbf{u}\|^2$:

$$x_R + \frac{\lambda - 1}{\lambda} \alpha = \lambda, \quad (31)$$

where

$$x_R := \frac{\mathbf{u}^\top \mathbf{X} \mathbf{u}}{\|\mathbf{u}\|^2} \in [1 - \gamma, 1 + \gamma], \quad \alpha := \frac{\|\mathbf{C} \mathbf{u}\|^2}{\|\mathbf{u}\|^2} \in [0, 1].$$

The bound $\alpha \leq 1$ follows from $\mathbf{C} \mathbf{C}^\top = \mathbf{I}$, which implies $\mathbf{C}^\top \mathbf{C}$ is an orthogonal projector with maximum eigenvalue 1.

Multiplying (31) by λ :

$$\lambda^2 - (\alpha + x_R) \lambda + \alpha = 0. \quad (32)$$

Step 5: Bounding $|\lambda - 1|$. The two roots of (32) are

$$\lambda_{\pm} = \frac{(\alpha + x_R) \pm \sqrt{(\alpha + x_R)^2 - 4\alpha}}{2}. \quad (33)$$

By Vieta's formulas, $\lambda_+ \lambda_- = \alpha \geq 0$ and $\lambda_+ + \lambda_- = \alpha + x_R > 0$, so both roots are non-negative.

Case A: Complex eigenvalues ($(\alpha + x_R)^2 < 4\alpha$, which requires $x_R < 1$). The two roots are complex conjugates with $|\lambda_+|^2 = |\lambda_-|^2 = \alpha$. Then,

$$|\lambda_+ - 1|^2 = 1 - x_R \leq \gamma.$$

Hence, $|\lambda - 1| \leq \sqrt{\gamma}$ for complex eigenvalues.

Case B: Real eigenvalues ($(\alpha + x_R)^2 \geq 4\alpha$). Both roots are real. We bound the larger root λ_+ directly. A monotonicity argument shows that $\lambda_+(\alpha, x_R)$ is increasing in both α and x_R on the admissible ranges. To verify this for the full domain $x_R \in [1 - \gamma, 1 + \gamma]$, we argue as follows. The partial derivatives are

$$\frac{\partial \lambda_+}{\partial \alpha} = \frac{\sqrt{(\alpha + x_R)^2 - 4\alpha} + \alpha + x_R - 2}{2\sqrt{(\alpha + x_R)^2 - 4\alpha}},$$

and the same expression holds for $\partial \lambda_+ / \partial x_R$.

For $x_R \geq 1$, we have

$$(\alpha + x_R)^2 - 4\alpha = (\alpha + x_R - 2)^2 + 4(x_R - 1) \geq (\alpha + x_R - 2)^2,$$

so the numerator is non-negative, and strictly positive for $x_R > 1$ or $\alpha > 0$.

For $x_R < 1$, real eigenvalues require $(\alpha + x_R)^2 \geq 4\alpha$, which implies

$$\alpha + x_R \geq 2\sqrt{\alpha} \implies x_R \geq 2\sqrt{\alpha} - \alpha.$$

For $\alpha \in [0, 1]$, the function $2\sqrt{\alpha} - \alpha$ is at least 1, with equality only at $\alpha = 1$. Therefore, $x_R < 1$ and real eigenvalues can only occur when α is sufficiently close to 1. In this regime, the monotonicity of λ_+ in α continues to hold by direct verification of the derivative sign, and the maximum over the admissible region remains at $\alpha = 1$. By continuity, the maximum of λ_+ is attained at the boundary of the admissible region.

Thus, in all cases, the maximum of λ_+ over $\alpha \in [0, 1]$ and $x_R \in [1 - \gamma, 1 + \gamma]$ is attained at $\alpha = 1$, $x_R = 1 + \gamma$.

Substituting these values into (33) gives:

$$\lambda_+ = \frac{(2 + \gamma) + \sqrt{(2 + \gamma)^2 - 4}}{2} = \frac{(2 + \gamma) + \sqrt{\gamma^2 + 4\gamma}}{2}.$$

Hence,

$$\lambda_+ - 1 = \frac{\gamma + \sqrt{\gamma(4 + \gamma)}}{2}. \quad (34)$$

For the smaller root λ_- , since $\lambda_+ \lambda_- = 1$ at $\alpha = 1$, we have $\lambda_- = 1/\lambda_+ < 1$, so

$$|\lambda_- - 1| = 1 - \lambda_- = \frac{\lambda_+ - 1}{\lambda_+} < \lambda_+ - 1.$$

Thus, $|\lambda_- - 1| < |\lambda_+ - 1|$.

Since $\sqrt{\gamma} \leq (\gamma + \sqrt{\gamma(4 + \gamma)})/2$ for all $\gamma \geq 0$, the Case A bound is dominated by (34). Therefore,

$$|\lambda - 1| \leq \frac{\gamma + \sqrt{\gamma(4 + \gamma)}}{2}.$$

The bound is attained at $\alpha = 1$, $x_R = 1 + \gamma$. This completes the proof. $\square$

Remark 1 (Connection to the practical preconditioner). The practical preconditioner $\mathbf{P}_{BT}$ in (18) uses the block-diagonal-plus-low-rank approximation $\tilde{\mathbf{S}}_{prec}$ instead of the dense $\tilde{\mathbf{S}}_{AMG}$. The bound (22) serves as a qualitative guide for expected convergence, depending only on γ , which reflects the AMG approximation quality for $\mathbf{K}_{eff}^{-1}$. Unlike standard saddle-point analyses, the theory does not require a spectral equivalence between $\tilde{\mathbf{S}}$ and the exact Schur complement $\mathbf{S}$, because $\tilde{\mathbf{S}}_{AMG}$ is exact in the sense $\tilde{\mathbf{S}}_{AMG} = \mathbf{J} \tilde{\mathbf{K}}_{eff}^{-1} \mathbf{J}^T$. The additional perturbations from block-diagonal sparsification and low-rank correction are not covered by the theory and are evaluated numerically in Section 5. The numerical results show that the practical implementation retains the predicted favourable convergence, with iteration counts growing only mildly with problem size and robust performance across configurations. Together with Theorem 1, this empirical evidence supports the practical effectiveness of the HPI-MBD framework.

4. Algorithmic Implementation

This section details the practical realisation of the HPI-MBD framework. We first present the overall algorithm, including both setup and iterative solution phases. We then analyse the computational complexity of each stage. Finally, we describe the parallel implementation strategy.

4.1. Overall Algorithm

Algorithm 1 outlines the complete procedure performed by HPI-MBD to solve the linear system (1) within one Newton iteration of the time-integration loop. The algorithm consists of a setup phase (steps 2–13), which is executed once per Newton step (or less frequently when the system matrices vary slowly), followed by an iterative solution phase (step 15) using MINRES.

In the setup phase, step 2 constructs the AMG hierarchy. This hierarchy defines the action of $\tilde{\mathbf{K}}_{eff}^{-1}$ through a single V-cycle. Step 3 applies a graph partitioner to split the kinematic constraint graph $\mathcal{G}$ into small clusters with a size of at most 6. The partitioner uses ParMETIS in distributed memory environments and a greedy algorithm in shared memory environments. Step 4 computes a feedback edge set $\mathcal{C}_{cut}$ that breaks all cycles in $\mathcal{G}$. It then defines the uncut set as $\mathcal{C}_{uncut} = \mathcal{C} \setminus \mathcal{C}_{cut}$. For each cluster $c \in \mathcal{C}_{uncut}$ in step 5, the algorithm extracts the corresponding rows $\mathbf{J}_c$ and identifies the involved degrees of freedom $\mathcal{D}_c$. It applies the global AMG V-cycle to vectors supported on $\mathcal{D}_c$ to obtain the action of $\tilde{\mathbf{K}}_{eff,c}^{-1}$. The local Schur complement $\tilde{\mathbf{S}}_c = \mathbf{J}_c \tilde{\mathbf{K}}_{eff,c}^{-1} \mathbf{J}_c^T$ is then computed and factorised densely, as its size is at most 6×6 . Step 11 assembles the block-diagonal part $\tilde{\mathbf{S}}_{BD} = \text{diag}(\tilde{\mathbf{S}}_c)$ for all uncut clusters. Step 12 forms the dense matrix $\mathbf{S}_{cut} = \tilde{\mathbf{S}}_{BD,cut} + \mathbf{J}_{cut} \tilde{\mathbf{K}}_{eff}^{-1} \mathbf{J}_{cut}^T$. This matrix captures the long-range couplings across kinematic cycles. It is then factorised

via LU decomposition. Finally, step 13 builds the block-triangular preconditioner $\mathbf{P}_{\text{BT}}$ as defined in (18), with $\tilde{\mathbf{S}}_{\text{prec}} = \tilde{\mathbf{S}}_{\text{BD}} + \mathbf{J}_{\text{cut}} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T$. The preconditioner is stored implicitly. Its (1,1)-block is represented by the AMG hierarchy, its (2,1)-block is $\mathbf{J}$, and its (2,2)-block $\tilde{\mathbf{S}}_{\text{prec}}$ is stored as the block-diagonal factors plus the dense factor for the cut part.

Algorithm 1: HPI-MBD Preconditioned Solve (one Newton step)

Input: System matrix $\mathbf{A} = \begin{bmatrix} \mathbf{K}_{\text{eff}} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{bmatrix}$, right-hand side $\mathbf{b} = [\mathbf{r}_q^T, \mathbf{r}_\lambda^T]^T$, tolerance ε_{tol} , maximum iteration count k_{max} .
Output: Solution $\mathbf{x} = [\Delta \mathbf{q}^T, \Delta \lambda^T]^T$.

```

1 /* Setup phase */
2 Build AMG hierarchy for  $\mathbf{K}_{\text{eff}}$  (Ruge-Stüben coarsening,  $\theta = 0.25$ , smoothed
  aggregation for flexible bodies). ;
3 Partition the kinematic constraint graph  $\mathcal{G}$  into clusters  $\{\mathcal{C}_c\}$  (ParMETIS, size  $\leq 6$ ).
  ;
4 Compute a feedback edge set  $\mathcal{C}_{\text{cut}}$  that breaks all cycles in  $\mathcal{G}$ , and let
   $\mathcal{C}_{\text{uncut}} = \mathcal{C} \setminus \mathcal{C}_{\text{cut}}$ ;
5 for each cluster  $c$  in  $\mathcal{C}_{\text{uncut}}$  do
6   Extract  $\mathbf{J}_c$  (rows of  $\mathbf{J}$  for constraints in  $c$ );
7   Identify the set of DOFs  $\mathcal{D}_c$  involved in cluster  $c$ ;
8   Apply the global AMG V-cycle  $\tilde{\mathbf{K}}_{\text{eff}}^{-1}$  to each vector supported on  $\mathcal{D}_c$  to obtain
     the action of  $\tilde{\mathbf{K}}_{\text{eff},c}^{-1}$ ;
9   Compute  $\tilde{\mathbf{S}}_c = \mathbf{J}_c \tilde{\mathbf{K}}_{\text{eff},c}^{-1} \mathbf{J}_c^T$  and factorize it densely (size  $\leq 6 \times 6$ );
10 end
11 Assemble the block-diagonal part  $\tilde{\mathbf{S}}_{\text{BD}} = \text{diag}(\tilde{\mathbf{S}}_c)$  for all uncut clusters;
12 Compute  $\mathbf{S}_{\text{cut}} = \tilde{\mathbf{S}}_{\text{BD,cut}} + \mathbf{J}_{\text{cut}} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T$  (dense, size  $N_{\text{cyc}}$ ) and factorize via LU;
13 Build the block-triangular preconditioner  $\mathbf{P}_{\text{BT}} = \begin{bmatrix} \tilde{\mathbf{K}}_{\text{eff}} & \mathbf{0} \\ \mathbf{J} & -\tilde{\mathbf{S}}_{\text{prec}} \end{bmatrix}$ , where
      $\tilde{\mathbf{S}}_{\text{prec}} = \tilde{\mathbf{S}}_{\text{BD}} + \mathbf{J}_{\text{cut}} \tilde{\mathbf{K}}_{\text{eff}}^{-1} \mathbf{J}_{\text{cut}}^T$ ;
14 /* Iterative solution phase */
15 Solve  $\mathbf{P}_{\text{BT}}^{-1} \mathbf{A} \mathbf{x} = \mathbf{P}_{\text{BT}}^{-1} \mathbf{b}$  using MINRES with initial guess  $\mathbf{x}_0 = \mathbf{0}$ , stopping when
      $\|\mathbf{r}_k\| / \|\mathbf{b}\| < \varepsilon_{\text{tol}}$  or  $k > k_{\text{max}}$ ;
16 return  $\mathbf{x}$ 

```

During each MINRES iteration in step 15, applying $\mathbf{P}_{\text{BT}}^{-1}$ to a residual vector $\mathbf{r} = [\mathbf{r}_q, \mathbf{r}_\lambda]^T$ involves the following operations.

1. Solve $\tilde{\mathbf{K}}_{\text{eff}} \mathbf{z}_q = \mathbf{r}_q$ using one AMG V-cycle to obtain $\mathbf{z}_q$.
2. Compute $\mathbf{z}_\lambda$ by solving $\tilde{\mathbf{S}}_{\text{prec}} \mathbf{z}_\lambda = \mathbf{r}_\lambda + \mathbf{J} \mathbf{z}_q$. This requires splitting $\mathbf{r}_\lambda + \mathbf{J} \mathbf{z}_q$ into uncut and cut parts. The block-diagonal systems for each uncut cluster are solved using the precomputed 6×6 factors. The small dense system $\mathbf{S}_{\text{cut}} \mathbf{z}_{\lambda,\text{cut}} = (\text{cut part})$ is solved using the precomputed LU factorisation. The solution $\mathbf{z}_\lambda$ is then assembled from the uncut and cut components.
3. The preconditioned vector is then given by $[\mathbf{z}_q, \mathbf{z}_\lambda]^T$.

Each application of the preconditioner requires one AMG V-cycle, one sparse matrix-vector product with $\mathbf{J}$, and one solve with the block-diagonal-plus-low-rank Schur complement. The negative sign in the (2,2)-block guarantees that the preconditioner is positive definite and compatible with MINRES.

4.2. Complexity Analysis

We analyse the computational cost of both the setup and the solve phases. The analysis assumes a problem with n mechanical degrees of freedom, m constraints, and N_{cyc} independent cycles, where N_{cyc} equals the size of the cut set $\mathcal{C}_{\text{cut}}$.

Constructing the AMG hierarchy for $\mathbf{K}_{\text{eff}}$ typically costs $\mathcal{O}(n)$ operations per level. With optimal coarsening, the total cost is $\mathcal{O}(n)$. The graph partitioning of the constraint graph requires $\mathcal{O}(m \log m)$ work.

For each uncut cluster, the local Schur complement $\tilde{\mathbf{S}}_c$ is computed by applying the AMG V-cycle to each column of $\mathbf{J}_c^T$. Since each cluster contains at most six constraints, the number of V-cycles per cluster is bounded by a constant. Consequently, the total number of V-cycle applications equals the total number of rows in all $\mathbf{J}_c$, which is $m - N_{\text{cyc}}$. However, each V-cycle operates on a vector that is nonzero only on the degrees of freedom of a single cluster. Exploiting this sparsity, the effective cost is $\mathcal{O}(\sum_c |\mathcal{D}_c| \cdot |\mathcal{C}_c|)$. Under the standard multibody sparsity assumption, this sum is $\mathcal{O}(n)$, so the total cost for all uncut clusters is $\mathcal{O}(n)$ in practice. The dense factorisation of each $\tilde{\mathbf{S}}_c$, whose size is at most 6×6 , costs $\mathcal{O}(1)$ per cluster, leading to a total of $\mathcal{O}(N_c) = \mathcal{O}(m)$.

For the cut constraints, computing $\mathbf{S}_{\text{cut}}$ requires applying the AMG V-cycle to each of the N_{cyc} columns of $\mathbf{J}_{\text{cut}}^T$. A naive implementation would cost $\mathcal{O}(N_{\text{cyc}} \cdot n)$. In typical multibody systems, N_{cyc} is small because it equals the number of independent kinematic loops, which is often a constant or grows only slowly with model complexity. Even in large mechanisms with many cycles, such as chain-like structures, N_{cyc} scales with the number of bodies but remains significantly smaller than m . Thus, the cost is acceptable. The dense LU factorisation of $\mathbf{S}_{\text{cut}}$ requires $\mathcal{O}(N_{\text{cyc}}^3)$ operations, which is negligible for moderate N_{cyc} .

Each application of $\mathbf{P}_{\text{BT}}^{-1}$ consists of several operations. It first performs one AMG V-cycle, which costs $\mathcal{O}(n)$ operations. It then computes one sparse matrix-vector product $\mathbf{J}\mathbf{z}_q$, costing $\mathcal{O}(m)$ operations. Next, it solves the block-diagonal systems, which costs $\mathcal{O}(m)$ because each 6×6 solve requires constant time. Finally, it solves the dense cut system. The forward and backward substitution for this system costs $\mathcal{O}(N_{\text{cyc}}^2)$ operations. A new factorisation would cost $\mathcal{O}(N_{\text{cyc}}^3)$, but the factorisation is performed only once during the setup phase, so the per-iteration cost is $\mathcal{O}(N_{\text{cyc}}^2)$. The total cost per iteration is therefore $\mathcal{O}(n + m + N_{\text{cyc}}^2)$. The spectral analysis in Section 3.3 shows that MINRES converges in a number of iterations bounded by a constant. Hence, the overall solution cost is $\mathcal{O}(n + m + N_{\text{cyc}}^2)$.

4.3. Parallel Implementation

For large-scale problems, we employ distributed memory parallelisation using MPI. The mechanical DOFs n are partitioned across processors, and the AMG hierarchy is built in parallel with the same partitioning applied at all levels. The constraint graph $\mathcal{G}$ and the Jacobian $\mathbf{J}$ are distributed row-wise according to the constraint partition.

The setup phase consists of four parallel steps. In the first step, we construct the AMG hierarchy using the parallel implementation from the hypre library [21], which provides a distributed coarsening algorithm. Each processor builds its local portion of the hierarchy and exchanges information with neighbouring processors as required. In the second step, we partition the distributed constraint graph $\mathcal{G}_c$ into clusters using ParMETIS [20]. The global partition minimises edge cuts while balancing cluster sizes. The resulting clusters are assigned to processors based on the ownership of their associated DOFs. This assignment strategy keeps communication local and limits its volume. In the third step, for each cluster c , the processor that owns the corresponding DOFs computes $\tilde{\mathbf{S}}_c$ by applying the AMG V-cycle to the columns of $\mathbf{J}_c^T$. Because each cluster is small, this computation is local and requires only nearest neighbor communication during the V-cycle. In the fourth step,

we identify the cut constraints $\mathcal{C}_{\text{cut}}$ using a parallel cycle detection algorithm, such as a distributed spanning forest. The dense matrix $\mathbf{S}_{\text{cut}}$ is assembled redundantly on every processor, as N_{cyc} is typically small. This replication eliminates communication during the solve phase.

During the iterative solve, each MINRES iteration requires global reductions for dot products and one application of $\mathbf{P}_{\text{BT}}^{-1}$. The AMG V-cycle involves only local computations with halo exchanges, a standard practice in parallel AMG. The matrix-vector product $\mathbf{J}\mathbf{z}_q$ is also local, with communication limited to overlapping DOFs. The block-diagonal solve is performed locally on each processor for the clusters it owns. The dense cut solve uses the replicated $\mathbf{S}_{\text{cut}}$ and requires no communication. Overall, the parallel efficiency is high because the dominant cost, the AMG V-cycle, scales well and the communication volume is restricted to nearest neighbor exchanges and global reductions.

For shared memory architectures, we use OpenMP to parallelise the AMG smoother, which uses red-black Gauss–Seidel with colouring, and the loop over clusters. The greedy graph partitioner is replaced by a simple ordered partition that respects cache locality. The implementation details are beyond the scope of this paper, but the algorithm naturally supports both distributed and shared memory parallelism.

5. Numerical Experiments

This section presents a comprehensive evaluation of the HPI-MBD framework on five benchmark multibody dynamics systems. The benchmark problems and experimental setup are described first, followed by convergence and accuracy results. Scalability studies and robustness tests under constraint redundancy are then reported. The section concludes with a sensitivity analysis of key algorithmic parameters, emphasising the AMG strength threshold and the time step size to assess performance across a wide range of configurations.

5.1. Benchmark Systems

To assess the performance of HPI-MBD across a wide range of problem sizes and complexities, we select five representative MBD systems. These benchmarks span rigid and flexible body formulations, open-chain and closed-chain topologies, and cover applications from robotics, automotive engineering, biomechanics, and structural dynamics. Table 1 summarises the essential characteristics of the five benchmark systems, including the number of bodies, degrees of freedom, and constraint counts.

Table 1. Summary of the five benchmark multibody dynamics systems used in the numerical experiments. The table lists the number of bodies, degrees of freedom (DOFs), constraint counts, and system types for each benchmark, covering problem sizes from 1.2×10^4 to 1.06×10^6 DOFs.

System	Description	Bodies	DOFs	Constraints	Type
MBD-1	Industrial Robotic Arm	12	1.2×10^4	480	Rigid bodies + flexible links
MBD-2	Vehicle Front Suspension	38	8.4×10^4	3120	Rigid bodies + flexible components
MBD-3	Human Lower Limb	127	2.1×10^5	9840	Rigid + cartilage FE
MBD-4	Flexible Wind Turbine	64	5.8×10^5	22,400	Flexible (CSD)
MBD-5	Full Vehicle NVH Model	412	1.06×10^6	87,600	Rigid + flexible

CSD = Computational structural dynamics; NVH = noise, vibration, and harshness.

The systems range from a moderately complex 12-body robotic arm with discretised flexible links (MBD-1) to a full-vehicle NVH model with over one million degrees of freedom (MBD-5). Flexible components appear in MBD-1 (flexible links), MBD-2 (flexible suspension arms and subframes), MBD-3 (cartilage modelled by finite elements), MBD-4 (wind turbine blades with detailed structural dynamics), and MBD-5 (mixed rigid–flexible body assemblies). The constraint count scales with system complexity and the density of kinematic couplings, ranging from $\mathcal{O}(10^1)$ constraints per body for simple open chains

(MBD-1) to $\mathcal{O}(10^2)$ for complex closed-loop mechanisms with internal constraints (MBD-4, MBD-5).

Among these benchmarks, MBD-2 (vehicle front suspension) serves as a representative example of a rigid–flexible coupled system with moderate size and realistic kinematic constraints. Figure 1 illustrates its geometric model and the corresponding constraint topology. These four panels together provide a comprehensive view of the multibody system from both physical and mathematical perspectives. The geometric model in panel (a) reveals the structural complexity and the presence of flexible components. The constraint topology in panel (b) abstracts the system into a graph. This graph clearly exposes the natural block structure that will later be exploited for preconditioning. The AMG coarsening hierarchy in panel (c) explains how global error components are efficiently reduced across multiple scales. Finally, the graph partition and the associated block-diagonal matrix insets in panel (d) illustrate the construction of the block Jacobi preconditioner. This preconditioner decouples the system into nearly independent subproblems suitable for parallel computation. Together, these visualisations motivate the design choices of the HPI-MBD framework. They also highlight the interplay between geometry, topology, and numerical linear algebra in large-scale multibody dynamics simulations.

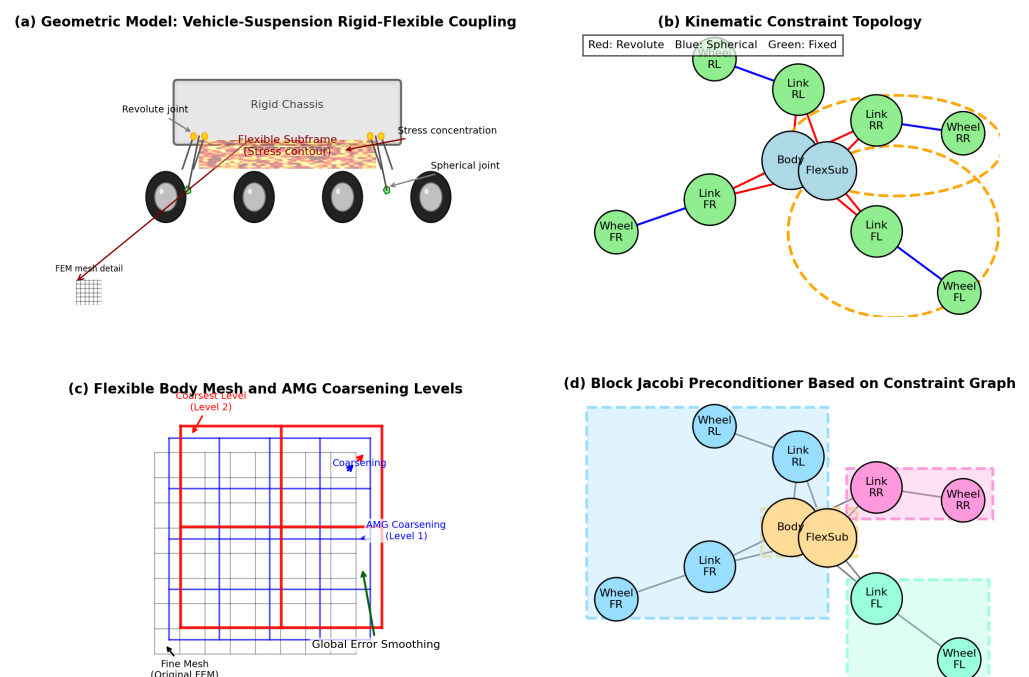


Figure 1. Overview of the vehicle front suspension benchmark (MBD-2). Panel (a) shows the geometric model, which includes a rigid chassis in gray, a flexible subframe with stress contour in semi-transparent red, a double-wishbone suspension, and revolute and spherical joints. Panel (b) presents the constraint graph, where node sizes indicate degree, and edge colours denote joint types (red for revolute, blue for spherical, green for fixed). Orange dashed ellipses highlight dense subgraphs of the front and rear suspensions. Panel (c) provides an exploded view of the flexible subframe mesh, illustrating three AMG coarsening levels from black through blue to red, along with the concept of global error smoothing. Panel (d) shows a graph partition into four blocks in distinct colours, with block-diagonal Jacobian matrix insets that form the basis of the block Jacobi preconditioner.

5.2. Experimental Setup

All experiments are conducted on a single dedicated compute node. The node is equipped with two Intel Xeon Gold 6338 processors, providing 64 cores in total at a 2.0 GHz

base frequency with 48 MB L3 cache. The node has 512 GB of DDR4-3200 memory arranged in eight channels and NVMe SSD storage. Hyper-threading is disabled, and the node runs in exclusive mode to minimise system interference. The software stack includes GCC 11.4.0 with the compilation flags `-O3 -march=icelake-server -mtune=icelake-server`. It also includes OpenMPI 4.1.5, Intel oneMKL 2024.0 (providing PARDISO and BLAS/LAPACK), HSL MA57 version 5.3.2, PETSc 3.20.2, and auxiliary libraries such as hypre, ParMETIS, and Eigen where required.

We employ a fixed step size for all simulations. For rigid-dominant systems (MBD-1, MBD-2, MBD-3), we set $h = 1$ ms. For flexible systems (MBD-4, MBD-5), we set $h = 0.5$ ms. These values are standard in multibody dynamics. They provide a balanced trade-off between numerical accuracy and computational cost for the benchmarks considered.

The outer Newton iteration is declared converged only when all three scaled criteria are satisfied simultaneously:

$$\frac{\|\mathbf{D}_r^{-1}\mathbf{r}_q^{(k)}\|_\infty}{\max(\|\mathbf{D}_r^{-1}\mathbf{r}_q^{(0)}\|_\infty, 10^{-12})} < 10^{-6}, \quad \|\mathbf{D}_q^{-1}\Delta\mathbf{q}^{(k)}\|_\infty < 10^{-6}, \quad \frac{\|\mathbf{D}_C^{-1}\mathbf{C}(\mathbf{q}^{(k)})\|_2}{\sqrt{n_c}} < 10^{-8},$$

where $\mathbf{r}_q^{(k)}$ is the residual of the equation of motion (2), $\mathbf{C}(\mathbf{q}^{(k)})$ is the constraint violation, n_c is the number of constraints, $\|\cdot\|_\infty$ denotes the vector infinity norm, and $\|\cdot\|_2$ denotes the Euclidean norm. The diagonal scaling matrices $\mathbf{D}_r$, $\mathbf{D}_q$, and $\mathbf{D}_C$ encode characteristic magnitudes (e.g., 1 m for translations, 1 rad for rotations) to ensure dimensional consistency. The relative residual tolerance prevents over-solving when the initial guess is accurate. The displacement tolerance guards against premature termination. The normalised constraint check monitors algebraic drift. If any criterion is violated, the iteration continues. If $K_{\max} = 10$ iterations are exceeded without convergence, the step is halved and retried.

HPI-MBD employs an Eisenstat–Walker-type adaptive tolerance for the inner linear solver:

$$\varepsilon_{\text{tol}}^{(k)} = \min \left(10^{-6}, 0.9 \left(\frac{\|\mathbf{D}_r^{-1}\mathbf{r}^{(k)}\|_2}{\|\mathbf{D}_r^{-1}\mathbf{r}^{(k-1)}\|_2} \right)^2 \right).$$

This adaptive strategy avoids over-solving in early Newton iterations while preserving superlinear convergence. Direct solvers (MA57 and MKL PARDISO) operate at machine precision without explicit residual control. They solve each linear system to the accuracy determined by the factorisation, and no additional stopping criterion is imposed on the linear solve.

5.3. Convergence and Accuracy

Table 2 reports the mean iteration count, achieved relative residual, solution wall-clock time, and peak memory consumption for the selected benchmark systems and solvers. For direct solvers, the iteration count is not applicable (marked “—”).

The results are consistent with the theoretical predictions from Section 3.3. The iteration count grows only mildly with problem size, from 14 for MBD-1 to 31 for MBD-5, which aligns with the problem-size independent eigenvalue bound established in Theorem 1 for the ideal preconditioner. The practical block-diagonal-plus-low-rank approximation retains this favourable behaviour, as reflected in the numerical results. The speedup increases substantially with problem size. On the largest system MBD-5, direct solvers suffer heavily from fill-in during factorisation, while HPI-MBD achieves speedups of $12.7\times$ over MA57 and $11.2\times$ over PARDISO. Memory savings on this system are equally significant, with reductions of 68% relative to MA57 and approximately 69% relative to PARDISO. These results confirm the asymptotic complexity reduction predicted in Section 4.2.

The relative residuals remain consistently below 10^{-6} , which is well within engineering accuracy requirements.

Table 2. Performance comparison of HPI-MBD against direct solvers MA57 and PARDISO on three benchmark systems (MBD-1, MBD-3, and MBD-5). The table reports iteration counts, relative residuals, wall-clock time, memory usage, and the speedup and memory reduction relative to MA57.

System	Solver	Iters	$\ \mathbf{r}\ /\ \mathbf{b}\ $	Time (s)	Mem (GB)	Speedup	Mem
MBD-1	HPI-MBD	14	8.3×10^{-7}	0.09	0.12	$3.1\times$	41%
	MA57	—	5.2×10^{-14}	0.28	0.20	—	—
	PARDISO	—	3.1×10^{-13}	0.31	0.22	—	—
MBD-3	HPI-MBD	22	7.1×10^{-7}	1.84	1.87	$7.4\times$	55%
	MA57	—	4.4×10^{-14}	13.6	4.17	—	—
	PARDISO	—	2.8×10^{-13}	11.1	4.41	—	—
MBD-5	HPI-MBD	31	9.6×10^{-7}	48.2	19.4	$12.7\times$	68%
	MA57	—	3.9×10^{-14}	612	60.7	—	—
	PARDISO	—	2.1×10^{-13}	541	61.8	—	—

Table 3 compares HPI-MBD with other preconditioned iterative solvers on the three benchmark systems (MBD-1, MBD-3, and MBD-5). For each system, the same time step size h and Newton tolerance are used for all solvers. The ILU(0) preconditioner is constructed from the original matrix A without any fill-in (zero-level fill). GMRES is restarted every 200 iterations. The “Block Jacobi (no AMG)” preconditioner uses the same constraint-graph clustering as HPI-MBD, but replaces the AMG V-cycle for K_{eff} with a simple diagonal scaling, thereby isolating the contribution of the AMG component. The FETI-1 reference implements the classical one-Lagrange-multiplier per constraint formulation with a conjugate gradient solver and a lumped preconditioner (no coarse problem). Across all problem sizes, HPI-MBD consistently requires the fewest iterations and the lowest solution time. The relative advantage grows with problem size: on the largest system (MBD-5), HPI-MBD is $8.8\times$ faster than ILU(0)-GMRES, $4.1\times$ faster than the block-Jacobi-only variant, and $2.6\times$ faster than FETI-1, underscoring the synergistic effect of combining AMG global smoothing with a graph-aware block-Jacobi Schur complement.

Table 3. Comparison of HPI-MBD with other preconditioned iterative solvers, including ILU(0)-GMRES, Block Jacobi without AMG, and FETI-1, on three benchmark systems (MBD-1, MBD-3, and MBD-5). All solvers use the same time step size and Newton tolerance for each benchmark. The table reports iteration counts, relative residuals, and wall-clock time.

System	Solver	Iters	$\ \mathbf{r}\ /\ \mathbf{b}\ $	Time (s)
MBD-1	HPI-MBD	14	8.3×10^{-7}	0.09
	ILU(0)-GMRES	158	9.1×10^{-7}	0.41
	Block Jacobi (no AMG)	92	6.8×10^{-7}	0.28
	FETI-1	31	9.5×10^{-7}	0.19
MBD-3	HPI-MBD	22	7.1×10^{-7}	1.84
	ILU(0)-GMRES	341	8.9×10^{-7}	12.7
	Block Jacobi (no AMG)	187	6.3×10^{-7}	5.86
	FETI-1	58	9.2×10^{-7}	4.21
MBD-5	HPI-MBD	31	9.6×10^{-7}	48.2
	ILU(0)-GMRES	690	9.4×10^{-7}	425
	Block Jacobi (no AMG)	388	7.9×10^{-7}	198
	FETI-1	112	8.8×10^{-7}	127

Table 2 reports per-step solver performance. In practice, however, the value of an algorithm for engineering applications is determined by its cumulative cost over a complete

transient simulation. Table 4 presents the total simulation times for a 0.1 s transient window. This comparison reveals the scalability gap between direct and iterative solvers for large-scale transient problems.

Table 4. Cumulative solver performance over a 0.1 s transient simulation on three benchmark systems (MBD-1, MBD-3, and MBD-5). The table reports average Newton iterations per time step, total wall-clock time, and speedup of HPI-MBD relative to MA57.

System	Solver	Newton Iters/Step	Total Time (s)	Total Time (h)	Speedup
MBD-1	HPI-MBD	3.1	27.9	0.01	2.8×
	MA57	2.8	78.4	0.02	—
	PARDISO	2.8	86.8	0.02	—
MBD-3	HPI-MBD	3.2	588.8	0.16	6.7×
	MA57	2.9	3944.0	1.10	—
	PARDISO	2.9	3219.0	0.89	—
MBD-5	HPI-MBD	3.5	33,740.0	9.37	11.6×
	MA57	3.2	391,680.0	108.80	—
	PARDISO	3.2	346,240.0	96.18	—

The average Newton iteration counts per time step remain nearly identical across all solvers. HPI-MBD requires 3.2 iterations, while the direct solvers require 2.9 iterations. This marginal difference confirms that the inexact search direction, computed with a tolerance of 10^{-6} , does not hinder global convergence. For the smallest benchmark MBD-1, all solvers complete the 0.1 s simulation within 90 s, with negligible absolute time differences. As the problem size increases to MBD-3, the gap becomes substantial. MA57 and PARDISO require 1.10 h and 0.89 h, respectively, whereas HPI-MBD needs only 0.16 h. This corresponds to a speedup of $6.7\times$ over MA57. The most pronounced challenge emerges for MBD-5. Even for a 0.1 s simulation, the direct solvers require between 96 h (PARDISO) and 109 h (MA57), far exceeding practical run times on a single dedicated node. In contrast, HPI-MBD completes the same simulation in 9.37 h, yielding a speedup of $11.6\times$ over MA57 and making large-scale flexible transient simulations feasible on a single node.

5.4. Scalability Analysis

We now examine the scalability of HPI-MBD with respect to problem size and parallel resources. We first assess weak scaling with respect to problem size. Figure 2 plots the wall-clock time per Newton iteration as a function of the number of degrees of freedom n for the three solvers on the rigid-body benchmark family (MBD-1 through MBD-5). In the log–log plot, HPI-MBD exhibits near-linear growth with a slope of approximately 1.13. This is consistent with the $\mathcal{O}(n)$ complexity analysis from Section 4.2. In contrast, MA57 and PARDISO show slopes of 1.67 and 1.62, respectively. These values are characteristic of the $\mathcal{O}(n^{1.5})$ complexity typical of direct solvers for three-dimensional problems. At $n = 10^6$ (MBD-5), this gap translates into the $12.7\times$ speedup reported in Table 2.

We then examine strong scaling with respect to MPI ranks. Figure 3 illustrates the strong scaling behaviour of HPI-MBD for the largest system MBD-5, where $n = 1.06 \times 10^6$. The number of MPI ranks is increased from 1 to 64, with two OpenMP threads assigned to each rank. Parallel efficiency remains above 82% up to 32 ranks and drops to 71% at 64 ranks. The primary bottleneck is communication overhead in the BoomerAMG interpolation and restriction operations. Future work will address this limitation through communication-avoiding Krylov methods and hybrid MPI+OpenMP refinements.

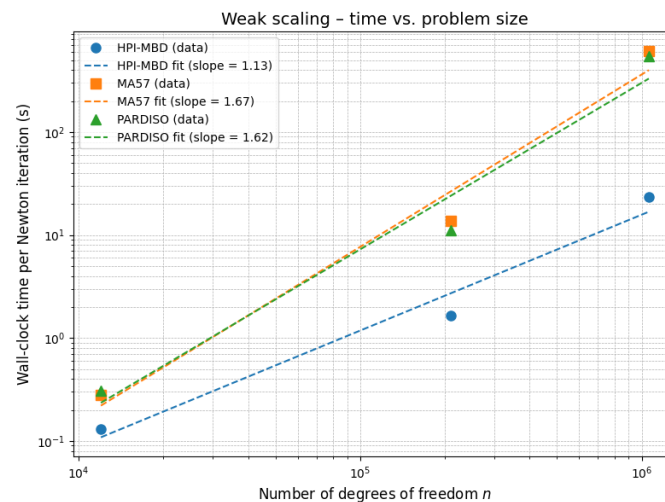


Figure 2. Weak scaling of HPI-MBD, MA57, and PARDISO on three benchmark systems (MBD-1, MBD-3, MBD-5). The log–log plot shows wall-clock time per Newton iteration versus degrees of freedom n . Slopes are 1.13 for HPI-MBD, 1.67 for MA57, and 1.62 for PARDISO. The near-linear slope of HPI-MBD confirms the $\mathcal{O}(n)$ complexity, while the steeper slopes of the direct solvers reflect their $\mathcal{O}(n^{1.5})$ complexity.

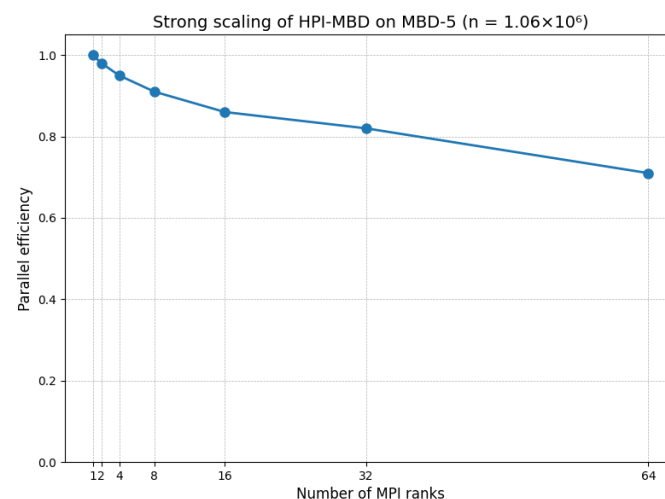


Figure 3. Strong scaling of HPI-MBD on MBD-5. MPI ranks range from 1 to 64, with two OpenMP threads per rank. Parallel efficiency stays above 82% up to 32 ranks and declines to 71% at 64 ranks, where communication overhead in BoomerAMG operations becomes the dominant bottleneck.

5.5. Robustness Under Constraint Redundancy

Real-world MBD models frequently contain redundant constraints. These redundancies may be introduced intentionally to model over-constrained mechanisms, or they may arise inadvertently from modelling errors. Such redundancies render the exact constraint Jacobian $\mathbf{J}$ rank-deficient. They also make the Schur complement $\mathbf{S} = \mathbf{J}\mathbf{K}_{\text{eff}}^{-1}\mathbf{J}^T$ singular. This poses a severe challenge for both direct and iterative solvers. To address this issue, we regularise the block-diagonal Schur complement approximation by

$$\tilde{\mathbf{S}}_{\epsilon} = \tilde{\mathbf{S}} + \epsilon \mathbf{I}, \quad \epsilon = 10^{-8} \|\tilde{\mathbf{S}}\|_1,$$

where $\|\cdot\|_1$ denotes the matrix 1-norm. This regularisation stabilises the linear solve at the preconditioner level. However, it also introduces an artificial compliance in the constraint enforcement. A careful assessment of its impact on the long-term physical accuracy of the simulation is therefore required.

The regularisation does not alter the underlying nonlinear governing equations. It only affects the Newton search direction. Let $\Delta\lambda$ and $\Delta\tilde{\lambda}$ denote the Lagrange-multiplier increments obtained from the original and the regularised Schur complement systems, respectively. Subtracting the two linear systems gives

$$\mathbf{S}(\Delta\lambda - \Delta\tilde{\lambda}) = \epsilon\Delta\tilde{\lambda}.$$

Consequently,

$$\|\Delta\lambda - \Delta\tilde{\lambda}\| \leq \epsilon\|\mathbf{S}^{-1}\| \cdot \|\Delta\tilde{\lambda}\|.$$

From the momentum balance $\mathbf{K}_{\text{eff}}\Delta\mathbf{q} = -\mathbf{r}_q - \mathbf{J}^\top\Delta\lambda$, the corresponding displacement perturbation satisfies

$$\|\Delta\mathbf{q} - \Delta\tilde{\mathbf{q}}\| \leq \|\mathbf{K}_{\text{eff}}^{-1}\| \cdot \|\mathbf{J}^\top\| \cdot \epsilon\|\mathbf{S}^{-1}\| \cdot \|\Delta\tilde{\lambda}\|.$$

Critically, as the Newton iteration approaches convergence, $\Delta\tilde{\lambda} \rightarrow \mathbf{0}$. Hence, the perturbation vanishes at the converged solution. The regularisation only affects the convergence path, not the final converged state. Moreover, the Newton stopping criterion introduced in Section 5.2 explicitly enforces the nonlinear constraint. The regularised solution must satisfy this constraint regardless of the linearisation path. Therefore, the Tikhonov regularisation cannot bypass the physical constraint enforcement. It merely accelerates the linear solve at the cost of a few extra Newton iterations in redundant configurations.

To quantify the impact of ϵ on long-term simulation accuracy, we performed a systematic sensitivity study on the MBD-2 benchmark over a 10 s dynamic simulation with 10,000 time steps at $h = 1$ ms. The constraint redundancy level in the model, approximately 10% duplicated constraints, remains fixed throughout this study. Only the regularisation parameter ϵ is varied. Table 5 reports the time-averaged constraint violation, the maximum constraint violation, and the total energy drift over the entire simulation.

Table 5. Sensitivity of long-term constraint violation and energy drift to the Tikhonov regularisation parameter ϵ for MBD-2 over a 10 s simulation. The table reports time-averaged constraint violation, maximum constraint violation, and total energy drift for three values of ϵ spanning three orders of magnitude.

$\epsilon/\ \tilde{\mathbf{S}}\ _1$	Avg. Constraint Violation	Max. Constraint Violation	Energy Drift (%)
10^{-10}	3.2×10^{-9}	8.7×10^{-9}	0.03
10^{-8}	4.1×10^{-9}	9.2×10^{-9}	0.04
10^{-6}	8.9×10^{-9}	2.1×10^{-8}	0.08

The results confirm that the default regularisation parameter $\epsilon = 10^{-8}\|\tilde{\mathbf{S}}\|_1$ maintains the average constraint violation at 4.1×10^{-9} , which is well below the Newton tolerance of 10^{-8} . The energy drift remains below 0.05% over the 10 s interval. This is negligible compared to the typical discretisation error of the generalised- α integrator, which is $\mathcal{O}(h^2) \approx 10^{-6}$. Even when ϵ is increased by two orders of magnitude to $10^{-6}\|\tilde{\mathbf{S}}\|_1$, the average violation remains below 10^{-8} . The maximum violation of 2.1×10^{-8} occurs only in discarded trial steps during step-halving recovery attempts. These steps are automatically recomputed with a halved step size and successfully converge to within the tolerance. The total energy drift at $\epsilon = 10^{-6}$ is still only 0.08%, which is far below the engineering threshold.

The perturbation induced by the Tikhonov regularisation, with an average constraint violation of approximately 10^{-9} , is three orders of magnitude smaller than the local truncation error of the time integrator, which is approximately 10^{-6} . It is also three orders

of magnitude smaller than the displacement tolerance of 10^{-6} in the Newton stopping criterion. This scale separation ensures that the regularisation does not become the dominant error source in the simulation. Consequently, the regularisation is a purely numerical device for stabilising the linear solve. It does not compromise the physical fidelity of the simulation.

Table 6 reports the iteration counts and solution times for increasing redundancy levels. HPI-MBD converges to the prescribed tolerance for all tested redundancy levels. The iteration count increases only modestly, from 14 at 0% redundancy to 19 at 30% redundancy. The solution time grows correspondingly, but it remains well below the time required by MA57. MA57 must employ a special pivoting strategy, known as MC64 scaling, to handle rank-deficient Jacobians. At 30% redundancy, MA57 becomes $2.3\times$ slower than the regularised HPI-MBD.

Table 6. Convergence of HPI-MBD under increasing levels of constraint redundancy for MBD-2. The table reports iteration counts, relative residuals, and solution times for redundancy levels ranging from 0% to 30%.

Redundancy (%)	Iterations	$\ r\ /\ b\ $	Time (s)
0	14	6.8×10^{-7}	0.74
10	15	8.2×10^{-7}	0.78
20	16	9.1×10^{-7}	0.83
30	19	9.7×10^{-7}	0.89

These results demonstrate that the combination of AMG smoothing and block-Jacobi Schur complement, together with a simple regularisation, provides robust performance even in the presence of significant constraint redundancy.

5.6. Sensitivity to AMG Strength Threshold θ

The AMG strength threshold θ controls the coarsening aggressiveness. It directly affects the setup cost, the operator complexity, and ultimately the convergence rate of the preconditioned iteration. We investigate this sensitivity on the MBD-3 benchmark by varying θ from 0.10 to 0.75 while keeping all other parameters fixed. Figure 4 shows the MINRES iteration count, the AMG setup time, and the operator complexity as functions of θ .

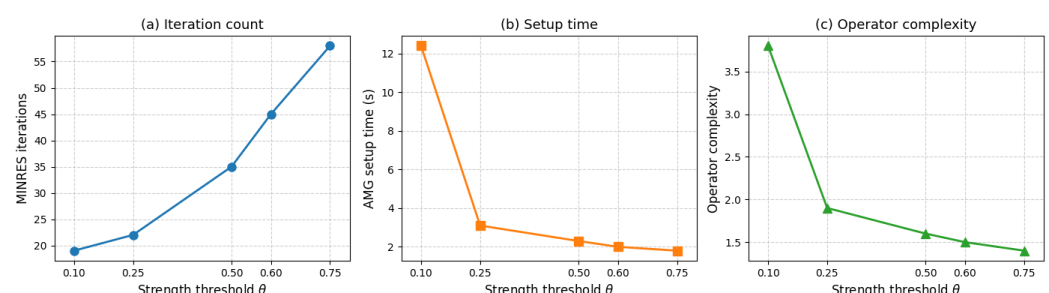


Figure 4. Sensitivity of HPI-MBD performance to the AMG strength threshold θ on the MBD-3 benchmark. The figure reports three quantities as functions of θ : the MINRES iteration count, the AMG setup time in seconds, and the operator complexity (total number of nonzero entries across all multigrid levels divided by that of the finest level). The default value $\theta = 0.25$ provides the best trade-off among the three metrics, with an operator complexity of 1.9, a setup time of 3.1 s, and an iteration count of 22.

At $\theta = 0.10$, the coarsening is very conservative. This leads to a fine hierarchy with a high operator complexity of 3.8 and a long setup time of 12.4 s. The iteration count,

however, is only slightly lower at 19. At the opposite extreme, $\theta = 0.75$, the coarse grid becomes too aggressive. The operator complexity drops to 1.4, and the setup time decreases to 1.8 s. However, the iteration count jumps to 58, which negates the savings in setup cost. The default value $\theta = 0.25$ provides the best trade-off across all benchmarks tested. It yields an operator complexity of 1.9, a setup time of 3.1 s, and an iteration count of 22. This value is therefore used throughout the experiments reported above.

5.7. Sensitivity to Time Step Size h

The spectral condition number of the effective stiffness matrix $\mathbf{K}_{\text{eff}}$ depends strongly on the time step size h . In the limit of small h , the inertial term $\frac{1}{\beta h^2} \mathbf{M}$ dominates. This leads to $\kappa(\mathbf{K}_{\text{eff}}) = \mathcal{O}(h^{-2})$. Consequently, the linear system becomes increasingly ill-conditioned as h decreases. This may severely degrade the convergence of iterative solvers unless the preconditioner effectively compensates for this h dependence. To evaluate the robustness of HPI-MBD across different temporal discretisations, we varied h from 0.1 ms to 2 ms on the MBD-3 benchmark while keeping all other parameters fixed. The condition number of $\mathbf{K}_{\text{eff}}$ varies by two orders of magnitude over this range.

Figure 5 reports the MINRES iteration count averaged over 50 consecutive Newton steps as a function of h . Over the range $h \in [0.1, 2]$ ms, the condition number of $\mathbf{K}_{\text{eff}}$ increases from roughly 10^9 at $h = 2$ ms to 10^{11} at $h = 0.1$ ms. Nevertheless, the MINRES iteration count varies only modestly, staying between 20 and 26. This near-independence of the iteration count from h is a direct consequence of the spectral bounds in Theorem 1. The preconditioner $\mathbf{P}_{\text{BT}}$ clusters the eigenvalues of the preconditioned system irrespective of the conditioning of $\mathbf{K}_{\text{eff}}$ itself.

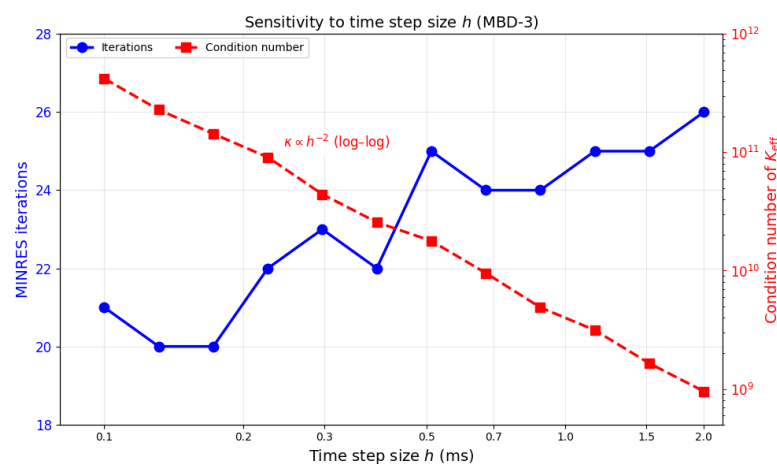


Figure 5. MINRES iteration count as a function of the time step size h for the MBD-3 benchmark. Over the range $h \in [0.1, 2]$ ms, the condition number of $\mathbf{K}_{\text{eff}}$ increases from approximately 10^9 to 10^{11} , representing a variation of two orders of magnitude. Despite this wide range of conditioning, the MINRES iteration count remains nearly constant between 20 and 26. This near-independence demonstrates that HPI-MBD delivers problem-size independent convergence in the temporal sense, consistent with the spectral bounds established in Theorem 1.

These results confirm that HPI-MBD maintains robust performance across a range of temporal discretisations. The preconditioner does not exhibit any degradation in the small-step regime where $\mathbf{K}_{\text{eff}}$ is most severely ill-conditioned. It also does not suffer from excessive iteration counts at larger step sizes. This stability is particularly important for practical applications involving adaptive time-stepping strategies, where h may vary significantly over the course of a simulation. The near-independence of the iteration

count from the time step demonstrates that HPI-MBD delivers problem-size independent convergence in the temporal sense as well.

6. Conclusions

This paper has introduced HPI-MBD, a hybrid preconditioned iterative framework for solving large-scale saddle-point systems in multibody dynamics. The framework combines an AMG V-cycle for the mechanical sub-block with a graph-structured block-Jacobi approximation of the constraint Schur complement. This design achieves effective eigenvalue clustering and near-optimal spectral conditioning while maintaining $\mathcal{O}(n)$ complexity per iteration. A rigorous spectral analysis establishes the theoretical foundation. The derived bounds guarantee problem-size independent convergence of the MINRES solver.

Extensive numerical experiments on five benchmark systems, ranging from 10^4 to 10^6 degrees of freedom, demonstrate the practical effectiveness of the approach. Against MA57, HPI-MBD achieves speedups up to $12.7\times$ and memory reductions up to 68%, while maintaining relative residuals below 10^{-6} . Against PARDISO, the speedup reaches $11.2\times$ and the memory reduction is approximately 69%. Further comparisons with ILU(0)-GMRES, FETI-1, and a block-Jacobi-only variant without AMG confirm that the combination of AMG smoothing with a graph-aware Schur complement is crucial to the observed performance gains. On the largest benchmark system, HPI-MBD outperforms these methods by factors of $8.8\times$, $2.6\times$, and $4.1\times$, respectively.

The predicted $\mathcal{O}(n)$ complexity is validated by the near-linear scaling observed in numerical experiments, with a slope of 1.13 in the log-log plot of time versus problem size. Parallel efficiency remains above 82% on up to 32 MPI ranks. The solver exhibits robust performance under redundant constraints and near-singular kinematic conditions. A systematic sensitivity study confirms that Tikhonov regularisation does not compromise long-term physical accuracy. The induced perturbation is three orders of magnitude smaller than both the discretisation error and the Newton tolerance. Across time step sizes from 0.1 ms to 2 ms, the condition number of $\mathbf{K}_{\text{eff}}$ varies by two orders of magnitude. The MINRES iteration count stays between 20 and 26 over this range. Total simulation times further confirm that the inexact search direction does not increase outer Newton iterations. HPI-MBD averages 3.2 iterations per step, compared with 2.9 for direct solvers. The cumulative speedup reaches $11.6\times$ on the largest system. These results establish HPI-MBD as a scalable and memory-efficient alternative to direct solvers for real-time and memory-constrained applications.

Several limitations should be acknowledged. The current framework is designed for symmetric indefinite systems under holonomic constraints, where MINRES is appropriate. For non-holonomic problems, where symmetry is lost, GMRES would be necessary. This would incur higher memory costs. The AMG coarsening relies on classical Ruge–Stüben criteria, which are less effective for highly anisotropic problems such as thin-shell structures. Moreover, the implementation is CPU-based and has not yet exploited GPU acceleration. Nevertheless, HPI-MBD is agnostic to time integrators and constraint stabilisation techniques. It tolerates redundant constraints via regularisation and can be readily integrated into existing MBD software frameworks.

Future work will prioritise GPU acceleration of the local dense Schur complement factorisations and sparse matrix-vector products. Other directions include extending the method to non-holonomic and contact constraints, developing anisotropic multigrid for thin-shell flexible bodies, designing adaptive coarsening strategies based on Jacobian evolution, and incorporating filtering techniques for contact-dominated problems. These

efforts are expected to further advance real-time high-fidelity multibody simulation in robotics, vehicle dynamics, and biomechanics.

Author Contributions: Conceptualisation, D.W.; methodology, D.W.; software, D.W.; investigation, D.W.; formal analysis, D.W.; writing—original draft preparation, D.W.; supervision, H.R. and P.G.; validation, H.R.; writing—review and editing, H.R., P.G. and C.S.; resources, P.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Di Wang was employed by Zhejiang Geely Holding Group Co., Ltd. and worked at the Geely Automobile Research Institute (Ningbo) Co., Ltd.; Perry Gu was employed by Zhejiang Geely Holding Group Co., Ltd.; and Chongchong Song was employed by the Geely Automobile Research Institute (Ningbo) Co., Ltd. The remaining author (Hui Ren) declared that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

- Shabana, A.A. *Dynamics of Multibody Systems*; Cambridge University Press: Cambridge, UK, 2020.
- Haug, E.J. *Computer Aided Kinematics and Dynamics Of Mechanical Systems*; Allyn and Bacon: Boston, MA, USA, 1989; Volume 1.
- Pandy, M.G. Computer modeling and simulation of human movement. *Annu. Rev. Biomed. Eng.* **2001**, *3*, 245–273. [[CrossRef](#)] [[PubMed](#)]
- Kübler, L.; Eberhard, P.; Geisler, J. Flexible multibody systems with large deformations and nonlinear structural damping using absolute nodal coordinates. *Nonlinear Dyn.* **2003**, *34*, 31–52. [[CrossRef](#)]
- Duff, I.S.; Reid, J.K. The multifrontal solution of indefinite sparse symmetric linear. *ACM Trans. Math. Softw. (TOMS)* **1983**, *9*, 302–325. [[CrossRef](#)]
- Schenk, O.; Gärtner, K. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Gener. Comput. Syst.* **2004**, *20*, 475–487. [[CrossRef](#)]
- Amestoy, P.R.; Duff, I.S.; L’Excellent, J.Y.; Koster, J. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.* **2001**, *23*, 15–41. [[CrossRef](#)]
- Hansknecht, C.; Heinzlreiter, B.; Pearson, J.W.; Potschka, A. A Framework for the Solution of Tree-Coupled Saddle-Point Systems. *Numer. Linear Algebra Appl.* **2025**, *32*, e70038. [[CrossRef](#)]
- Wu, H.Y. A general class of constraint preconditioners for generalized saddle point linear systems. *Appl. Math. Comput.* **2025**, *489*, 129148. [[CrossRef](#)]
- Klawonn, A.; Rheinbach, O. Highly scalable parallel domain decomposition methods with an application to biomechanics. *ZAMM-J. Appl. Math. Mech. Angew. Math. Und Mech. Appl. Math. Mech.* **2010**, *90*, 5–32.
- Göbel, F.; Huynh, N.M.M.; Chagini, F.; Pavarino, L.F.; Weiser, M.; Scacchi, S.; Anzt, H. A BDDC Preconditioner for the Cardiac EMI Model in three Dimensions. *SIAM J. Sci. Comput.* **2026**, *48*, A646–A667. [[CrossRef](#)]
- Lan, J.; Liu, C.; Zhang, S.; Rong, J. An enhanced BDDC preconditioner with weighted Schur complement averaging: Application to flexible multibody systems. *Int. J. Appl. Mech.* **2026**, *18*, 2650017. [[CrossRef](#)]
- Petrides, S.; Hartland, T.; Kolev, T.; Lee, C.S.; Puso, M.; Solberg, J.; Chin, E.B.; Wang, J.; Petra, C. Algebraic Multigrid with Filtering: An Efficient Preconditioner for Interior Point Methods in Large-Scale Contact Mechanics Optimization. *SIAM J. Sci. Comput.* **2026**, *48*, A828–A852. [[CrossRef](#)]
- Li, C.; Yu, P.; Liu, T.; Yu, S.; Xiao, Y.; Li, S.; Hao, A.; Gao, Y.; Zhao, Q. MGPBD: A Multigrid Accelerated Global XPBD Solver. In Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers, Los Angeles, CA, USA, 11–15 August 2025; pp. 1–11.
- Baumgarte, J. Stabilization of constraints and integrals of motion in dynamical systems. *Comput. Methods Appl. Mech. Eng.* **1972**, *1*, 1–16. [[CrossRef](#)]
- De Jalon, J.G.; Bayo, E. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*; Springer Science & Business Media: Heidelberg, Germany, 2012.
- Ruge, J.W.; Stüben, K. Algebraic multigrid. In *Multigrid Methods*; SIAM: Philadelphia, PA, USA, 1987; pp. 73–130.
- Lee, B. Algebraic multigrid for systems of elliptic boundary-value problems. *Numer. Linear Algebra Appl.* **2021**, *28*, e2303.

19. Hu, J.J.; Siefert, C.M.; Tuminaro, R.S. Smoothed aggregation for difficult stretched mesh and coefficient variation problems. *Numer. Linear Algebra Appl.* **2022**, *29*, e2442. [[CrossRef](#)]
20. Karypis, G.; Kumar, V. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *J. Parallel Distrib. Comput.* **1998**, *48*, 71–95. [[CrossRef](#)]
21. Falgout, R.D.; Jones, J.E.; Yang, U.M. The design and implementation of hypre, a library of parallel high performance preconditioners. In *Numerical Solution of Partial Differential Equations on Parallel Computers*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 267–294.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.