Research Article

An improved convergence based on accelerated modulus-based Gauss–Seidel method for interactive rigid body simulations

Shugo Miyamoto[1] · Makoto Yamashita[2]

Received: 6 July 2020 / Accepted: 18 January 2021 / Published online: 3 February 2021

Abstract
In this paper, we propose a novel method that fits linear complementarity problems arising in interactive rigid-body simulations, based on the accelerated modulus-based Gauss–Seidel (AMGS) method. We give a new sufficient condition for the convergence of the generated sequence under a milder condition on the matrix splitting than the special case of the AMGS method. This gives a flexibility in the choice of the matrix splitting, and an appropriate matrix splitting can lead to a better convergence rate in practice. Numerical experiments show that the proposed method is more efficient than the simple application of the AMGS method, and that the accuracy in each step of the proposed method is superior to that of the projected Gauss–Seidel method.

Keywords Iterative methods for linear systems · Dynamics of multibody systems · Linear complementarity problems · Interactive simulations

1 Introduction

In rigid-body simulations, interactions (for example, normal forces) between rigid bodies are often mathematically modeled as constraints. For computing these constraint forces, we usually need to solve certain equations. Two representative constraint formulations for rigid-body simulations are acceleration-based formulations [3] and velocity-based formulations [1]. In the acceleration-based formulations, the constraints are described with forces and accelerations of rigid bodies; we first compute forces and accelerations, then integrate them to obtain velocity changes. On the other hand, in the velocity-based formulations, the variables in the constraints are impulses and velocities of the rigid bodies. In this paper, we focus on the velocity-based formulations, since the velocity-based formulations are widely used and are known to be superior to the acceleration-based formulations in many aspects (for example, see [9, 14]). In recent years, position-based formulations [8] have also been developed for rigid-body simulations. The position-based method was originally proposed for cloth simulations [7], but it is widely used for various simulations especially in the context of computer graphics [6].

There are two main categories for solving constraints, iterative approaches and direct approaches. Our interest in this paper is the iterative approaches rather than the direct approaches, since the direct approaches such as pivoting methods often suffer from time complexity and numerical instability as pointed in [13]. In the impulse-based iterative approaches [9], impulses are applied to the rigid bodies sequentially, until certain convergence conditions are satisfied. Tang et al. [16] proposed an impulse-based energy tracking method that computes accurate velocities by applying impulses iteratively. This method implicitly computes relative velocities after collisions, since an

✉ Shugo Miyamoto, miyamoto-s@g.ecc.u-tokyo.ac.jp; Makoto Yamashita, Makoto.Yamashita@c.titech.ac.jp | [1]Department of Systems Innovation, School of Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan. [2]Department of Mathematical and Computing Science, Tokyo Institute of Technology, 2-12-1-W8-29 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan.

explicit computation of the velocities may lead to physically inaccurate results when multiple contacts occur simultaneously.

Contact constraints are frequently modeled in a form of complementarity problems [4]; in particular, linear complementarity problems (LCPs) give mathematical formulations for frictionless contacts. For frictional contacts, we can use nonlinear complementarity problems (NCPs) to formulate accurate Coulomb friction [18], or, as we will discuss later in Sect. 4.1, we can use boxed LCPs for approximated contact friction which employs friction pyramids instead of accurate friction cones. Among various iterative methods for solving LCPs, the projected Gauss–Seidel (PGS) method [5] has many extensions for solving contact constraints [9, 10, 14]. Another iterative approach for solving LCPs is the use of modulus-based methods. Bai [2] established modulus-based matrix splitting iteration (MMSI) methods, which include modulus-based Jacobi (MJ), modulus-based Gauss–Seidel (MGS), modulus-based successive over relaxation (MSOR), and modulus-based accelerated overrelaxation (MAOR) iteration methods as special cases. Mezzadri and Galligani [12] proposed an extension of the MMSI methods so that they can be used to solve horizontal linear complementarity problems (HLCPs). Zheng and Vong [19] examined the convergence of the MMSI methods for HLCPs, and they proposed a more general convergence result. Zheng and Yin [20] proposed accelerated modulus-based matrix splitting iteration (AMMSI) methods as an improvement of Bai [2]. In a similar way to the MMSI methods, the AMMSI methods include accelerated modulus-based Jacobi (AMJ), accelerated modulus-based SOR (AMSOR), and accelerated modulus-based accelerated overrelaxation (AMAOR) iteration methods. Furthemore, the AMMSI methods also devised the accelerated modulus-based Gauss–Seidel (AMGS) method.

In this paper, we give a theoretical proof on the convergence of the accelerated modulus-based Gauss–Seidel (AMGS) method. Zheng and Yin [20] already discussed the convergence in a general case, but their assumption for the case of positive-definite coefficient matrices is too restrictive to apply the same discussion to rigid-body simulations. We propose another sufficient condition of the convergence of the AMGS method when the coefficient matrix of the LCP is positive definite, so that we can choose a parameter which leads to a faster convergence. We also show a simple example that is covered by the condition we propose, but is not covered by the condition of a general case proposed in [20].

We also improve the efficency of the proposed method. In many applications of real-time simulations, interactive computer graphics and operations are considered most important, since, if the computation in each simulation step is considerably slower than real time, the quality of the users' experience would be seriously degraded. Since the AMGS method proposed in [20] is not designed for interactive rigid-body simulations, a simple application of the AMGS method causes inefficiency and it is a serious disadvantage for real-time simulations. The proposed method focuses the update formula in the AMGS method and exploits the structures related to the generalized velocity vector of rigid bodies.

Through numerical experiments, we observed that the proposed AMGS method attained shorter computation time than the original AMGS method. Furthermore, its convergence rate in each iteration was better than that of the PGS method. These results indicate that the proposed method is useful for practical real-time simulations. Mezzadri [11] showed that under a specific condition, the PGS method and the AMGS method are equivalent in that AMGS iterations can be written as PGS iterations. Mezzadri also pointed out that the AMGS method also performs like the projected successive over-relaxation (PSOR) method under a specific parameter choice, and this is consistent with our numerical results.

The outline of this paper is as follows. In Sect. 2, we briefly introduce a formulation of velocity-based constraints as an LCP. We also discuss the PGS method and the AMGS method to solve LCPs. We prove convergence theorems of the AMGS method in Sect. 3, and the application of the AMGS method to rigid-body simulations is developed in Sect. 4. In Sect. 5, we will show numerical results to verify the efficiency of the AMGS method. Finally, we will give a conclusion in Sect. 6.

2 Preliminaries

2.1 Linear complementarity problem with velocity-based constraints

For the latter discussions, we briefly introduce an LCP that arises from velocity-based constraints. For more details, the readers can refer to [4, 15, 17].

During a rigid-body simulation, we keep tracking movements of the rigid bodies in multiple time periods, therefore, an entire simulation is divided into a sequence of simulation steps, and each simulation step corresponds to a small time step. Since the constraints on the rigid bodies should be satisfied at each time, we solve the following LCP in each simulation step:

$$\begin{cases} \boldsymbol{\lambda} \geq \mathbf{0} \\ \boldsymbol{JM}^{-1}\boldsymbol{J}^{T}\boldsymbol{\lambda} + \boldsymbol{Jv} + \boldsymbol{b} \geq \mathbf{0} \\ (\boldsymbol{JM}^{-1}\boldsymbol{J}^{T}\boldsymbol{\lambda} + \boldsymbol{Jv} + \boldsymbol{b})^{T}\boldsymbol{\lambda} = 0 \end{cases} \tag{1}$$

In this paper, we use the superscript T to denote the transpose of a vector or a matrix.

The decision variable in this LCP is $\lambda \in \mathbb{R}^m$, which is the impulse vector applied to the rigid bodies in the constraint space. The first constraint in (1) requires λ to be nonnegative to ensure that the constraint impulse must be repulsive.

The second constraint in (1) corresponds to the velocity constraints in the rigid-body simulation. The vectors $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$ are the bias vector in a constraint space and the generalized velocity vector of rigid bodies, respectively. More precisely, when we have N rigid bodies, $\boldsymbol{v}$ is a vector that consists of N linear and angular velocities, *i.e.*,

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}_1^T & \boldsymbol{\omega}_1^T & \boldsymbol{v}_2^T & \boldsymbol{\omega}_2^T & \cdots & \boldsymbol{v}_N^T & \boldsymbol{\omega}_N^T \end{pmatrix}^T$$

where $\boldsymbol{v}_i \in \mathbb{R}^3$ and $\boldsymbol{\omega}_i \in \mathbb{R}^3$ are the linear velocity and the angular velocity of the ith rigid body, respectively, for $i = 1, \ldots, N$; thus the length of $\boldsymbol{v}$ is $n = 6N$. The matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the Jacobian matrix corresponding to the velocity constraints. The generalized mass matrix of the rigid bodies $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ consists of masses and inertia tensor matrices in the diagonal positions:

$$\boldsymbol{M} = \begin{pmatrix} m_1\boldsymbol{E}_3 & & & & & & \\ & \boldsymbol{I}_1 & & & & & \\ & & m_2\boldsymbol{E}_3 & & & & \\ & & & \boldsymbol{I}_2 & & & \\ & & & & \ddots & & \\ & & & & & m_N\boldsymbol{E}_3 & \\ & & & & & & \boldsymbol{I}_N \end{pmatrix}, \tag{2}$$

where $m_i \in \mathbb{R}$ and $\boldsymbol{I}_i \in \mathbb{R}^{3\times 3}$ are the mass and the inertia tensor matrix of the ith rigid body, respectively. We also use $\boldsymbol{E}_r \in \mathbb{R}^{r \times r}$ to denote the identity matrix of order r. The inertia tensor matricies $\boldsymbol{I}_1, \ldots, \boldsymbol{I}_N$ are symmetric, so is $\boldsymbol{M}$.

The third constraint in (1) is a complementarity condition. We can understand this complementarity condition as follows. If $(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\lambda + \boldsymbol{J}\boldsymbol{v} + \boldsymbol{b})_i > 0$ holds for some i, then the rigid bodies are moving away from each other in the direction of the ith constraint, therefore, the ith constraint should be "inactive". However, λ is the impulse vector, thus $\lambda_i > 0$ implies that the ith constraint must be "active". Hence, $\lambda_i > 0$ and $(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\lambda + \boldsymbol{J}\boldsymbol{v} + \boldsymbol{b})_i > 0$ should not hold simultaneously, and this requirement is implemented in the complementarity condition.

By denoting $\boldsymbol{q} = \boldsymbol{J}\boldsymbol{v} + \boldsymbol{b}$ and $\boldsymbol{A} = \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T$ and introducing an auxiliary variable $\boldsymbol{w} \in \mathbb{R}^m$, the LCP (1) can be expressed in a general LCP as follows:

$$\mathrm{LCP}(\boldsymbol{q}, \boldsymbol{A}) \quad \begin{cases} \boldsymbol{A}\lambda + \boldsymbol{q} = \boldsymbol{w} \\ \boldsymbol{w}^T\lambda = 0 \\ \lambda, \boldsymbol{w} \geq \boldsymbol{0} \end{cases} \tag{3}$$

It is known that if the constraints are non-degenerate, the Jacobian matrix $\boldsymbol{J}$ is full row rank and the matrix $\boldsymbol{A} = \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T$ is positive definite (see [5], for example). Throughout this paper, we assume that $\boldsymbol{A}$ is positive definite. Since any positive definite matrix is a P-matrix, $\boldsymbol{A}$ is also a P-matrix and therefore the LCP (3) has a unique solution for an arbitrary vector $\boldsymbol{q}$.

At the end of this subsection, we should note that the input data $\boldsymbol{b}, \boldsymbol{v}, \boldsymbol{J}$ and $\boldsymbol{M}$ vary in accordance with simulation steps. If we express the time dependence explicitly, they should be $\boldsymbol{b}^{(t)}, \boldsymbol{v}^{(t)}, \boldsymbol{J}^{(t)}$ and $\boldsymbol{M}^{(t)}$ where t is the simulation step. However, in this paper, we mainly focus on solving (1) in each simulation step, therefore, we usually drop the simulation step (t) from $\boldsymbol{b}^{(t)}, \boldsymbol{v}^{(t)}, \boldsymbol{J}^{(t)}$ and $\boldsymbol{M}^{(t)}$.

2.2 Projected Gauss–Seidel method

In the LCP (3) from the rigid-body simulation, the matrix $\boldsymbol{A}$ has a structure such that $\boldsymbol{A} = \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T$. The projected Gauss–Seidel (PGS) method [9] is designed to solve more general LCPs (3) in the sense that the assumption for $\boldsymbol{A}$ is only positive definiteness. The PGS method is an iterative method and generates a sequence $\{\lambda^k\}_{k=0}^{\infty} \subset \mathbb{R}^m$.

A key property in the PGS method is to decompose $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$ such that $\boldsymbol{D}$ is a diagonal matrix, $\boldsymbol{L}$ a strictly lower triangular matrix, and $\boldsymbol{U}$ a strictly upper triangular matrix. Since we assume $\boldsymbol{A}$ is a positive definite matrix, $\boldsymbol{D}$ is invertible. Due to this decomposition, $\boldsymbol{A}\lambda + \boldsymbol{q} = \boldsymbol{0}$ is equivalent to $\lambda = \boldsymbol{D}^{-1}(\boldsymbol{L}\lambda + \boldsymbol{U}\lambda - \boldsymbol{q})$.

Taking this formula and the complementarity condition into consideration, the PGS method computes the next iteration λ^{k+1} by the following update formula:

$$\lambda^{k+1} = \max\left\{\boldsymbol{0}, \boldsymbol{D}^{-1}\left(\boldsymbol{L}\lambda^{k+1} + \boldsymbol{U}\lambda^k - \boldsymbol{q}\right)\right\}. \tag{4}$$

Throughout this paper, we use $\max\{\boldsymbol{a}, \boldsymbol{b}\}$ ($\min\{\boldsymbol{a}, \boldsymbol{b}\}$) to denote the element-wise maximum (minimum, respectively) of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The PGS method continues the update by (4) until the sequence $\{\lambda^k\}_{k=0}^{\infty}$ converges enough, or the number of the iterations reaches a certain limit.

In the rigid-body simulation, the initial vector λ^0 is usually set as a zero vector $\boldsymbol{0}$ or the impulse vector obtained in the previous simulation step. Since the initial vector often affects the performance of iterative approaches, the use of the solution from the previous simulation step makes the convergence faster [9]. Such a technique is called warm start.

2.3 Accelerated modulus-based Gauss–Seidel method

For solving the general LCP (3), Bai [2] devised the following implicit fixed-point equation that is essential

for the modulus-based matrix splitting iteration (MMSI) methods:

$$(\boldsymbol{M}_0\boldsymbol{\Gamma}+\boldsymbol{\Omega}_1)\boldsymbol{x}=(\boldsymbol{N}_0\boldsymbol{\Gamma}-\boldsymbol{\Omega}_2)\boldsymbol{x}+(\boldsymbol{\Omega}-\boldsymbol{A}\boldsymbol{\Gamma})|\boldsymbol{x}|-\boldsymbol{q} \tag{5}$$

Here, the matrices $\boldsymbol{M}_0 \in \mathbb{R}^{m\times m}$ and $\boldsymbol{N}_0 \in \mathbb{R}^{m\times m}$ are a splitting pair of $\boldsymbol{A}$ such that $\boldsymbol{A}=\boldsymbol{M}_0-\boldsymbol{N}_0$. $\boldsymbol{\Omega}\in\mathbb{R}^{m\times m}$ and $\boldsymbol{\Gamma}\in\mathbb{R}^{m\times m}$ are two diagonal matrices whose diagonal entries are positive. $\boldsymbol{\Omega}_1\in\mathbb{R}^{m\times m}$ and $\boldsymbol{\Omega}_2\in\mathbb{R}^{m\times m}$ are nonnegative diagonal matrices such that $\boldsymbol{\Omega}=\boldsymbol{\Omega}_1+\boldsymbol{\Omega}_2$. We should emphasize that the variable in (5) is $\boldsymbol{x}$, and we use $|\boldsymbol{x}|$ to denote the element-wise absolute values of $\boldsymbol{x}$. The relation between $\boldsymbol{x}$ and the pair of λ and $\boldsymbol{w}$ in (3) will be discussed in Theorem 1.

By setting $\boldsymbol{\Omega}_1=\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_2=\boldsymbol{O}$, and $\boldsymbol{\Gamma}=\frac{1}{\gamma}\boldsymbol{E}_m$ with a parameter $\gamma>0$ in (5), we can obtain a simplified implicit fixed-point equation:

$$(\boldsymbol{M}_0+\gamma\boldsymbol{\Omega})\boldsymbol{x}=\boldsymbol{N}_0\boldsymbol{x}+(\gamma\boldsymbol{\Omega}-\boldsymbol{A})|\boldsymbol{x}|-\gamma\boldsymbol{q}. \tag{6}$$

Based on (6), the iteration of the MMSI method can be derived as follows:

$$(\boldsymbol{M}_0+\gamma\boldsymbol{\Omega})\boldsymbol{x}^{k+1}=\boldsymbol{N}_0\boldsymbol{x}^k+(\gamma\boldsymbol{\Omega}-\boldsymbol{A})\left|\boldsymbol{x}^k\right|-\gamma\boldsymbol{q}. \tag{7}$$

We decompose $\boldsymbol{A}$ into $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$ in the same way as the PGS method. Set $\gamma=2$, and let $\alpha>0$ and $\beta>0$ be two parameters. Then, we can derive the update formula of four methods from (7); the modulus-based Jacobi (MJ) by setting $\boldsymbol{M}_0=\boldsymbol{D}$ in (7), the modulus-based Gauss–Seidel (MGS) by $\boldsymbol{M}_0=\boldsymbol{D}-\boldsymbol{L}$, the modulus-based successive over relaxation (MSOR) by $\boldsymbol{M}_0=\frac{1}{\alpha}\boldsymbol{D}-\boldsymbol{L}$, and the modulus-based accelerated overrelaxation (MAOR) iteration method by $\boldsymbol{M}_0=\frac{1}{\alpha}(\boldsymbol{D}-\beta\boldsymbol{L})$, respectively.

Zheng and Yin [20] utilized two splitting pairs of the matrix $\boldsymbol{A}$ such that $\boldsymbol{A}=\boldsymbol{M}_1-\boldsymbol{N}_1=\boldsymbol{M}_2-\boldsymbol{N}_2$, and devised a new equation based on (5):

$$\begin{aligned}(\boldsymbol{M}_1\boldsymbol{\Gamma}+\boldsymbol{\Omega}_1)\boldsymbol{x}&=(\boldsymbol{N}_1\boldsymbol{\Gamma}-\boldsymbol{\Omega}_2)\boldsymbol{x}+(\boldsymbol{\Omega}-\boldsymbol{M}_2\boldsymbol{\Gamma})|\boldsymbol{x}|\\&\quad+\boldsymbol{N}_2\boldsymbol{\Gamma}|\boldsymbol{x}|-\boldsymbol{q}.\end{aligned} \tag{8}$$

Zheng and Yin [20] established the following theorem to show an equivalence between (8) and LCP($\boldsymbol{q},\boldsymbol{A}$) in (3). Since a detailed proof is not given in [20], we give the proof here.

Theorem 1 [20] *The following statements hold between* (8) *and* LCP($\boldsymbol{q},\boldsymbol{A}$):

(i) *if* $(\lambda,\boldsymbol{w})$ *is a solution of* LCP($\boldsymbol{q},\boldsymbol{A}$), *then* $\boldsymbol{x}=\frac{1}{2}(\boldsymbol{\Gamma}^{-1}\lambda-\boldsymbol{\Omega}^{-1}\boldsymbol{w})$ *satisfies* (8).
(ii) *if* $\boldsymbol{x}$ *satisfies* (8), *then the pair of* $\lambda=\boldsymbol{\Gamma}(|\boldsymbol{x}|+\boldsymbol{x})$ *and* $\boldsymbol{w}=\boldsymbol{\Omega}(|\boldsymbol{x}|-\boldsymbol{x})$ *is a solution of* LCP($\boldsymbol{q},\boldsymbol{A}$).

Proof We first prove (i). Since $(\lambda,\boldsymbol{w})$ is a solution of LCP($\boldsymbol{q},\boldsymbol{A}$), $(\lambda,\boldsymbol{w})$ satisfies the four constraints, $\boldsymbol{A}\lambda+\boldsymbol{q}=\boldsymbol{w}$, $\boldsymbol{w}^T\lambda=0$, $\lambda\geq\boldsymbol{0}$ and $\boldsymbol{w}\geq\boldsymbol{0}$. The first constraint $\boldsymbol{A}\lambda+\boldsymbol{q}=\boldsymbol{w}$ is equivalent to

$$\begin{aligned}&(\boldsymbol{\Omega}+\boldsymbol{A}\boldsymbol{\Gamma})(\boldsymbol{\Gamma}^{-1}\lambda-\boldsymbol{\Omega}^{-1}\boldsymbol{w})\\&\quad=(\boldsymbol{\Omega}-\boldsymbol{A}\boldsymbol{\Gamma})(\boldsymbol{\Gamma}^{-1}\lambda+\boldsymbol{\Omega}^{-1}\boldsymbol{w})-2\boldsymbol{q}.\end{aligned}$$

From the rest three constraints and the fact that $\boldsymbol{\Gamma}$ and $\boldsymbol{\Omega}$ are diagonal matrices whose diagonal entries are positive, if $\boldsymbol{x}=\frac{1}{2}(\boldsymbol{\Gamma}^{-1}\lambda-\boldsymbol{\Omega}^{-1}\boldsymbol{w})$, it holds that $|\boldsymbol{x}|=\frac{1}{2}(\boldsymbol{\Gamma}^{-1}\lambda+\boldsymbol{\Omega}^{-1}\boldsymbol{w})$. Therefore, $\boldsymbol{x}$ satisfies

$$(\boldsymbol{\Omega}+\boldsymbol{A}\boldsymbol{\Gamma})\boldsymbol{x}=(\boldsymbol{\Omega}-\boldsymbol{A}\boldsymbol{\Gamma})|\boldsymbol{x}|-\boldsymbol{q} \tag{9}$$

and this is equivalent to (8).

To prove (ii), from (9), it holds that $\boldsymbol{A}\boldsymbol{\Gamma}(|\boldsymbol{x}|+\boldsymbol{x})+\boldsymbol{q}=\boldsymbol{\Omega}(|\boldsymbol{x}|-\boldsymbol{x})$. By the relations $\lambda=\boldsymbol{\Gamma}(|\boldsymbol{x}|+\boldsymbol{x})$ and $\boldsymbol{w}=\boldsymbol{\Omega}(|\boldsymbol{x}|-\boldsymbol{x})$, we obtain $\boldsymbol{A}\lambda+\boldsymbol{q}=\boldsymbol{w}$. Since $\boldsymbol{\Gamma}$ and $\boldsymbol{\Omega}$ are positive diagonal matrices, it is easy to check that λ and $\boldsymbol{w}$ are nonnegative vectors. Finally, it is also easy to show the element-wise complementarity between λ and $\boldsymbol{w}$. □

We may use Theorem 1 to establish some iterative methods for solving LCP($\boldsymbol{q},\boldsymbol{A}$), but we need to set appropriate matrices for the implicit fixed-point equation (8) in actual computations. In particular, the splitting pair of $\boldsymbol{\Omega}$ is not unique. By fixing $\boldsymbol{\Omega}_1=\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_2=\boldsymbol{O}$ and $\boldsymbol{\Gamma}=\frac{1}{\gamma}\boldsymbol{E}_m$, we derive a simplified update equation of (8) as follows:

$$(\boldsymbol{M}_1+\gamma\boldsymbol{\Omega})\boldsymbol{x}=\boldsymbol{N}_1\boldsymbol{x}+(\gamma\boldsymbol{\Omega}-\boldsymbol{M}_2)|\boldsymbol{x}|+\boldsymbol{N}_2|\boldsymbol{x}|-\gamma\boldsymbol{q}. \tag{10}$$

As mentioned in Sect. 2.1, we use $\boldsymbol{E}_r\in\mathbb{R}^{r\times r}$ to denote the identity matrix of order r. Based on this equation, Zheng and Yin [20] provided an update formula of the AMMSI methods:

$$\begin{aligned}&(\boldsymbol{M}_1+\gamma\boldsymbol{\Omega})\boldsymbol{x}^{k+1}\\&\quad=\boldsymbol{N}_1\boldsymbol{x}^k+(\gamma\boldsymbol{\Omega}-\boldsymbol{M}_2)\left|\boldsymbol{x}^k\right|+\boldsymbol{N}_2\left|\boldsymbol{x}^{k+1}\right|-\gamma\boldsymbol{q}\end{aligned} \tag{11}$$

When the sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ converges enough, the AMMSI methods output the impulse vector by using the relation $\lambda=\boldsymbol{\Gamma}(|\boldsymbol{x}|+\boldsymbol{x})=\frac{|\boldsymbol{x}|+\boldsymbol{x}}{\gamma}$. As Mezzadri [11] points out, every AMMSI method can be written in a projection form, and the value of γ does not play an important role in convergence rates as it just changes the first iteration of the method in its projection form.

By changing the splitting pairs of $\boldsymbol{A}$, the update formula (11) above yields variant methods; MMSIM ($\boldsymbol{M}_2=\boldsymbol{A}$ and $\boldsymbol{N}_2=\boldsymbol{O}$), the accelerated modulus-based Jacobi (AMJ) iteration method ($\boldsymbol{M}_1=\boldsymbol{D}$, $\boldsymbol{N}_1=\boldsymbol{L}+\boldsymbol{U}$, $\boldsymbol{M}_2=\boldsymbol{D}-\boldsymbol{U}$ and $\boldsymbol{N}_2=\boldsymbol{L}$), the accelerated modulus-based SOR (AMSOR) iteration method ($\boldsymbol{M}_1=\frac{1}{\alpha}\boldsymbol{D}-\boldsymbol{L}$, $\boldsymbol{N}_1=\left(\frac{1}{\alpha}-1\right)\boldsymbol{D}+\boldsymbol{U}$,

$\boldsymbol{M}_2 = \boldsymbol{D} - \boldsymbol{U}$ and $\boldsymbol{N}_2 = \boldsymbol{L}$), and the accelerated modulus-based accelerated overrelaxation (AMAOR) iteration method ($\boldsymbol{M}_1 = \frac{1}{\alpha}(\boldsymbol{D} - \beta\boldsymbol{L})$, $\boldsymbol{N}_1 = \frac{1}{\alpha}((\alpha-1)\boldsymbol{D} + (\alpha-\beta)\boldsymbol{L} + \alpha\boldsymbol{U})$, $\boldsymbol{M}_2 = \boldsymbol{D} - \boldsymbol{U}$ and $\boldsymbol{N}_2 = \boldsymbol{L}$).

In particular, the update formula of the accelerated modulus-based Gauss–Seidel (AMGS) method in [20] is derived with $\boldsymbol{M}_1 = \boldsymbol{D} - \boldsymbol{L}, \boldsymbol{N}_1 = \boldsymbol{U}, \boldsymbol{M}_2 = \boldsymbol{D} - \boldsymbol{U}$ and $\boldsymbol{N}_2 = \boldsymbol{L}$ as follows:

$$\begin{aligned}&(\boldsymbol{D} + \gamma\boldsymbol{\Omega} - \boldsymbol{L})\boldsymbol{x}^{k+1}\\ &\quad = \boldsymbol{U}\boldsymbol{x}^k + (\gamma\boldsymbol{\Omega} - \boldsymbol{D} + \boldsymbol{U})\left|\boldsymbol{x}^k\right| + \boldsymbol{L}\left|\boldsymbol{x}^{k+1}\right| - \gamma\boldsymbol{q}.\end{aligned} \tag{12}$$

Let $\Delta\boldsymbol{x}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ be the difference between $\boldsymbol{x}^k$ and $\boldsymbol{x}^{k+1}$. Then, (12) is equivalent to

$$\begin{aligned}&(\boldsymbol{D} + \gamma\boldsymbol{\Omega})\Delta\boldsymbol{x}^k\\ &\quad = \boldsymbol{L}\boldsymbol{x}^{k+1} - (\gamma\boldsymbol{\Omega} + \boldsymbol{D} - \boldsymbol{U})\boldsymbol{x}^k + (\gamma\boldsymbol{\Omega} - \boldsymbol{D} + \boldsymbol{U})\left|\boldsymbol{x}^k\right|\end{aligned} \tag{13}$$

$$+ \boldsymbol{L}\left|\boldsymbol{x}^{k+1}\right| - \gamma\boldsymbol{q}. \tag{14}$$

By Theorem 1 and $\boldsymbol{\Gamma} = \frac{1}{\gamma}\boldsymbol{E}_m$, the sequence $\{\lambda^k\}_{k=0}^{\infty}$ for the LCP (3) can be associated with the sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ generated by (12) by the relation $\lambda^k = \frac{|\boldsymbol{x}^k| + \boldsymbol{x}^k}{\gamma} = \frac{2}{\gamma}\max\{\boldsymbol{0}, \boldsymbol{x}^k\}$, thus λ^k is a multiple of the positive part of $\boldsymbol{x}^k$. This motivates us to split $\boldsymbol{x}^k$ into the positive and negative parts such that $\boldsymbol{x}^k = \boldsymbol{x}_+^k - \boldsymbol{x}_-^k$, where $\boldsymbol{x}_+^k = \max\{\boldsymbol{0}, \boldsymbol{x}^k\} = \frac{1}{2}(|\boldsymbol{x}^k| + \boldsymbol{x}^k)$ and $\boldsymbol{x}_-^k = -\min\{\boldsymbol{0}, \boldsymbol{x}^k\} = \frac{1}{2}(|\boldsymbol{x}^k| - \boldsymbol{x}^k)$. From the relations $\boldsymbol{x}^k = \frac{\gamma}{2}\lambda^k - \boldsymbol{x}_-^k$ and $|\boldsymbol{x}^k| = \frac{\gamma}{2}\lambda^k + \boldsymbol{x}_-^k$, (14) is equivalent to

$$\begin{aligned}(\boldsymbol{D} + \gamma\boldsymbol{\Omega})\Delta\boldsymbol{x}^k &= \gamma\boldsymbol{L}\lambda^{k+1} - (\gamma\boldsymbol{\Omega} + \boldsymbol{D} - \boldsymbol{U})(\frac{\gamma}{2}\lambda^k - \boldsymbol{x}_-^k)\\ &\quad + (\gamma\boldsymbol{\Omega} - \boldsymbol{D} + \boldsymbol{U})(\frac{\gamma}{2}\lambda^k + \boldsymbol{x}_-^k) - \gamma\boldsymbol{q}\\ &= \gamma\boldsymbol{L}\lambda^{k+1} - \gamma(\boldsymbol{D} - \boldsymbol{U})\lambda^k + 2\gamma\boldsymbol{\Omega}\boldsymbol{x}_-^k - \gamma\boldsymbol{q}.\end{aligned}$$

Therefore, for computing Δx_i^k, we only need the ith component of $\boldsymbol{x}_-^k$, which will be denoted as $(\boldsymbol{x}_-^k)_i$, since $\boldsymbol{D}$ and $\boldsymbol{\Omega}$ are diagonal matrices. This simplifies the computation of $\Delta\boldsymbol{x}^k$. Recalling the decomposition of $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$, we compute Δx_i^k for each $i = 1, \ldots, m$ by

$$\begin{aligned}\Delta x_i^k &= \frac{\gamma}{D_{ii} + \gamma\Omega_{ii}}\left(\sum_{j=1}^{i-1} L_{ij}\lambda_j^{k+1} - D_{ii}\lambda_i^k\right.\\ &\quad \left. + \sum_{j=i+1}^{m} U_{ij}\lambda_j^k + 2\Omega_{ii}(\boldsymbol{x}_-^k)_i - q_i\right)\\ &= -\frac{\gamma}{A_{ii} + \gamma\Omega_{ii}}\left(\sum_{j=1}^{i-1} A_{ij}\lambda_j^{k+1}\right.\\ &\quad \left. + \sum_{j=i}^{m} A_{ij}\lambda_j^k + q_i - 2\Omega_{ii}(\boldsymbol{x}_-^k)_i\right).\end{aligned} \tag{15}$$

We can summarize a framework of the AMGS method as follows.

Method 1 (the AMGS method for (3))

Choose a nonnegative vector $\lambda^0 \in \mathbb{R}^m$ as an initial vector. Generate the iteration sequence $\{\lambda^k\}_{k=0}^{\infty}$ by the following procedure:

$\boldsymbol{x}^0 \leftarrow \frac{\gamma}{2}\boldsymbol{\lambda}^0$
repeat $k = 1, 2, \ldots$
  for $i = 1, 2, \ldots, m$ ***do***
    $x_i^{k+1} \leftarrow x_i^k - \frac{\gamma}{A_{ii} + \gamma\Omega_{ii}}\left(\sum_{j=1}^{i-1} A_{ij}\lambda_j^{k+1} + \sum_{j=i}^{m} A_{ij}\lambda_j^k + q_i - 2\Omega_{ii}(\boldsymbol{x}_-^k)_i\right)$
  $\boldsymbol{\lambda}^k = \frac{2}{\gamma}\boldsymbol{x}_+^k$.
until $\boldsymbol{x}^k$ *satisfies a certain convergence threshold*

3 Convergence theorem

In this section, we focus on the convergence of the accelerated modulus-based Gauss–Seidel (AMGS) method.

Although Zheng and and Yin [20] gave a generic form of a sufficient condition of the convergence, a more detailed discussion of the case that $\boldsymbol{A}$ is a positive definite matrix is somewhat limited: $\boldsymbol{\Omega}$ is assumed to be a multiple of the identity matrix ($\boldsymbol{\Omega} = \bar{\omega}\boldsymbol{E}_m$ for some $\bar{\omega} > 0$), so we cannot take $\boldsymbol{\Omega} = \alpha\boldsymbol{D}$ that is actually used in the numerical experiments in [20].

Since the matrix $\boldsymbol{A}$ is always positive definite in the rigid-body simulation, it is worthwhile to give a sufficient condition of the convergence when $\boldsymbol{A}$ is positive definite while allowing a more generic form of $\boldsymbol{\Omega}$. For example, if we can take $\boldsymbol{\Omega} = \alpha\boldsymbol{D}$, we may be able to improve the convergence, as we will show in Sect. 5. Here, $\alpha \in \mathbb{R}$ is a positive constant, and $\boldsymbol{D}$ is the diagonal matrix whose diagonal elements are those of $\boldsymbol{A}$. We also give an example in Example 1 that is covered by the theorem we will show, but is not covered by theorems in [20].

We use $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$ to denote the Euclidean norm of $\boldsymbol{x} \in \mathbb{R}^m$. We also use $\|\boldsymbol{L}\|$ to denote an arbitrary norm of a matrix $\boldsymbol{L} \in \mathbb{R}^{m\times m}$ that is consistent with the Euclidean vector norm, so that we can employ $\|\boldsymbol{L}\boldsymbol{x}\| \le \|\boldsymbol{L}\|\|\boldsymbol{x}\|$. Especially, $\|\boldsymbol{L}\|_2$ represents the spectral norm of a matrix L.

The following theorem covers the case that $\boldsymbol{\Omega} = \alpha\boldsymbol{D}$, which will be actually used in numerical experiments of Sect. 5.

For the subsequent discussion, we assume that $\boldsymbol{A}$ is a symmetric positive definite matrix and hence $\boldsymbol{U} = \boldsymbol{L}^T$. Let $\bar{\boldsymbol{\Omega}} = \gamma\boldsymbol{\Omega}$. The matrix $\bar{\boldsymbol{\Omega}}$ is also a positive diagonal matrix.

Theorem 2 *Let $\boldsymbol{A}$ be a symmetric positive definite matrix, $\bar{\boldsymbol{\Omega}}$ be a positive diagonal matrix, and $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{L}^T$ be a splitting of the matrix $\boldsymbol{A}$ such that $\boldsymbol{D}$ is a positive diagonal matrix and $\boldsymbol{L}$ is a strictly lower triangular matrix. Also, let $\|\cdot\|$ be an arbitrary submultiplicative matrix norm consistent with the Euclidean vector norm. Then, if the inequality*

$$2\|\boldsymbol{L}\| < \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|^{-1} - \|\boldsymbol{D} - \boldsymbol{L} - \bar{\boldsymbol{\Omega}}\|$$

holds, the iteration sequence $\{\lambda^k\}_{k=0}^{\infty} \subset \mathbb{R}^m$ generated by Method 1 with an arbitrary nonnegative initial vector $\lambda^0 \in \mathbb{R}^m$ converges to the unique solution $\lambda^* \in \mathbb{R}^m$ of LCP($\boldsymbol{q}, \boldsymbol{A}$).

Proof In the AMMSI methods [20] from which the AMGS was derived, we chose $\boldsymbol{\Omega}_1 = \boldsymbol{\Omega}, \boldsymbol{\Omega}_2 = \boldsymbol{O}$ and $\boldsymbol{\Gamma} = \frac{1}{\gamma}\boldsymbol{E}_m$ for the simplified fixed-point equation (10). Let $(\lambda^*, \boldsymbol{w}^*) \in \mathbb{R}^m \times \mathbb{R}^m$ be a solution of LCP($\boldsymbol{q}, \boldsymbol{A}$), and let $\boldsymbol{x}^* = \frac{1}{2}(\gamma\lambda^* - \boldsymbol{\Omega}^{-1}\boldsymbol{w}^*)$. From Theorem 1, $\boldsymbol{x}^*$ satisfies (10), and the convergence of $\{\lambda^k\}_{k=0}^{\infty}$ to λ^* can be guaranteed by that of $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ to $\boldsymbol{x}^*$, therefore, we discuss the convergence of the sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$. Subtracting (10) with $\boldsymbol{x}^*$ from the update formula (11), we obtain

$$\begin{aligned}(\boldsymbol{M}_1 + \bar{\boldsymbol{\Omega}})(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) &= \boldsymbol{N}_1(\boldsymbol{x}^k - \boldsymbol{x}^*) + (\bar{\boldsymbol{\Omega}} - \boldsymbol{M}_2)(\left|\boldsymbol{x}^k\right| - |\boldsymbol{x}^*|) \\ &\quad + \boldsymbol{N}_2(\left|\boldsymbol{x}^{k+1}\right| - |\boldsymbol{x}^*|).\end{aligned}$$

In the setting of the AMGS method ($\boldsymbol{M}_1 = \boldsymbol{D} - \boldsymbol{L}, \boldsymbol{N}_1 = \boldsymbol{L}^T, \boldsymbol{M}_2 = \boldsymbol{D} - \boldsymbol{L}^T$ and $\boldsymbol{N}_2 = \boldsymbol{L}$), we can further evaluate this inequality as follows:

$$\begin{aligned}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) &= (\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}^T(\boldsymbol{x}^k - \boldsymbol{x}^*) \\ &\quad + (\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}(\bar{\boldsymbol{\Omega}} - \boldsymbol{D} + \boldsymbol{L}^T)(\left|\boldsymbol{x}^k\right| - |\boldsymbol{x}^*|) \\ &\quad + (\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}(\left|\boldsymbol{x}^{k+1}\right| - |\boldsymbol{x}^*|).\end{aligned}$$

Using the triangular inequality $\||\boldsymbol{a}| - |\boldsymbol{b}|\| \le \|\boldsymbol{a} - \boldsymbol{b}\|$ for any two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same dimension, we have

$$\begin{aligned}&\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\right\| \\ &\le \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}^T\right\|\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\| \\ &\quad + \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}(\bar{\boldsymbol{\Omega}} - \boldsymbol{D} + \boldsymbol{L}^T)\right\|\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\| \\ &\quad + \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}\right\|\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\right\| \\ &\le \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\left\|\boldsymbol{L}^T\right\|\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\| \\ &\quad + \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\left\|\bar{\boldsymbol{\Omega}} - \boldsymbol{D} + \boldsymbol{L}^T\right\|\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\| \\ &\quad + \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\|\boldsymbol{L}\|\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\right\| \\ &\le \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\left(\|\boldsymbol{L}\| + \|\boldsymbol{D} - \boldsymbol{L} - \bar{\boldsymbol{\Omega}}\|\right)\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\| \\ &\quad + \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\|\boldsymbol{L}\|\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\right\|,\end{aligned} \tag{16}$$

and it holds that

$$\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\right\| \le \frac{\left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\left(\|\boldsymbol{L}\| + \|\boldsymbol{D} - \boldsymbol{L} - \bar{\boldsymbol{\Omega}}\|\right)}{1 - \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\|\boldsymbol{L}\|}\left\|\boldsymbol{x}^k - \boldsymbol{x}^*\right\|$$

under the assumption of $\left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\|\boldsymbol{L}\| < 1$. Therefore, if

$$\frac{\left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\left(\|\boldsymbol{L}\| + \|\boldsymbol{D} - \boldsymbol{L} - \bar{\boldsymbol{\Omega}}\|\right)}{1 - \left\|(\boldsymbol{D} - \boldsymbol{L} + \bar{\boldsymbol{\Omega}})^{-1}\right\|\|\boldsymbol{L}\|} < 1,$$

or equivalently

$$2\|\boldsymbol{L}\| < \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\right\|^{-1} - \|\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}}\| \tag{17}$$

holds, we obtain the convergence of $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ by the fixed-point theorem and we know that $\boldsymbol{x}^*$ is the unique point to which the sequence $\{\boldsymbol{x}^k\}_{k=0}^{\infty}$ converges. Note that (17) implies $\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\|\|\boldsymbol{L}\| < 1$. □

Based on Theorem 2, we obtain the following corollary.

Corollary 1 *Let $\boldsymbol{A}$ be a symmetric positive definite matrix, $\bar{\boldsymbol{\Omega}}$ be a positive diagonal matrix, and $\boldsymbol{A} = \boldsymbol{D}-\boldsymbol{L}-\boldsymbol{L}^T$ be a splitting of the matrix $\boldsymbol{A}$ such that $\boldsymbol{D}$ is a positive diagonal matrix and $\boldsymbol{L}$ is a strictly lower triangular matrix. Also, let $\|\cdot\|_2$ be the spectral matrix norm, $\sigma_{\min}$ be the minimum singular value of the matrix $\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}}$, and $\sigma_{\max}$ be the maximum singular value of the matrix $\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}}$. Then, if the inequality*

$$2\|\boldsymbol{L}\|_2 < \sigma_{\min} - \sigma_{\max}$$

holds, the iteration sequence $\{\lambda^k\}_{k=0}^{\infty} \subset \mathbb{R}^m$ generated by Method 1 with an arbitrary nonnegative initial vector $\lambda^0 \in \mathbb{R}^m$ converges to the unique solution $\lambda^* \in \mathbb{R}^m$ of LCP($\boldsymbol{q}, \boldsymbol{A}$).

Proof By applying the spectral norm to (17), we get

$$2\|\boldsymbol{L}\|_2 < \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\right\|_2^{-1} - \|\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}}\|_2.$$

From the fact that

$$\left\|\boldsymbol{P}^{-1}\right\|_2 = \max_{\sigma_i}\sigma_i^{-1} = \left(\min_{\sigma_i}\sigma_i\right)^{-1}$$

holds for any invertible matrix $\boldsymbol{P}$ and its singular values σ_i, we have

$$\left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\right\|_2^{-1} = \sigma_{\min}$$
$$\|\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}}\|_2 = \sigma_{\max}.$$

Thus,

$$2\|\boldsymbol{L}\|_2 < \sigma_{\min} - \sigma_{\max} = \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\right\|_2^{-1} - \|\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}}\|_2$$

follows from the assumption and this completes the proof. □

Example 1 The following example gives a case that is applicable to Corollary 1, but is not applicable to Theorem 4.1 in [20]:

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},\ \boldsymbol{D} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix},\ \boldsymbol{L} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},\ \bar{\boldsymbol{\Omega}} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix},\ \gamma = 1$$

Indeed, $\|\boldsymbol{L}\|_2 = 1, \sigma_{\min} = \frac{1}{2}\left(\sqrt{65}-1\right) = 3.5311\ldots, \sigma_{\max} = 1$ and $2\|\boldsymbol{L}\|_2 < \sigma_{\min} - \sigma_{\max}$ holds. However, it holds that

$$\mu(\bar{\boldsymbol{\Omega}}) = \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}(\boldsymbol{D}-\boldsymbol{L}-\bar{\boldsymbol{\Omega}})\right\|_2 = 0.25$$
$$\xi(\bar{\boldsymbol{\Omega}}) = \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}^T\right\|_2 = \frac{\sqrt{17}}{16} = 0.2576\ldots$$
$$\eta(\bar{\boldsymbol{\Omega}}) = \left\|(\boldsymbol{D}-\boldsymbol{L}+\bar{\boldsymbol{\Omega}})^{-1}\boldsymbol{L}\right\|_2 = 0.25,$$

so $\mu(\bar{\boldsymbol{\Omega}}) + 2\xi(\bar{\boldsymbol{\Omega}}) + 2\eta(\bar{\boldsymbol{\Omega}}) = 1.2653\ldots > 1$ and therefore the example does not fulfill the assumption of Theorem 1.4 in [20].

4 Accelerated modulus-based matrix splitting iteration methods for interactive rigid-body simulation

Since the AMGS method is a general method for LCPs, it is possible to simply apply the AMGS method to (3). However, explicit evaluation of $\boldsymbol{A} = \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T$ is inefficient even though the matrices $\boldsymbol{J}$ and $\boldsymbol{M}^{-1}$ are sparse. Thus, such a simple application of the AMGS method is not practical. To overcome this inefficiency, we modify the AMGS method so that it does not require the explicit evaluation of the matrix $\boldsymbol{A}$. In a similar way to the AMGS method, we can also modify the AMSOR method for solving LCPs in the rigid-body simulations.

As already pointed out at above, the direct computation of $\boldsymbol{A}$ is unfavorable for real-time simulations. Thus, we should avoid the computations of $\sum_{j=1}^{i-1} A_{ij}\lambda_j^{k+1}$ and $\sum_{j=i}^{m} A_{ij}\lambda_j^k$ in (15), which involve all the off-diagonal elements of $\boldsymbol{A}$.

To improve the computational efficiency, we introduce an intermediate variables $\lambda^{k+1,i} \in \mathbb{R}^m$ and $\boldsymbol{v}^{k+1,i} \in \mathbb{R}^n$. In particular, $\boldsymbol{v}^{k+1,i}$ stores information of applied impulse [9, 17]. For $i = 1, \ldots, m$, let

$$\lambda^{k+1,i} = \left(\lambda_1^{k+1} \ \ldots \ \lambda_{i-1}^{k+1} \ \lambda_i^k \ \ldots \ \lambda_m^k\right)^T \tag{18}$$

$$\boldsymbol{v}^{k+1,i} = \boldsymbol{v} + \hat{\boldsymbol{M}}\lambda^{k+1,i} \tag{19}$$

where $\hat{\boldsymbol{M}} = \boldsymbol{M}^{-1}\boldsymbol{J}^T$. By the definitions of $\boldsymbol{A}$ and $\boldsymbol{q}$, it holds

$$\sum_{j=1}^{i-1} A_{ij}\lambda_j^{k+1} + \sum_{j=i}^{m} A_{ij}\lambda_j^{k} + q_i = \left(\boldsymbol{A}\lambda^{k+1,i}\right)_i + q_i$$
$$= \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\lambda^{k+1,i}\right)_i + (\boldsymbol{J}\boldsymbol{v})_i + b_i = \left(\boldsymbol{J}\boldsymbol{v}^{k+1,i}\right)_i + b_i$$
$$= \sum_{j=1}^{m} J_{ij}v_j^{k+1,i} + b_i,$$

and this leads to

$$x_i^{k+1} = x_i^k + \Delta x_i^k = x_i^k - \frac{\gamma}{A_{ii}+\gamma\Omega_{ii}}\left(\sum_{j=1}^{m} J_{ij}v_j^{k+1,i} + b_i - 2\Omega_{ii}(\boldsymbol{x}_-^k)_i\right).$$

Due to the relation $\lambda^k = \frac{2}{\gamma}\boldsymbol{x}_+^k$, we can compute $\lambda^{k+1,i+1}$ by updating only the ith position of $\lambda^{k+1,i}$,

$$\lambda^{k+1,i+1} = \left(\lambda_1^{k+1} \ \dots \ \lambda_{i-1}^{k+1} \ \frac{2}{\gamma}(\boldsymbol{x}_+^{k+1})_i \ \lambda_{i+1}^{k} \ \dots \ \lambda_m^{k}\right)^T.$$

Thus, from (19), we obtain

$$\boldsymbol{v}^{k+1,i+1} = \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}(\lambda^{k+1,i+1} - \lambda^{k+1,i})$$
$$= \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}_{*i}(\lambda_i^{k+1,i+1} - \lambda_i^{k+1}),$$

where $\widehat{\boldsymbol{M}}_{*i}$ is the ith column of $\widehat{\boldsymbol{M}}$.

The following method summarizes the proposed AMGS method for rigid-body simulations.

Method 2 (the proposed AMGS method for rigid-body simulations)

Choose a nonnegative vector $\lambda^0 \in \mathbb{R}^m$ as an initial vector. Generate the iteration sequence $\{\lambda^k\}_{k=0}^{\infty}$ by the following procedure:

$\widehat{\boldsymbol{M}} = \boldsymbol{M}^{-1}\boldsymbol{J}^T$
$\boldsymbol{v}^{1,1} \leftarrow \boldsymbol{v} + \widehat{\boldsymbol{M}}\boldsymbol{\lambda}^0$
repeat $k = 1, 2, \dots$
  for $i = 1, 2, \dots, m$ ***do***
    $x_i^{k+1} \leftarrow x_i^k - \frac{\gamma}{A_{ii}+\gamma\Omega_{ii}}\left(\sum_{j=1}^{m} J_{ij}v_j^{k+1,i} + b_i - 2\Omega_{ii}(\boldsymbol{x}_-^k)_i\right)$
    $\lambda_i^{k+1} \leftarrow \frac{2}{\gamma}\max\left\{0, x_i^{k+1}\right\}$
    if $i = m$ ***then***
      $\boldsymbol{v}^{k+2,1} \leftarrow \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}_{*i}(\lambda_i^{k+1} - \lambda_i^k)$
    else
      $\boldsymbol{v}^{k+1,i+1} \leftarrow \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}_{*i}(\lambda_i^{k+1} - \lambda_i^k)$
until $\boldsymbol{x}^k$ *satisfies a certain convergence threshold*

In Sect. 5, we compare the computational costs of Method 1 and Method 2, which is an optimized version of Method 1 for interactive rigid-body simulations. As the experiments will show, Method 2 achieves considerably lower computational costs than that of Method 1 in all cases, and is suitable for interactive rigid-body simulations.

4.1 Linear complementarity problems with lower and upper bounds

In this section, we discuss a method for solving LCPs with lower and upper bounds on λ, which often occur in the formulations of contact constraints with friction [14, 17]. We call such LCPs with lower and upper bounds "Boxed LCPs (BLCPs)". Consider the following BLCP:

BLCP($\boldsymbol{q}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{A}$)

$$\begin{cases} \boldsymbol{A}\lambda + \boldsymbol{q} = \boldsymbol{w} \\ \boldsymbol{l} \le \lambda \le \boldsymbol{u} \\ \text{for each } i = 1, \dots, m, \begin{cases} w_i \ge 0 & (\lambda_i = l_i) \\ w_i \le 0 & (\lambda_i = u_i) \\ w_i = 0 & (l_i < \lambda_i < u_i). \end{cases} \end{cases}$$

Here, $\boldsymbol{l} \in \mathbb{R}^m$ and $\boldsymbol{u} \in \mathbb{R}^m$ are the lower and the upper bounds, respectively. Without loss of generality, we assume $0 \le l_i < u_i$ for each $i = 1, \dots, m$.

We define a projection function of λ to the interval $\boldsymbol{l}$ and $\boldsymbol{u}$ by

$$p_{\text{BLCP}}(\lambda) = \min\{\max\{\boldsymbol{l}, \lambda\}, \boldsymbol{u}\}.$$

Since $\boldsymbol{I}$ is a nonnegative vector, so is $p_{\mathrm{BLCP}}(\lambda)$. With this projection, we can give an AMGS method for Boxed LCPs as follows.

Method 3 (the AMGS method for Boxed LCPS in rigid-body simulations)

Choose a nonnegative vector $\lambda^0 \in \mathbb{R}^m$ as an initial vector. Generate the iteration sequence $\{\lambda^k\}_{k=0}^{\infty}$ by the following procedure:

$\boldsymbol{x}^0 \leftarrow \frac{\gamma}{2}\boldsymbol{\lambda}^0$
$\boldsymbol{\lambda}^1 \leftarrow \boldsymbol{\lambda}^0$
$\widehat{\boldsymbol{M}} = \boldsymbol{M}^{-1}\boldsymbol{J}^T$
$\boldsymbol{v}^{1,1} \leftarrow \boldsymbol{v} + \widehat{\boldsymbol{M}}\boldsymbol{\lambda}^0$
repeat $k = 1, 2, \ldots$
  for $i = 1, 2, \ldots, m$ ***do***
    $x_i^{k+1} \leftarrow x_i^k - \frac{\gamma}{A_{ii}+\gamma\Omega_{ii}}\left(\sum_{j=1}^{m} J_{ij}v_j^{k+1,i} + b_i - 2\Omega_{ii}(\boldsymbol{x}_-^k)_i\right)$
    $\lambda_i^{k+1} \leftarrow \frac{2}{\gamma}p_{BLCP}(x_i^{k+1})$
    if $i = m$ ***then***
      $\boldsymbol{v}^{k+2,1} \leftarrow \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}_{*i}(\lambda_i^{k+1} - \lambda_i^k)$
    else
      $\boldsymbol{v}^{k+1,i+1} \leftarrow \boldsymbol{v}^{k+1,i} + \widehat{\boldsymbol{M}}_{*i}(\lambda_i^{k+1} - \lambda_i^k)$
until $\boldsymbol{x}^k$ *satisfies a certain convergence threshold*

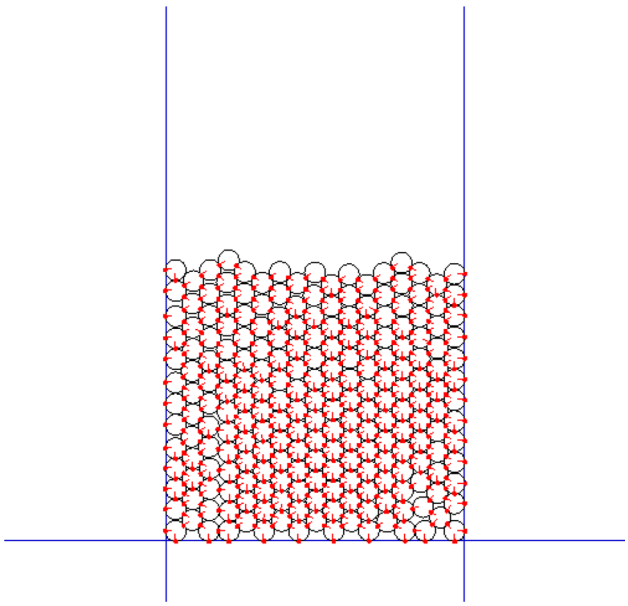

Fig. 1 A simulation of circles in Pool 1 and Pool 2. Small red points represent the contact points between the circles, and short red line segments the normal vectors at the contact points

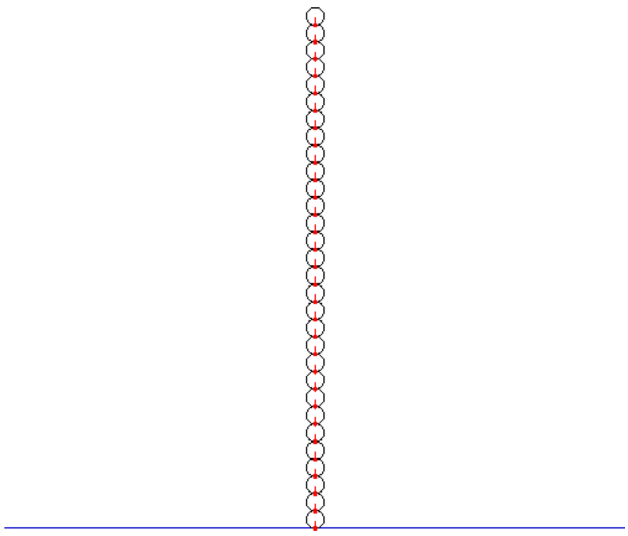

Fig. 2 A simulation of vertically stacked circles. Small red points represent the contact points of the circles, and short red line segments the normal vectors at the contact points

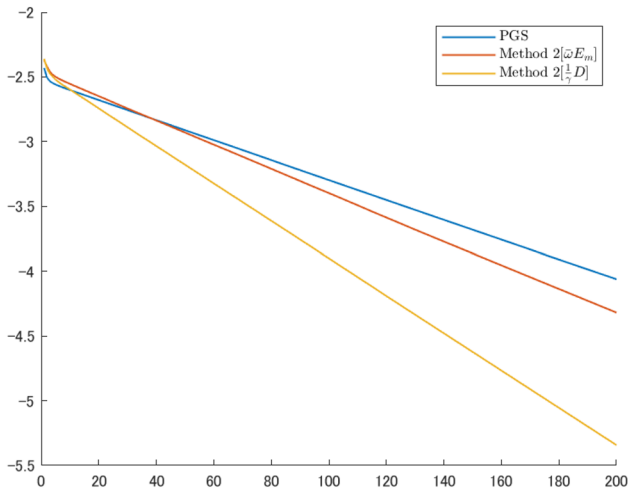

Fig. 3 Residuals of the sequence generated in the 100th simulation step for Pool 1

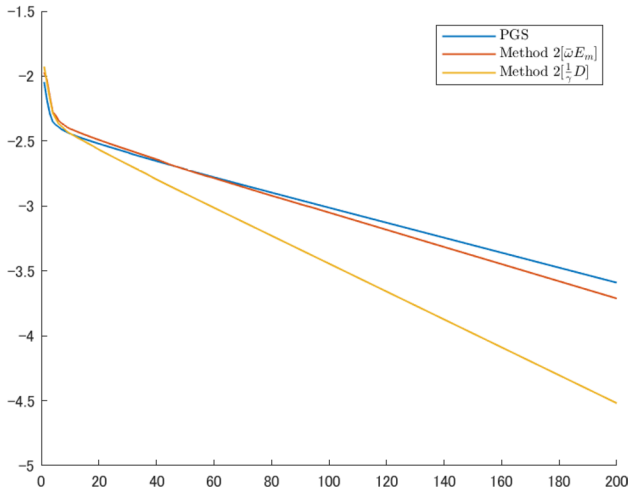

Fig. 4 Residuals of the sequence generated in the 100th simulation step for Pool 2

Note that BLCP($\boldsymbol{q}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{A}$) with $\boldsymbol{l} = \boldsymbol{0}$ and $\boldsymbol{u}$ being a very large vector represents the same problem as LCP($\boldsymbol{q}, \boldsymbol{A}$), thus Method 3 can be considered as a generalization of Method 1.

5 Numerical experiments

In this section, we show numerical results of several 2-dimensional examples to verify the numerical performance of the proposed method. All tests were performed on an Windows 8 computer with Intel Core i7-5500U (2.4 GHz CPU) and 8 GB memory space.

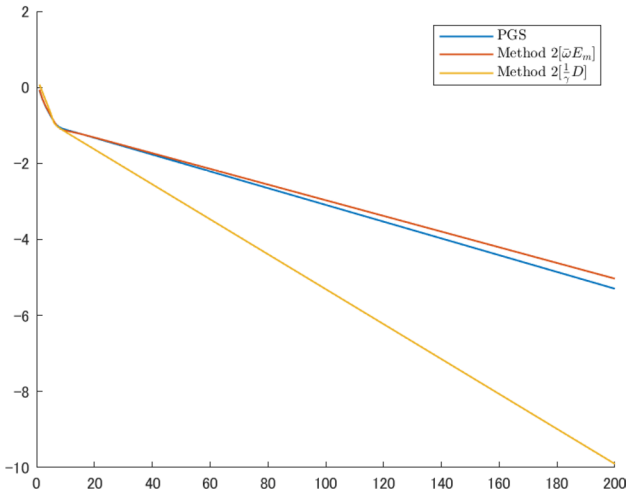

Fig. 5 Residuals of the sequence generated in the 100th simulation step for Stacking

In the numerical experiments, we utilized the warm-start strategy [9], that is, the final solution obtained in a simulation step will be used as the first guess of the solution in the next simulation step. More precisely, let $\{\lambda^{(t),k}\}_{k=0}^{\infty}$ denote the sequence generated for solving LCP($\boldsymbol{q}^{(t)}, \boldsymbol{A}^{(t)}$) at a simulation step t. We set the number of iterations in a single simulation step to 10, that is, $\lambda^{(t),10}$ is used as the first guess $\lambda^{(t+1),0}$. The initial point of the entire simulation is set as the zero vector ($\lambda^{(0),0} = \boldsymbol{0}$). To evaluate the accuracy of $\lambda^{(t),k}$, we use a residual function in [2]:

$$RES(\lambda^{(t),k}) = \left\| \min\{\lambda^{(t),k}, \boldsymbol{A}\lambda^{(t),k} + \boldsymbol{q}\} \right\|. \tag{20}$$

For the AMGS methods, we set $\gamma = 2$, and $\bar{\omega}$ is chosen as $\frac{\sum_{i=1}^{m} D_{ii}}{m\gamma}$ (the average of $D_{11}, \ldots, D_{mm}$ divided by γ) based on preliminary experiments.

For the numerical experiments, we use examples "Pool 1", "Pool 2" and "Stacking"; in the first two examples, rigid circles are stuffed into a small space, while, in the last case, rigid circles are vertically stacked.

Pool 1

Figure 1 displays an example with 221 circles in an area of 6 meters wide. All the circles have the same mass (2 kilograms) and the same radius (21 centimeters). The coefficients of friction are set to 0.1, and the coefficients of restitution are set to 0.2. When the coefficient of restitution is 1, collisions are perfectly elastic (i.e., no energy loss), and when the coefficient of restitution is 0, collisions are perfectly inelastic. The gravitational acceleration is set to 9.80665 m/s^2 in the downward direction in the figure.

In Pool 1, the size n in the matrices $\boldsymbol{J} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is 672, while the size m depend on the simulation step t, since the number of contact points between

Table 1 The iteration number and the computation time to reach 10^{-4}

Pool 1		
	Iteration	Time (s)
PGS method	1452	0.042
Method 1[$\frac{1}{\gamma}\boldsymbol{D}$]	766	1.251
Method 2[$\bar{\omega}\boldsymbol{E}_m$]	1522	0.055
Method 2[$\frac{1}{\gamma}\boldsymbol{D}$]	766	0.023
Pool 2		
	Iteration	Time (s)
PGS method	1359	0.042
Method 1[$\frac{1}{\gamma}\boldsymbol{D}$]	744	1.428
Method 2[$\bar{\omega}\boldsymbol{E}_m$]	1393	0.048
Method 2[$\frac{1}{\gamma}\boldsymbol{D}$]	744	0.028
Stacking		
	Iteration	Time (s)
PGS method	16186	0.162
Method 1[$\frac{1}{\gamma}\boldsymbol{D}$]	7754	0.293
Method 2[$\bar{\omega}\boldsymbol{E}_m$]	16431	0.183
Method 2[$\frac{1}{\gamma}\boldsymbol{D}$]	7754	0.066

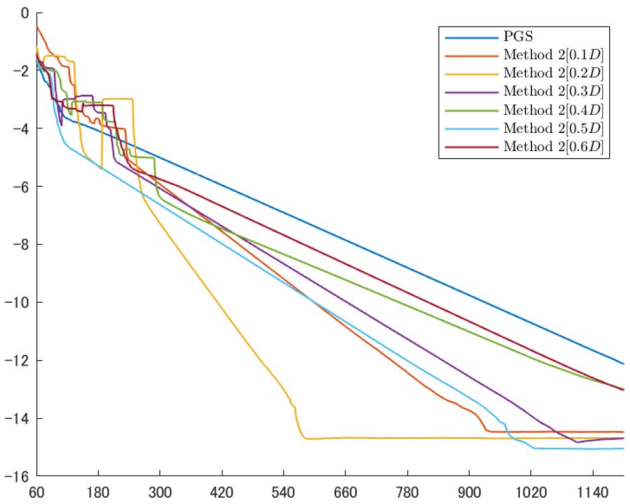

Fig. 6 The residuals in the entire simulation of Pool 1

the circles changes as the simulation proceeds. During the entire simulation, the average of m is 516.

Pool 2

The most part of this example is same as Pool 1, but the masses of the circles increases linearly from 1.0 kilograms to 3.0 kilograms in accordance with their initial heights from the ground; circles in higher positions have larger masses than those in lower positions, and the mass of the heaviest circles (the top circles) are three times of that of the lightest circles (the bottom circles). This is expected that the convergence will be slower, since it is hard for the lower (lighter) circles to support higher (heavier) circles. The sizes n and m of the matrices are the same as Pool 1.

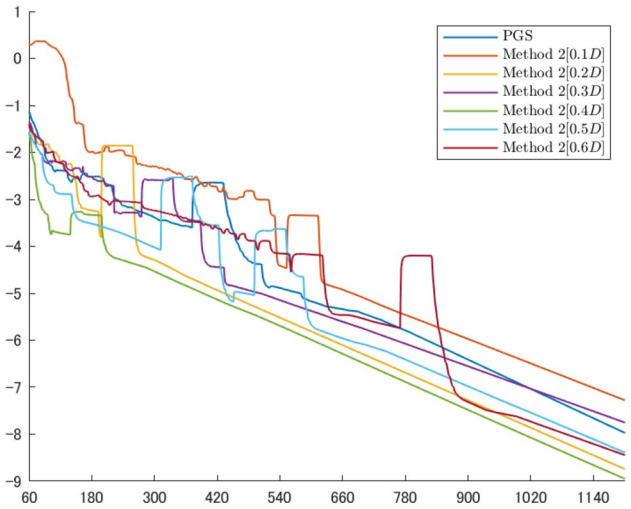

Fig. 7 The residuals in the entire simulation of Pool 2

Stacking

Figure 2 displays a simple example with 30 vertically stacked circles of 18-centimeter radius. All circles have the same masses (1.0 kilograms), and the coefficients of restitution are set to 0.2. The coefficients of friction are set to 0.1, but no frictional forces are produced because of the arrangement of the circles. The sizes of n and m (average) are 48 and 14, respectively.

5.1 Convergence in each simulation step

In each simulation step t, we execute only 10 iterations and we move to the next simulation step $t+1$ with the first guess $\lambda^{(t+1),0} = \lambda^{(t),10}$. In this subsection, we execute more iterations to compare the convergence of the PGS method and the proposed AMGS methods in each simulation step. The average values of $\bar{\omega}$ in the numerical experiments are 1.928, 1.086, and 3.872 in Pool 1, Pool 2 and Stacking, respectively.

For Pool 1, Fig. 3 plots $RES(\lambda^{(100),k})$ for $k = 1, \ldots, 200$ of the PGS method, Method 2 with $\boldsymbol{\Omega} = \bar{\omega}\boldsymbol{E}_m$ (shortly, Method 2[$\bar{\omega}\boldsymbol{E}_m$]) and Method 2 with $\boldsymbol{\Omega} = \frac{1}{\gamma}\boldsymbol{D}$ (shortly, Method 2[$\frac{1}{\gamma}\boldsymbol{D}$]). The horizontal axis is the iteration number k of $RES(\lambda^{(100),k})$. In a similar way, Figs. 4 and 5 show

Table 2 The computation time for 1000 simulation steps

Pool 1	
	Time (s)
PGS method	2.639
Method 2[0.1***D***]	2.083
Method 2[0.2***D***]	1.974
Method 2[0.3***D***]	2.051
Method 2[0.4***D***]	1.996
Method 2[0.5***D***]	1.968
Method 2[0.6***D***]	2.022
Pool 2	
	Time (s)
PGS method	2.093
Method 2[0.1***D***]	1.921
Method 2[0.2***D***]	1.907
Method 2[0.3***D***]	1.849
Method 2[0.4***D***]	1.904
Method 2[0.5***D***]	1.915
Method 2[0.6***D***]	1.915
Stacking	
	Time (s)
PGS method	0.095
Method 2[0.1***D***]	0.099
Method 2[0.2***D***]	0.081
Method 2[0.3***D***]	0.085
Method 2[0.4***D***]	0.085
Method 2[0.5***D***]	0.089
Method 2[0.6***D***]	0.082

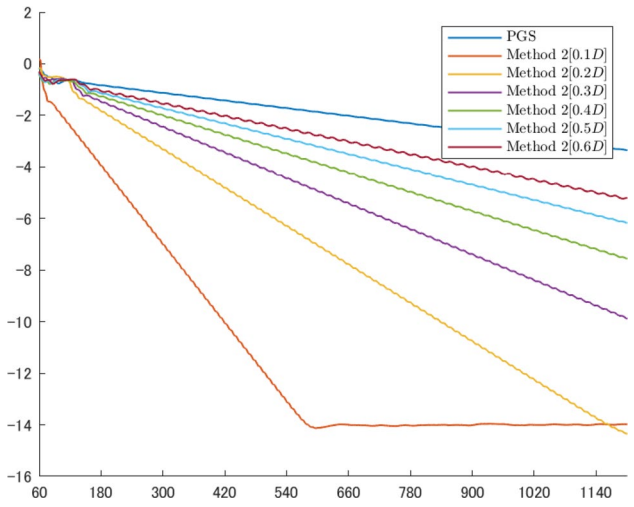

Fig. 8 The residuals in the entire simulation of Stacking

$RES(\lambda^{(100),k})$ of Pool 2 and Stacking, respectively. We chose the 100th simulation step, since the early steps contained a lot of noise and the warm-start strategy did not work effectively there.

From Figs. 3, 4, and 5 , we observe that Method 2 attains better convergence than the PGS method in most cases.

Table 1 reports the iteration number k and the computation time in seconds of each method to reach $RES(\lambda^{(100),k}) < 10^{-4}$. Since the standard AMGS method cannot directly handle contacts with frictions, coefficients of friction are set to 0 only for this experiment. Also, the computation time for the 30 circles in Stacking (Fig. 2) was too short to measure (shorter than 0.001 s), therefore we used 150 circles instead of 30 circles. Method 1 computes the coefficient matrix $\boldsymbol{A}$ explicitly, and we observe that Method 1 is the slowest in Table 1. This computation time is insufficient for real-time simulations.

In the comparison between Method 2[$\bar{\omega}\boldsymbol{E}_m$] and Method 2[$\frac{1}{\gamma}\boldsymbol{D}$], we can see that Method 2[$\frac{1}{\gamma}\boldsymbol{D}$] achieves better convergences than Method 2[$\bar{\omega}\boldsymbol{E}_m$]. Furthermore, Method 2[$\frac{1}{\gamma}\boldsymbol{D}$] achieves the smallest number of iterations and computation times in all three cases.

5.2 Convergence in entire simulation

In this subsection, we report the computational errors $RES(\lambda^{(t),10})$ along with the progress of simulation steps $t \geq 60$. We removed the first 60 steps from the figure, since the early steps contained much noise. In the previous subsection, we observed that Method 2[$\frac{1}{\gamma}\boldsymbol{D}$] is superior to Method 1 and Method 2[$\bar{\omega}\boldsymbol{E}_m$] in each simulation step. Thus, we use only Method 2[$\alpha\boldsymbol{D}$] in this subsection, changing the value of α.

In Fig. 6, the horizontal axis is the simulation step t, and the vertical axis is the residual $RES(\lambda^{(t),10})$. From Fig. 6, we observe that Method 2 converges faster than the PGS method for $t \geq 300$. Among different values of α, $\alpha = 0.2$ shows the fastest convergence in the figure.

The result of Pool 2 illustrated in Fig. 7 indicates that there are no clear differences of the convergence speed between the PGS method and the AMGS method with various values of α, but when $\alpha = 0.1$, the simulation is unstable during about the first 100 simulation steps.

From Fig. 8 for Stacking, in a similar way to Pool 1, we can again observe that the AMGS method with the smaller α gives the faster convergence, and the AMGS method with $\alpha = 0.6$ still converges faster than the PGS method.

Finally, Table 2 shows the entire computation time for 1,000 simulation steps, with 200 iterations for each step, that is, we computed $\lambda^{(1000),200}$. The entire computation time is shorter in the proposed method than in the PGS

method. As mentioned in Table 1, the convergence rate is better in the proposed method (Method 2[$\alpha \boldsymbol{D}$]) than in the PGS method, in other words, the proposed method simulates the circles more accurately than the PGS method. Therefore, the proposed method has the advantages for interactive rigid-body simulations.

6 Conclusion

We presented a numerical method based on the AMGS method for interactive rigid-body simulations. We established the convergence theorem of the AMGS method for the case the matrix $\boldsymbol{A}$ is positive definite and $\boldsymbol{\Omega} = \alpha \boldsymbol{D}$ with $\alpha > 0$. This case was examined in the numerical experiments, and we observed that the proposed method attained the better accuracy than the PGS method and the computation time of the proposed method was shorter than that of a simple application of the AMGS method.

In practical cases, however, determining a proper value of α is not simple. The numerical results showed that a smaller value of α gave a better convergence, but it is also shown that a too small value of α results in an unstable behavior. An approach that adaptively determines the value of α may resolve this problem, and we leave a discussion on such an approach as a future task of this paper. Further numerical experiments in 3-dimensional spaces that take frictions into consideration will be another topic of our future studies. Related to computing frictions correctly, applying the proposed method to more general form of LCPs, such as NCPs and HLCPs, will also be an interesting extension of this paper.

Funding This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and material All data generated or analysed during this study are included in the article.

Code Availability Not applicable

Disclosure of potential conflicts of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Research involving Human Participants and/or Animals Not applicable

Informed consent Not applicable

References

1. Anitescu M, Potra FA (1997) Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dyn 14(3):231–247
2. Bai ZZ (2010) Modulus-based matrix splitting iteration methods for linear complementarity problems. Numer Linear Algebra Appl 17(6):917–933
3. Baraff D (1993) Non-penetrating rigid body simulation. In: Eurographics '93 State of the art reports
4. Baraff D (1994) Fast contact force computation for nonpenetrating rigid bodies. In: Proceedings of the 21st annual conference on computer graphics and interactive techniques. ACM, pp 23–34
5. Bender J, Erleben K, Trinkle J (2014) Interactive simulation of rigid body dynamics in computer graphics. In: Computer Graphics Forum, vol 33. Wiley Online Library, pp 246–270
6. Bender J, Müller M, Macklin M (2015) Position-based simulation methods in computer graphics. In: Eurographics (tutorials), p 8
7. Bender J, Müller M, Otaduy M.A, Teschner M, Macklin M (2014) A survey on position-based simulation methods in computer graphics. In: Computer graphics forum, vol 33. Wiley Online Library, pp 228–251
8. Deul C, Charrier P, Bender J (2016) Position-based rigid-body dynamics. Comput Anim Virtual Worlds 27(2):103–112
9. Erleben K (2004) Stable, robust, and versatile multibody dynamics animation. Unpublished Ph. D. Thesis, University of Copenhagen, Copenhagen
10. Erleben K, Sporring J, Henriksen K, Dohlmann H (2005) Physics-based animation. Charles River Media Hingham
11. Mezzadri F (2019) On the equivalence between some projected and modulus-based splitting methods for linear complementarity problems. Calcolo 56(4):41
12. Mezzadri F, Galligani E (2020) Modulus-based matrix splitting methods for horizontal linear complementarity problems. Numer Algor 83(1):201–219
13. Nakaoka S, Hattori S, Kanehiro F, Kajita S, Hirukawa H (2007) Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In: IROS 2007 (IEEE/RSJ international conference on intelligent robots and systems, 2007). IEEE, pp 3641–3647
14. Poulsen M, Abel S.M.N, Erleben K (2010) Heuristic convergence rate improvements of the projected gauss-seidel method for frictional contact problems. In: 18th international conference in Central Europe on computer graphics, visualization and computer vision. Václav Skala-Union Agency, pp 135–142
15. Stewart DE, Trinkle JC (1996) An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. Int J Numer Meth Eng 39(15):2673–2691

16. Tang X, Paluszny A, Zimmerman RW (2014) An impulse-based energy tracking method for collision resolution. Comput Methods Appl Mech Eng 278:160–185
17. Tonge R, Benevolenski F, Voroshilov A (2012) Mass splitting for jitter-free parallel rigid body simulation. ACM Trans Graph (TOG) 31(4):105
18. Xie J, Chakraborty N (2016) Rigid body dynamic simulation with line and surface contact. In: 2016 IEEE international conference on simulation, modeling, and programming for autonomous robots (SIMPAR). IEEE, pp 9–15
19. Zheng H, Vong S (2020) On convergence of the modulus-based matrix splitting iteration method for horizontal linear complementarity problems of h+-matrices. Appl Math Comput 369:124890
20. Zheng N, Yin JF (2013) Accelerated modulus-based matrix splitting iteration methods for linear complementarity problem. Numer Algor 64(2):245–262