基于 GPU 的刚体动力学并行求解性能分析

梁睿凯，罗旭锟，郭煜中，何小伟

(中国科学院软件研究所人机交互北京重点实验室，北京 100190)

摘　　要：包含刚体和约束的多体动力学模拟在物理仿真中占有重要地位，广泛应用于工程分析、虚拟现实以及游戏动画等领域。传统的刚体物理引擎主要依赖于 CPU 进行计算，而在现代计算机图形学和实时物理模拟中，GPU 的并行计算能力被证明能够显著提高计算性能。为此，研究探索了 5 种基于雅可比方法的约束求解器在 GPU 上的实现并对其进行了性能与稳定性分析。具体包括：投影雅可比求解器(PJ)、结合投影雅可比与非线性雅可比的求解器(PJNJ)、投影雅可比与软约束求解器(PJSoft)、基于子步骤策略的雅可比求解器(TJ)和结合子步骤策略的雅可比与软约束求解器(TJSoft)。在基准测试中，软约束方法展现出平滑的约束冲量响应，且子步骤策略在处理高质量比和复杂场景时提供了更为稳定的解决方案。本研究为评估多体模拟中基于 GPU 的约束求解方案提供了新的视角，对实时物理模拟和交互式计算机图形学具有重要参考价值。

关　键　词：多体动力学模拟；GPU 实现；雅可比法；软约束；子步骤；性能与稳定性分析

中图分类号：TP 391　　**DOI**：10.11996/JG.j.2095-302X.2025030642

文献标识码：A　　**文 章 编 号**：2095-302X(2025)03-0642-13

Performance analysis of GPU-based parallel solvers for rigid body dynamics

LIANG Ruikai, LUO Xukun, GUO Yuzhong, HE Xiaowei

(Beijing Key Laboratory of Human-Computer Interaction, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Multi-body dynamic simulation involving rigid bodies and constraints plays a critical role in physical simulation and has widespread applications in engineering analysis, virtual reality, and game animation. Traditional rigid-body physics engines primarily rely on CPUs for computation. However, in modern computer graphics and real-time physics simulation, the parallel computing power of GPUs has been demonstrated to significantly enhance performance. This study explored the implementation of five Jacobian-based constraint solvers on the GPU and analyzed their performance and stability. These solvers included the projected Jacobi (PJ) solver, the combined projected Jacobi and nonlinear Jacobi (PJNJ) solver, the projected Jacobi with soft constraints (PJSoft) solver, the substep-based Jacobi (TJ) solver, and the substep-based Jacobi with soft constraints (TJSoft) solver. Benchmark tests revealed that the soft-constraint method provided smoother constraint impulse responses, while employing a substep strategy results in more stable solutions, particularly for high mass ratios and complex

收稿日期：2024-08-23；**定稿日期**：2025-01-13

Received：23 August，2024；**Finalized**：13 January，2025

基金项目：国家重点研发青年科学家项目(2021YFB1715800)；国家自然科学基金青年基金(62302490)

Foundation items：National Key R&D Program of China (2021YFB1715800); National Natural Science Foundation of China (62302490)

第一作者：梁睿凯(2001–)，男，硕士研究生。主要研究方向为基于物理的仿真、刚体动力学仿真。E-mail：liangruikai23@mails.ucas.ac.cn

First author：LIANG Ruikai (2001–), master student. His main research interests cover physics-based simulation and rigid body dynamics simulation. E-mail：liangruikai23@mails.ucas.ac.cn

通信作者：何小伟(1985–)，男，研究员，博士。主要研究方向为计算机图形学与物理仿真。E-mail：xiaowei@iscas.ac.cn

Corresponding author：HE Xiaowei (1985–), researcher, Ph.D. His main research interests cover computer graphics and physical simulation. E-mail：xiaowei@iscas.ac.cn

scenarios. Overall, this work offered a fresh perspective on evaluating GPU-based constraint solver strategies in multi-body simulations and served as an important reference for real-time physics simulation and interactive computer graphics.

Keywords: multi-body dynamic simulation; GPU implementation; Jacobi method; soft constraints; substep; performance and stability analysis

物理仿真是许多交互式计算机图形应用程序的重要组成部分，无论是工程分析或是游戏动画，其均能为虚拟世界增添逼真的物理行为，使用户和观众感受到更真实的互动和场景。为了实现这一目标，虚拟环境中的物理系统通常被描述为由多个刚体组成的多体系统。在物理仿真中，刚体模型是基础组件之一。刚体指的是在整体位移的情况下，其形状和大小保持不变的物体。这意味着刚体内部不允许任何变形，各部分之间的距离保持恒定。这种模型简化了计算，因为可以忽略内部变形带来的复杂性，只需关注刚体的位移和旋转。

刚体之间的交互通过约束来实现，确保物理系统遵循现实世界的物理法则，并模拟出逼真的运动和碰撞效果。约束主要分为双边约束和单边约束。前者模拟了刚体之间的关节连接和关节运动。常见的双边约束包括铰链、球形关节和滑动关节等，确保连接在一起的刚体按照预定的方式相对运动。例如，铰链关节允许 2 个刚体围绕一个固定轴旋转，而球形关节允许自由旋转但不允许平移。后者通常用于表示刚体之间的直接接触，特点是在特定条件下施加约束力。例如，当 2 个刚体接触时，约束力会阻止刚体进一步穿透对方，而当其分开时，约束力则不会存在。

一个典型的物理引擎包含了各种关节约束和碰撞摩擦约束，使用基于迭代法的线性求解器来解决这些约束并计算需要施加在物体上的力，以保持约束的满足。然后，使用数值积分技术(例如半隐式欧拉方法)，可以找到物体的新位置和速度，从而模拟其随时间的变化。

常见的游戏物理引擎中，诸如 Box2d[1]，使用基于投影高斯赛德尔方法(projected Gauss-Seidel, PGS)的线性求解器来解决约束问题。然而，由于高斯赛德尔方法的顺序执行性，这些引擎大多数采用 CPU 实现方式，导致效率相对较低。为了提高效率，GPU 并行实现的求解器通常采用加权雅可比迭代法(Weighted Jacobi)。加权雅可比迭代法通过引入一个阻尼因子(也称为加权因子)，使得每次迭代的更新步长变小，从而提高迭代过程的收敛性。这种方法在 GPU 上的并行计算能力下，可以更加有效地求解物理引擎中的约束问题，显著提升计算效率和性能。

本文的工作重点是比较多个基于 GPU 实现的雅可比迭代法的约束求解器的效率、收敛性、整体性能和稳定性等，为实际应用中的求解器选择提供指导，并为实时物理仿真和交互式计算机图形学中的多体系统约束求解提供有价值的参考。

1　相关工作

多位研究者将包含接触与摩擦的刚体动力学约束问题建模为线性互补性问题(linear complementarity problem, LCP)[2-4]。在早期研究中，接触模型通常在力-加速度层面离散化，分别处理碰撞与静止接触。随着时间步进方法的引入，接触模型实现了统一，并转向基于位置-冲量[4]或速度-冲量[5]的离散化。尽管在速度层面上，碰撞与摩擦约束可被建模为线性互补性问题，但此方法会导致位置误差，因此需采用约束稳定化技术来弥补该缺陷[6]。

在解决刚体系统中的碰撞摩擦互补性问题或关节约束问题时，研究者可采用直接或迭代方法。其中，PGS 求解器因其广泛应用而备受关注[7-11]。该方法可进一步拓展为基于序列冲量的求解器[12]，用于解决约束并计算施加在刚体上的力，以保持约束的满足。基于 Jacobi 方法的求解器较少使用，因其在处理某些刚体系统时存在不收敛的问题，即便收敛，其收敛速度也通常慢于 PGS。为保证或改善 Jacobi 方法的收敛性，相关文献中提出了多种方法，如线搜索、块处理和缩放策略[9,13]。

在硬件实现方面，尽管许多刚体接触动力学引擎如 Box2D[1]和 Bullet[14]主要在 CPU 上串行实现，但随着 GPU 等并行设备计算能力的增强，适用于并行设备的算法逐渐受到关注。并行刚体求解器的研究案例包括文献[3]、文献[10-11]和文献[15]。TONGE 等[16]提出了一种改进的投影 Jacobi 方法，同时兼备了保证 PGS 的收敛性和 Jacobi 方法的无

抖动性及其并行扩展能力。通过将每个接触块的有效质量中的质量项除以该块的接触数，并使用整个块的质量来施加冲量，确保了算法的高效性和扩展性。此方法同样适用于处理关节约束。

近些年有些研究者也开始研究非线性约束进行建模。MACKLIN 等[17]提出了一种基于 GPU 上实现的非光滑牛顿迭代的框架，能够直接解决非线性互补问题(nonlinear complementarity problem, NCP)，从而支持非线性动力学模型，包括超弹性可变形体和关节刚性机制。通过引入新的互补预处理器和几何刚度近似方法，提高求解过程的收敛性和鲁棒性。

2 约束动力学

2.1 动力学

刚体约束动力学的主要目的是模拟多刚体系统中物体的运动和相互作用，同时确保系统内外的约束条件得到满足。约束条件可以包括碰撞和摩擦约束、或是铰链、滑块等关节约束，对多刚体系统中各个刚体的相对运动进行限制。刚体约束动力学的理论基础主要包括经典力学和数值优化技术。经典力学中的牛顿运动定律是描述刚体运动的基本方程，而数值优化技术则用于在满足约束条件的前提下求解系统的运动方程。对此部分的相关研究已有详细阐述[5]，因此本文仅作简要说明。

考虑一个包括 n 个刚体的三维场景，广义坐标 $\boldsymbol{x}\in R^{6n}$，包含刚体的平移和旋转量。外力 $\boldsymbol{F}_{\text{ext}}\in R^{6n}$，约束力 $\boldsymbol{F}_c\in R^{6n}$，惯性矩阵 $\boldsymbol{M}\in R^{6n\times 6n}$，描述刚体的质量分布和转动惯量。根据牛顿运动定律可知该多体系统中刚体的运动方程为

$$\boldsymbol{M}\ddot{\boldsymbol{x}}=\boldsymbol{F}_{\text{ext}}+\boldsymbol{F}_c \tag{1}$$

常用于多体系统的运动方程求解采用半隐式欧拉法(Semi-implicit Euler method)[18]，通过在速度更新时使用当前时刻的加速度，而在位置更新时使用更新后的速度，能够在大多数实际应用中提供足够的稳定性，同时保持较低的计算复杂度。假设使用时间步长Δt 进行迭代，可得到

$$\begin{aligned}\boldsymbol{v}_{i+1}&=\boldsymbol{v}_i+\Delta t\boldsymbol{M}^{-1}(\boldsymbol{F}_{\text{ext}}+\boldsymbol{F}_c)\\ \boldsymbol{x}_{i+1}&=\boldsymbol{x}_i+\Delta t\boldsymbol{v}_{i+1}\end{aligned} \tag{2}$$

式中：$\boldsymbol{F}_c$ 为约束力，确保给定的约束在整个模拟过程中始终得到满足。每对刚体约束使用约束方程 $C(\boldsymbol{x}_1,\boldsymbol{x}_2)=0$ 进行建模，进行一次时间微分将约束与常微分方程结合起来，得到速度层面的约束[19]

$$\dot{C}(\boldsymbol{x})=\frac{\mathrm{d}C}{\mathrm{d}\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=\boldsymbol{J}\boldsymbol{v}=0 \tag{3}$$

式中：$\boldsymbol{J}\in R^{m\times 12}$ 为约束的雅可比矩阵；m 为约束的数量。当约束满足时，约束力不做功，因此约束力为

$$\boldsymbol{F}_c=\boldsymbol{J}^{\mathrm{T}}\lambda \tag{4}$$

式中：$\boldsymbol{\lambda}$为约束的拉格朗日算子，用以描述约束力的大小。最后得到计算约束力施加冲量大小$\mu=\lambda\Delta t$ 的公式，从而通过计算得到的冲量大小更新刚体的广义速度和坐标[3]，即

$$\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{\mu}=-\boldsymbol{J}\boldsymbol{M}^{-1}(\boldsymbol{P}+\Delta t\boldsymbol{F}_{\text{ext}}) \tag{5}$$

式中：$\boldsymbol{P}$ 为刚体的动量。求解该线性方程系统的方法有多种，其中常用的迭代方法包括 PGS 和投影雅各比法(projected Jacobi，PJ)。由于 PJ 具有天然的并行性，因此特别适用于基于 GPU 的刚体物理引擎。在本文后续部分，将对该方法进行详细讨论。

2.2 碰撞和摩擦约束(图 1)

三维空间中的碰撞约束确保 2 个接触的物体不会互相穿透，也称为穿透约束。

假设有 2 个刚体 B_1 和 B_2 处于接触状态，分别将 B_1 和 B_2 上的 2 个接触点(在世界空间坐标中)称为 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$。如果 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 分别是 B_1 和 B_2 的质心位置，则 $\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 是从各自质心到接触点的向量。为了找到穿透约束，需要计算 2 个接触物体的穿透深度。即接触点 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 之间在接触法线 $\boldsymbol{n}_1$ 方向上的距离。这个穿透深度是穿透约束函数[19]，即

$$C_{\text{pen}}=(\boldsymbol{p}_2-\boldsymbol{p}_1)\cdot\boldsymbol{n}_1=(\boldsymbol{x}_2+\boldsymbol{r}_2-\boldsymbol{x}_1-\boldsymbol{r}_1)\cdot\boldsymbol{n}_1 \tag{6}$$

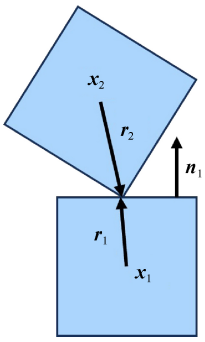

图 1 碰撞约束定义

Fig. 1 Collision constraint definition

2 个物体之间的穿透约束在其分开(未接触)时是大于零的，而在 2 个物体穿透时是小于零的，静止接触是等于零的。接触约束描述物体未穿透状态。因此，穿透约束在以下情况下有效，即

$$C_{\text{pen}}\geqslant 0 \tag{7}$$

同时还需要另一个约束来模拟 2 个接触物体

之间的摩擦力，称为摩擦约束，其直接定义在速度约束层面。假设 2 个正交于接触法向向量 $\boldsymbol{n}$ 的单位向量 $\boldsymbol{u}_1$ 和 $\boldsymbol{u}_2$。摩擦约束用于限制刚体在这 2 个方向上的运动。由此可得到摩擦约束函数[19]为

$$\begin{aligned}\dot{C}_{\text{fric1}} &= (\boldsymbol{v}_2 + \boldsymbol{w}_2 \times \boldsymbol{r}_2 - \boldsymbol{v}_1 - \boldsymbol{w}_1 \times \boldsymbol{r}_1)\cdot \boldsymbol{u}_1 \\ \dot{C}_{\text{fric2}} &= (\boldsymbol{v}_2 + \boldsymbol{w}_2 \times \boldsymbol{r}_2 - \boldsymbol{v}_1 - \boldsymbol{w}_1 \times \boldsymbol{r}_1)\cdot \boldsymbol{u}_2\end{aligned} \tag{8}$$

摩擦约束定义为限制刚体在碰撞平面的相对移动。当以下条件满足时，约束得到满足，即

$$\begin{aligned}\dot{C}_{\text{fric1}} &= 0 \\ \dot{C}_{\text{fric2}} &= 0\end{aligned} \tag{9}$$

2.3　关节约束

1) 球窝关节仅允许 2 个物体之间的任意旋转，但不允许相对平移。其有 3 个自由度。要创建一个球窝关节，用户只需要在世界空间坐标中指定一个锚点。在关节创建时，本文将锚点存储在每个物体的局部空间中。然后，在每个帧中，对每个物体，将局部空间的锚点转换回世界空间，以便为每个物体 B_i 得到锚点 $\boldsymbol{p}_i$。

球窝关节不限制旋转运动，因此，定义限制相对平移的位置约束函数[19]为

$$C_{\text{trans}} = \boldsymbol{x}_2 + \boldsymbol{r}_2 - \boldsymbol{x}_1 - \boldsymbol{r}_1 \tag{10}$$

式中：$\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 分别为物体 B_1 和 B_2 的世界空间位置；$\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 分别为从物体中心到各自的世界空间锚点 ($\boldsymbol{p}_i$=$\boldsymbol{x}_i$+$\boldsymbol{r}_i$)的向量。这个约束规定了 2 个物体的锚点的世界空间位置必须相等，其从系统中移除了 3 个平移自由度。当以下条件满足时，约束得到满足，即

$$C_{\text{trans}} = 0 \tag{11}$$

2) 滑块关节只允许 2 个物体在某一给定方向上相对平移。其只有一个自由度。滑块关节通过一个滑动轴 a 定义，该轴是为相对平移的方向，并通过一个在世界空间坐标中的锚点定义。锚点定义同球窝关节保持一致。由于滑动关节约束了相对平移的 2 个自由度和相对旋转的 3 个自由度，因此位置约束函数[19]为

$$\begin{aligned}C_{\text{trans}} &= \begin{bmatrix}(\boldsymbol{x}_2 + \boldsymbol{r}_2 - \boldsymbol{x}_1 - \boldsymbol{r}_1)\cdot \boldsymbol{n}_1 \\ (\boldsymbol{x}_2 + \boldsymbol{r}_2 - \boldsymbol{x}_1 - \boldsymbol{r}_1)\cdot \boldsymbol{n}_2\end{bmatrix} \\ C_{\text{rot}} &= \begin{bmatrix}\theta_{2x} - \theta_{1x} \\ \theta_{2y} - \theta_{1y} \\ \theta_{2z} - \theta_{1z}\end{bmatrix}\end{aligned} \tag{12}$$

式中：$\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 分别为物体 B_1 和 B_2 的世界空间位置；$\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 分别为从物体中心到各自的世界空间锚点 ($\boldsymbol{p}_i$=$\boldsymbol{x}_i$+$\boldsymbol{r}_i$)的向量；θ_{ix}，θ_{iy} 和 θ_{iz} 分别为物体 B_i 绕 x 轴、y 轴和 z 轴的方向角。2 个向量 $\boldsymbol{n}_1$ 和 $\boldsymbol{n}_2$ 是与滑动轴 a 正交的 2 个单位正交向量。在关节创建时，将滑动轴 a 转换到 B_1 物体的局部空间，得到向量 $\boldsymbol{a}_l$。然后，在每个帧中，将向量 $\boldsymbol{a}_l$ 转换回世界空间以获得向量 $\boldsymbol{a}_w$。接着，创建 2 个与 $\boldsymbol{a}_w$ 正交的正交向量 $\boldsymbol{n}_1$ 和 $\boldsymbol{n}_2$。平移约束限制不发生相对于滑动轴 a 的正交平移。旋转约束意味着 2 个物体之间不发生相对旋转。当以下条件满足时，约束得到满足，即

$$\begin{aligned}C_{\text{trans}} &= 0 \\ C_{\text{rot}} &= 0\end{aligned} \tag{13}$$

3) 铰链关节仅允许 2 个物体围绕单一轴线进行相对旋转。其有一个自由度。铰链关节由一个单位长度的世界空间铰链轴向量 $\boldsymbol{a}$ 定义，该向量是 2 个物体可以绕其旋转的轴线，以及一个世界空间锚点。锚点定义同球窝关节保持一致。由于铰链关节约束了相对平移的 3 个自由度，和相对旋转的 2 个自由度，因此位置约束函数[19]为

$$\begin{aligned}C_{\text{trans}} &= \boldsymbol{x}_2 + \boldsymbol{r}_2 - \boldsymbol{x}_1 - \boldsymbol{r}_1 \\ C_{\text{rot}} &= \begin{bmatrix}\boldsymbol{a}_1 \cdot \boldsymbol{b}_2 \\ \boldsymbol{a}_1 \cdot \boldsymbol{c}_2\end{bmatrix}\end{aligned} \tag{14}$$

这里平移位置约束与球窝关节相同，规定了 2 个物体的锚点的世界空间位置必须相等。而旋转位置约束中，$\boldsymbol{a}_1$，$\boldsymbol{a}_2$ 分别为由物体局部空间存储的旋转轴转换到世界坐标系下的铰链轴向量，而 $\boldsymbol{b}_2$，$\boldsymbol{c}_2$ 为 2 个正交与向量 $\boldsymbol{a}_2$ 的单位正交向量。旋转约束意味着，物体之间唯一允许的旋转是围绕铰链轴进行的。当以下条件满足时，约束得到满足，即

$$\begin{aligned}C_{\text{trans}} &= 0 \\ C_{\text{rot}} &= 0\end{aligned} \tag{15}$$

4) 固定关节不允许关节两端的物体发生任何相对运动(既不平移也不旋转)，其自由度为零。固定关节由一个锚点定义。锚点定义同球窝关节保持一致。

应用于固定关节约束的平移位置约束与球窝关节相同，而旋转位置约束与滑动关节相同，限制 2 个物体之间的相对平移和旋转，位置约束函数[19]为

$$\begin{aligned}C_{\text{trans}} &= \boldsymbol{x}_2 + \boldsymbol{r}_2 - \boldsymbol{x}_1 - \boldsymbol{r}_1 \\ C_{\text{rot}} &= \begin{bmatrix}\theta_{2x} - \theta_{1x} \\ \theta_{2y} - \theta_{1y} \\ \theta_{2z} - \theta_{1z}\end{bmatrix}\end{aligned} \tag{16}$$

当以下条件满足时，约束得到满足，即

$$\begin{aligned} C_{\text{trans}} &= 0 \\ C_{\text{rot}} &= 0 \end{aligned} \tag{17}$$

3 基于投影雅各比法的求解器

3.1 投影雅各比法

投影雅可比法通过独立求解每个约束，然后将结果合并，从而可应用于高效的并行计算。其核心思想是将复杂的约束问题分解成多个简单的子问题，每个子问题对应一个约束。在每次迭代中，对每个约束独立进行求解，得到其对解的贡献。随后，合并所有的约束贡献，更新整体解。因此，所有约束的求解过程可以并行执行，从而大大提高计算效率。一般而言，应用于公式雅各比迭代形式可抽象为

$$\Delta\boldsymbol{\mu}_j = \boldsymbol{\mu}_j^{i+1} - \boldsymbol{\mu}_j^{i} = \boldsymbol{K}_j^{-1}(\boldsymbol{\eta}_j - \boldsymbol{J}_j\boldsymbol{M}^{-1}\boldsymbol{J}_j^{\mathrm{T}}\boldsymbol{\mu}^{i}) \tag{18}$$

式中：j 为约束的序号；$\boldsymbol{\eta}_j = -\boldsymbol{J}_j\boldsymbol{M}^{-1}(\boldsymbol{P} + \Delta t\boldsymbol{F}_{\text{ext}})$；$\boldsymbol{K}_j = \boldsymbol{J}_j\boldsymbol{M}^{-1}\boldsymbol{J}_j^{\mathrm{T}}$。计算出约束冲量的变化量后，通过投影操作对累积约束冲量进行调整，以限制约束冲量的大小(如为确保接触力为非负值，当累计接触约束冲量小于 0 时进行截断)。虽然雅各比迭代更适合 GPU 上的并行化实现，但大多数刚体系统使用雅可比法并不收敛。本文基于 GPU 实现的投影雅各比法基于 Mass splitting solver[16]的思想，对 $\boldsymbol{K}$ 进行修改，将有效质量中的每个物体质量项除以影响其接触点个数或关节连接个数，使用完整的质量来施加冲量。设 n 为涉及物体的接触点数量或关节连接数量，修改后的雅各比法为

$$\begin{aligned} \Delta\boldsymbol{\mu}_j &= \boldsymbol{\mu}_j^{i+1} - \boldsymbol{\mu}_j^{i} = \boldsymbol{K}_j^{-1}(\boldsymbol{\eta}_j - \boldsymbol{J}_j\boldsymbol{M}^{-1}\boldsymbol{J}_j^{\mathrm{T}}\boldsymbol{\mu}^{i}) \\ \boldsymbol{K}_j &= n\boldsymbol{J}_j\boldsymbol{M}^{-1}\boldsymbol{J}_j^{\mathrm{T}} \end{aligned} \tag{19}$$

雅各比迭代法作为一种经典的迭代方法，在求解线性方程组时展现出显著的并行优势。该方法在每次迭代过程中更新解向量的各个元素时，仅依赖于上一轮迭代的结果，且各元素之间不存在数据依赖关系。这一解耦特性使得雅各比迭代法特别适合并行计算环境。在每个迭代步骤中，不同的约束方程计算任务可以被分配到多个线程上同时处理。在 GPU 架构中，成千上万的核心能够并行执行这些约束方程的计算任务，从而显著提升计算速度和效率。

然而，尽管雅各比迭代法在并行实现方面具有明显优势，但与适用于 CPU 实现的高斯-赛德尔迭代法相比，其收敛速度通常较慢。这主要是因为雅各比方法在每次迭代中使用的是上一轮的全部解向量，而高斯-赛德尔方法则利用了最新的中间更新结果，从而加快了收敛过程。

在适配性方面，雅各比迭代法在 GPU 上的并行化实现具有明显的优势，能够充分利用 GPU 的大规模并行计算能力，适用于约束数量庞大的应用场景。

3.2 Baumgarte stablization

由于时间步长的限制和积分误差，约束往往无法被完美保持，从而导致物体逐渐偏离其应有位置或姿态。这类误差可以通过位置约束函数 $C(\boldsymbol{x})$进行量化。Baumgarte stablization[20]是一种用于数值解算机械系统中约束方程的技术，特别是在刚体动力学模拟中，用以改善约束条件在数值积分过程中的稳定性和准确性。Baumgarte stabilization 通过向系统的约束方程中引入一个补偿项来解决约束误差问题。速度约束增加了一个与位置误差成比例的反馈项，即

$$\boldsymbol{J}\boldsymbol{v} + \beta C(\boldsymbol{x}) = 0 \tag{20}$$

式中：β的值通常选在 0~1/Δt，系数为 1/Δt 的情况下尝试在一个时间步内解决所有位置误差，而系数为 0 则无效果。Baumgarte stabilization 在实际应用中简单而高效，主要不足在于物理上不够准确，可能导致非物理的行为，如能量增加或系统不稳定。

3.3 软约束(Soft constraint)

软约束(Soft constraint)[21]可以认为是 Baumgarte stablization 方法的一种演进。软约束模拟一个包含质量、弹簧和阻尼器的谐振子系统，在物理上类似于一个质量-弹簧-阻尼模型。在该模型中，弹簧代表恢复力，质量代表系统的惯性，而阻尼器则负责吸收系统能量，减少振动。具体来说，软约束通过模拟弹簧的弹性力来解决位置约束产生的误差问题，而阻尼则控制系统回到平衡状态的速度，从而避免过度振动。这种方法的优势在于其提供了一种直观的方式来调整系统的动态响应。系统的自然频率(以赫兹为单位) 和无量纲阻尼比是调整这些响应的关键参数。

一个质量-弹簧-阻尼系统为[21]

$$\begin{aligned} &\ddot{\boldsymbol{x}} + 2\xi\omega\dot{\boldsymbol{x}} + \omega^2\boldsymbol{x} = 0 \\ &\text{其中，}\ \omega = \sqrt{\frac{k}{m}},\ \ \xi = \frac{c}{2\omega m} \end{aligned} \tag{21}$$

式中：m 为物体的质量；k 为弹簧的弹性系数；ω 为自然频率，决定了系统在无阻尼情况下的振荡

频率；ξ为阻尼比，其是无量纲的，控制解中的振荡量。

从而应用于刚体系统中的速度约束公式可以被软化为[21]

$$\boldsymbol{J}\boldsymbol{v}+\frac{\beta}{\Delta t}C(\boldsymbol{x})+\gamma\boldsymbol{\mu}=0 \tag{22}$$

式中：γ为约束力混合(constraint force mixing, CFM)项，其作用是软化刚性约束。软约束的优点在于其可以抑制响应，并且可以通过指定以赫兹为单位的自然频率和无量纲的阻尼比来直观地调整。$\boldsymbol{\mu}$为累积冲量，冲量系数会随着总冲量的增大而减少增量冲量。

3.4　非线性雅各比迭代

非线性雅各比迭代算法的引入主要是为了解决在处理物理仿真中的 Baumgarte stabilization 方法所带来的能量不守恒问题。为了解决这一问题，使用非线性雅各比迭代算法的求解器采用了一种分阶段处理的策略。

在这种策略中，算法首先解决速度约束问题，确保所有的动力学约束在速度层面得到满足。此时，算法不会考虑由于积分和约束求解过程中引入的位置误差。完成速度约束的求解后，算法再通过非线性雅各比迭代法处理位置约束问题。该方法属于 PBD (position based dynamics)[22]类方法在 GPU 上的拓展，不同之处在于使用雅各比法替换传统的串行高斯赛德尔法。其核心是通过每一步的迭代逐步调整刚体的位置和姿态，使得其满足位置约束。对于每个约束 C_j，该方法首先将约束线性化，即

$$C_j(\boldsymbol{x}_i+\Delta\boldsymbol{x})\approx C_j(\boldsymbol{x}_i)+\nabla_x C_j(\boldsymbol{x}_i)\cdot\Delta\boldsymbol{x} \tag{23}$$

同时考虑物体质量和惯性的影响，得到每次迭代使得约束误差减小的位置修正项，即

$$\Delta\boldsymbol{x}=\frac{-C_j(\boldsymbol{x}_i)}{\nabla C_j M^{-1}\nabla C_j^{\mathrm{T}}}\cdot M^{-1}\nabla C_j(\boldsymbol{x}_i) \tag{24}$$

位置约束的求解器能够矫正位置错误而不会影响系统的动能，因为其并不直接修改物体的速度，而是通过调整位置来确保约束被满足。非线性雅各比迭代算法能够有效地在保持系统动能守恒的同时，确保所有的物理约束都能得到适当的处理，从而提高了物理模拟的准确性和稳定性。

3.5　子步骤

MACKLIN 等[23]的研究表明与执行较大时间步长并进行多次约束求解迭代相比，将时间步长细分为多个较小的子步长，并在每个子步长中只进行单次约束求解迭代，通常更为有效。这一现象可通过泰勒级数展开解释：随着时间步长的减小，近似的精度提高，非线性问题的影响减弱。

该方法涉及将每个大的视觉时间步长分解成多个较小的子步长，每个子步长中执行单次约束求解迭代。这有助于减少约束求解中的误差，并在处理各种刚性参数时提供更高的稳定性和健壮性。考虑到高斯-赛德尔迭代和雅各比迭代方法在收敛速度上的差异，当使用雅各比迭代法时，可能需要在每个子步长中进行多次迭代以确保收敛。

在刚体动力学模拟中，子步骤法的一个重要优点是，在子步骤中不需要重新检测碰撞点。相反，可以通过在局部坐标中存储接触点信息，并在各个子步骤中更新这些接触点。因此，可追踪 2 个物体间的接触点，并据此更新分离距离以适当地推开物体。尽管保持接触点在子步骤中不变是一种近似处理，实践表明，这种方法通常能达到良好的模拟效果。重新计算接触点将会消耗大量计算资源，因此采用更新接触点的策略使子步骤法成为一种高效的解决方案。

3.6　多种求解器实现

综上所述，通过各种策略融合可以得到 5 种不同却又彼此相关的约束求解器，这些求解器由于基于通用的投影雅各比迭代法并行求解约束，均可高效的在 GPU 上并行实现。图 2 展示了各求解器的算法流程上的异同点。

1) 投影雅可比求解器 (projected Jacobi solver, PJ)。主要利用投影雅各比迭代法处理速度约束，同时采用 Baumgarte stabilization 技术进行误差校正。Baumgarte stabilization 通过向速度约束方程中添加与位置误差成比例的反馈项，使得在有限的时间步内减少位置误差。这种求解器可能会引入非物理的反馈，如系统能量的增加。

2) 结合投影雅可比与非线性雅可比的求解器(projected Jacobi and non-linear Jacobi solve，PJNJ)。结合了投影雅各比方法和非线性雅各比迭代方法。首先，使用投影雅各比法处理速度约束，确保动力学约束在速度层面得到满足；然后采用非线性雅各比迭代法单独矫正位置约束误差。

3) 投影雅可比与软约束求解器(projected Jacobi with soft constraint solver, PJSoft)。在处理速度约束时使用投影迭代法，并引入软约束模型(质量-弹簧-阻尼系统) 来处理位置误差。软约束通过模拟质量-弹簧-阻尼系统来提供一个物理上更为合理的误差补偿机制，其通过阻尼和弹簧力来抑制系

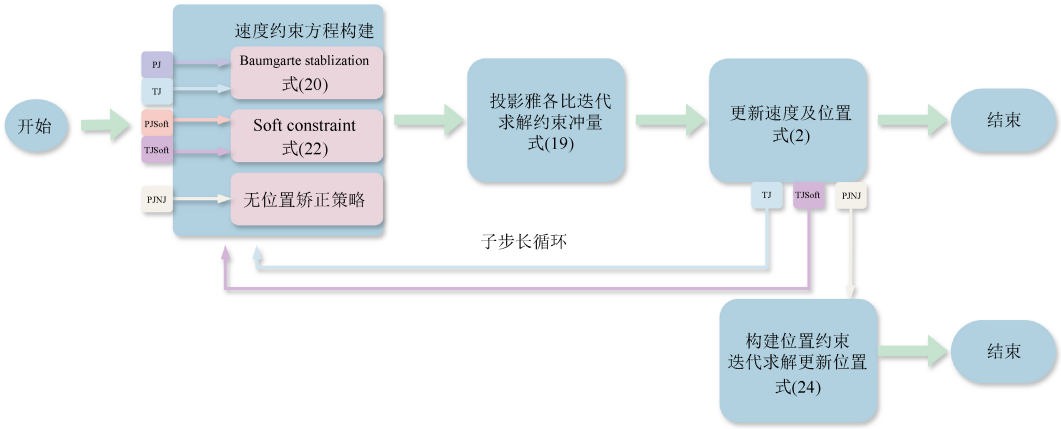

图 2 求解器算法流程

Fig. 2 Solvers algorithm flow

统过度振动，从而更精确地控制系统回到平衡状态的速度。

4) 基于子步骤策略的雅可比求解器(temporal Jacobi solver, TJ)。采用将较大的时间步长分解为多个小的子步长的方法，并在每个子步长内使用投影雅各比迭代法单独处理速度约束。Baumgarte stabilization 再被用来在每个子步长内快速校正位置误差。通过细分时间步长，可以提高仿真的精度和稳定性，减少大时间步长所带来的非线性问题影响。

5) 合子步骤策略的雅可比与软约束求解器(temporal Jacobi with soft constraint solver, TJSoft)。与 TJ 类似，但在处理位置误差时引入了软约束方法。其结合了时间步长的细分和软约束的物理模型，能够在提高仿真稳定性的同时，通过弹簧和阻尼的调节来细致控制系统的动态响应，从而优化模拟效果。

4 结果与讨论

本节首先测试了关节约束的正确性，其次展示了不同求解器收敛性和整体性能与稳定性的测量结果。本文算法实现基于 RTX 3060 Laptop GPU (6 GB)实现。

实验通过设置多个场景进行测试，为了保证实验的严谨性，时间步长均设为 16 ms，单个时间步内不同求解器中使用投影雅各比迭代法的总的迭代次数均为 300 次，而测量结果限制在 0~3 000 帧内。表 1 给出了各求解器的评估结果。

4.1 关节单元测试

首先通过成对刚体创建每种关节的单元测试验证了第 3.3 节公式中定义的关节约束的正确性，图 3 展示了多种关节约束的效果。其次构建了一个曲柄连杆结构(包含铰链关节、滑块关节和固定关节)的运动模拟，确保多关节运动的正确性。图 4 展示了该模拟的效果。曲柄能够正常驱动活塞前后移动。

4.2 稳定性测试

1) Box stack。本文建立了一个包含 125 个刚体方块堆叠的场景，保持方块堆叠的稳定性是一个刚体模拟中的一个关键挑战，尤其是在更大的时间步长下保持堆叠的稳定更为考验算法求解器的稳定性。如果接触约束和摩擦约束处理不当，盒子堆叠可能会出现穿透、滑动或非物理的跳动，影响模拟的真实性。在堆叠稳定性中，算法需要处理浮点数精度问题。物理引擎中的数值误差会在多个接触和碰撞计算中积累，最终导致模拟结果的不稳定。图 5 展示了不同求解器在该场景下的稳定性相关的对比结果。

表 1 求解器评估结果

Table 1 Solver evaluation results

算法求解器	步长	数值稳定性	数值精度	性能表现	应用场景
PJ	★★	★★	★★	★★★	性能优先的实时场景
PJSoft	★★	★★★	★★	★★★	性能优先且需要更高稳定性的实时场景
TJ	★★★	★★★★	★★★	★★	高精度大步长的实时场景
TJSoft	★★★	★★★★★	★★★	★★	高精度大步长且需要更高稳定性的实时场景
PJNJ	★★	★★	★★	★	单帧位置误差限度要求较高的场景

注：★表示越多越优。

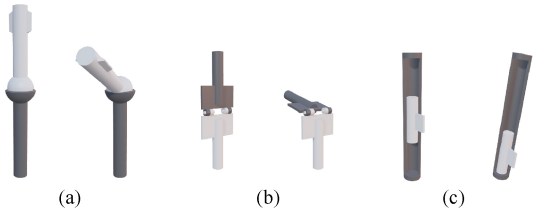

图 3　各种关节正确性验证((a)球窝关节；(b)铰链关节；(c)滑块关节)

Fig. 3　Verification of various joint correctness ((a) Ball-and-socket joint; (b) Hinge joint; (c) Siding joint)

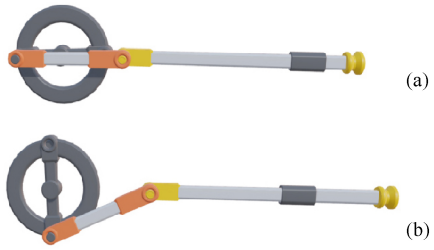

图 4　曲柄连杆多关节运动((a)静止；(b)运动)

Fig. 4　Crank-rocker multi-joint motion ((a) Rest; (b) Motion)

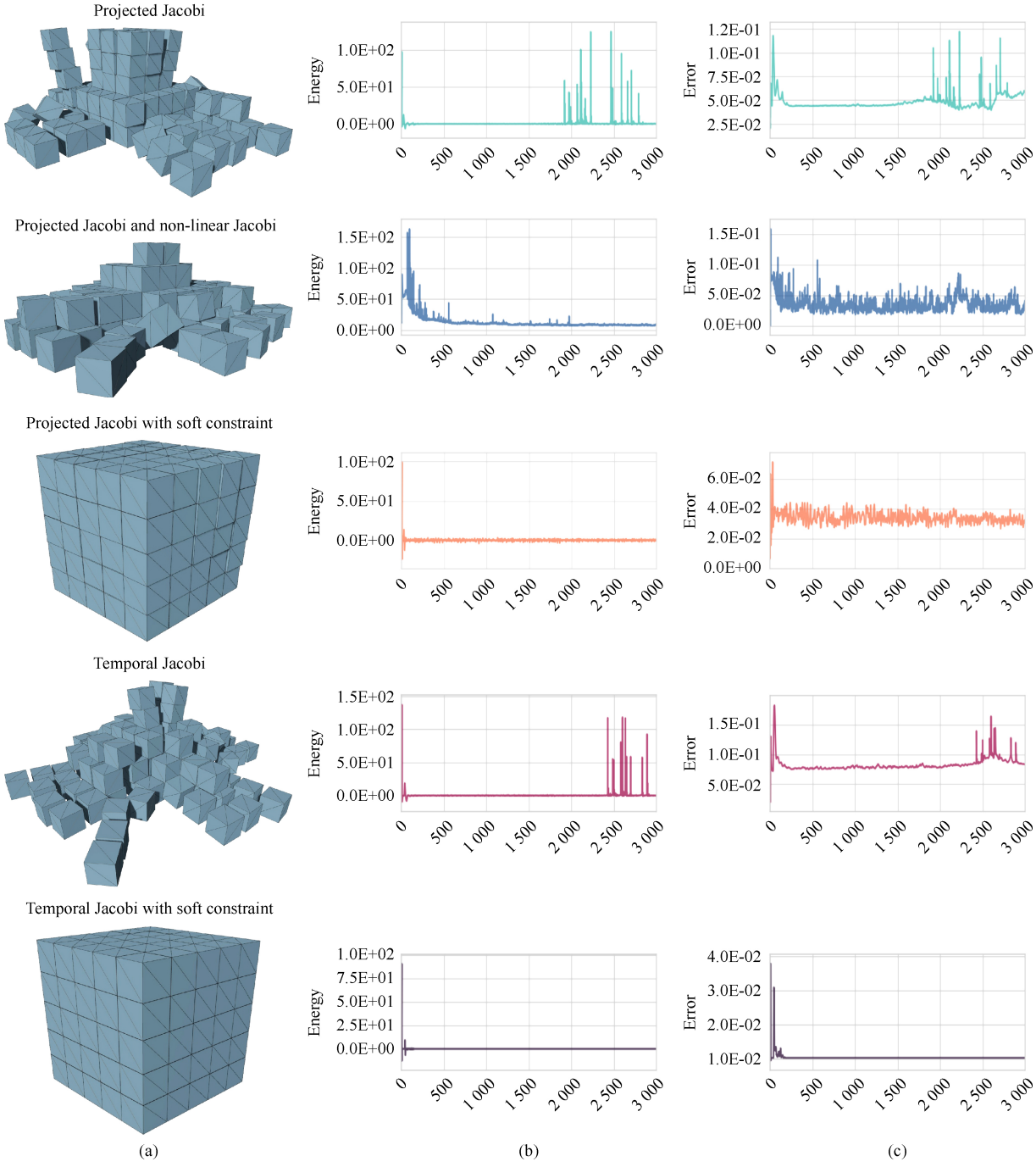

图 5　方块堆叠场景((a) Box stack 场景表现；(b)求解前后系统能量变化量；(c)约束误差变化曲线)

Fig. 5　Box stack scene ((a) Box stacking scene performance; (b) System energy change before and after solving; (c) Constraint error variation curve)

观察在随着模拟进行单个时间步内求解前后系统能量变化量的变化情况，系统能量的变化量定义为系统内刚体的动能的变化量。可以发现该场景是否稳定的关键在于是否能通过接触和摩擦有效地耗散刚体的势能。

采用 Baumgarte 稳定化策略的 2 个求解器 PJ 和 TJ，在前 2 000 帧内可以保持系统的稳定性。然而，由于 Baumgarte 参数人为设定，系统在物理上不够精确，这导致在 2 000 帧以后出现非物理的过冲行为，最终导致堆叠的盒子散开，即使是使用了子步骤法的 TJ 求解器也难以保持堆叠的稳定性。

采用非线性雅各比迭代策略修正位置误差的求解器 PJNJ 在步长较大时，在此类堆叠问题上效果较差，观察求解前后系统能量变化曲线不难发现，系统在初期就发生频繁的能量抖动变化，原因在于非线性雅各比迭代直接修正位置误差，尽管未将修正冲量引入动能，但却非物理地不断修改了刚体本身的势能。除此之外由于受到雅各比迭代的收敛速度的影响，速度层面约束求解也同样存在误差，而速度层面求解的约束误差通过数值积分积累到位置层面求解的约束误差，导致系统中的刚体发生频繁的抖动，产生不真实的场景表现。

而在该场景下使用 Soft constraint 策略替代 Baumgarte 稳定化策略，在一定程度上缓解了系统求解可能存在的过冲问题，通过调节弹簧和阻尼，能够更精细地控制系统的动态响应，使得在整体时间步推进过程中，由求解器导致的系统能量变化量较小，从而一定程度上达到了稳定。

所有求解器均采用半隐式积分法，当时间步长增大时，这种积分方法往往会伴随着数值不稳定和精度下降的问题。更小的时间步长可以捕捉更细微的系统变化，提高仿真的整体精度。而更大的时间步长在处理高频振荡或快速变化的系统时，误差会显著增大。因此进一步地，将子步骤策略与 Soft constraint 策略结合使用，可以显著增强系统的稳定性。子步骤法通过减小时间步长，提高了近似的精度，减少了非线性问题的影响。这种结合策略有效地改进了系统的动态响应，确保了在更长时间步内的稳定性。

而系统约束误差变化情况则展示了不同策略构成的求解器对于误差修正的效率和稳定性上存在差异，误差的大小并不是影响堆叠性能的唯一因素，维持稳定的关键在于求解器是否存在打破平衡的过度修正，除 TJSoft 求解器和 PJSoft 外，在约束误差上均呈现出强烈的陡然波动的情况，而这很容易导致盒子堆叠的平衡状态被打破，导致堆叠失败，方块四处散开的情景。

上述结果表明单纯依赖 Baumgarte 稳定化策略的求解方法，在长期模拟中可能会出现物理不准确的问题，特别是在参数设定不当的情况下。相比之下，Soft constraint 策略提供了一种更为灵活的解决方案，通过弹簧和阻尼的调节，可以细致地控制系统能量的变化。而将子步骤策略与 Soft constraint 策略结合使用，进一步展示了其在提高系统稳定性方面的有效性。子步骤策略通过缩短每个时间步的长度，减少了由于非线性引起的计算误差，从而提高了整体求解的精度。该方法不仅在短期内保持了系统的稳定性，还在较长时间的模拟过程中展现出良好的稳定性能。

2) Chain and ball。本文建立了一个包含 50 个方块与一个球体通过铰链关节连接的场景，球体与盒子的质量比为 300∶1。起始状态为水平状态，由方块组成的铰链会约束球体做圆周运动。高质量比的场景对于测试求解器的稳定性有着重要的意义。质量比较大的场景会在仿真过程中引入显著的数值不稳定性。当 2 个物体的质量差距悬殊时，较重的球体会对较轻的盒子施加巨大的力，这种力的作用会使得盒子在碰撞和运动时产生剧烈的响应。算法需要能够精确地处理这些力，以确保系统的稳定和准确性。除此之外，在高质量比的情况下，保持铰链关节的稳定性变得更加困难，因为较重的球体可能会对关节产生巨大的扭矩和剪切力。图 6 展示了不同求解器在该场景下的稳定性相关的对比结果。

在该场景中，未使用子步骤策略的求解器均失效，呈现的问题包括铰链关节无法承受较大质量的小球引发的关节脱落问题，以及组成长铰链的盒子出现严重的非理性振动现象。通过观察约束误差曲线，可以看到未使用子步骤策略的求解器在高质量比的关节连接场景下无法有效降低约束误差，难以满足关节连接的约束条件。相比之下，采用子步骤策略的求解器在约束误差上表现出逐渐振荡并收敛的趋势，虽然在视觉上关节处有轻微的脱节现象，但整体表现较好。

通过观察能量曲线，发现使用子步骤策略的求解器在模拟初期呈现出不断增长的系统势能变化，这意味着更准确的约束力能够维持关节约束，避免连接在一端的小球因重力影响而不断下降。而未使

用子步骤策略的求解器失效的原因在于，在高质量比的关节连接中，较大质量的小球会对铰链关节施加较大的力，未使用子步骤策略的求解器无法在整个大时间步内计算出准确的响应，导致铰链关节失效。视觉效果表现为小球因重力影响不断下降，导致关节脱落。此外，高质量比会放大系统中的数值误差，使求解器在处理约束条件时更加不稳定，导致严重的非理性振动。

图 6　锁链和球场景((a) Ball and Chain 场景表现；(b)求解前后系统能量变化量；(c)约束误差变化曲线)

Fig. 6　Chain and ball scene ((a) Ball and chain scene performance; (b) System energy change before and after solving; (c) Constraint error variation curve)

相比之下，子步骤策略通过将一个大时间步分解为多个小时间步，细化了求解过程。这样可以更精确地捕捉系统的动态行为，减少数值误差。通过多个小时间步的迭代计算，约束误差逐步减少，使系统趋于稳定，从而更好地满足关节连接的约束条件。

3) Overlap。为了进一步探究基于 Baumgarte 稳定化策略和软约束策略的求解器在约束力响应上的差异，本文构建了一个包含 10 个盒子的场景，并且这些盒子排成一列并且已经发生穿透。离散碰撞检测是一种在物理仿真和计算机图形学中常用的技术，用于检测模拟中的物体是否在离散时刻发生了碰撞。在基于离散碰撞检测的物理引擎中，随着时间步长的增大，物体发生穿透的情况变得不可避免。通过这个场景，可以更好地比较和分析 2 种求解器在处理穿透和约束力响应方面的表现差异。图 7 展示了不同求解器在该场景下的稳定性相关的对比结果。

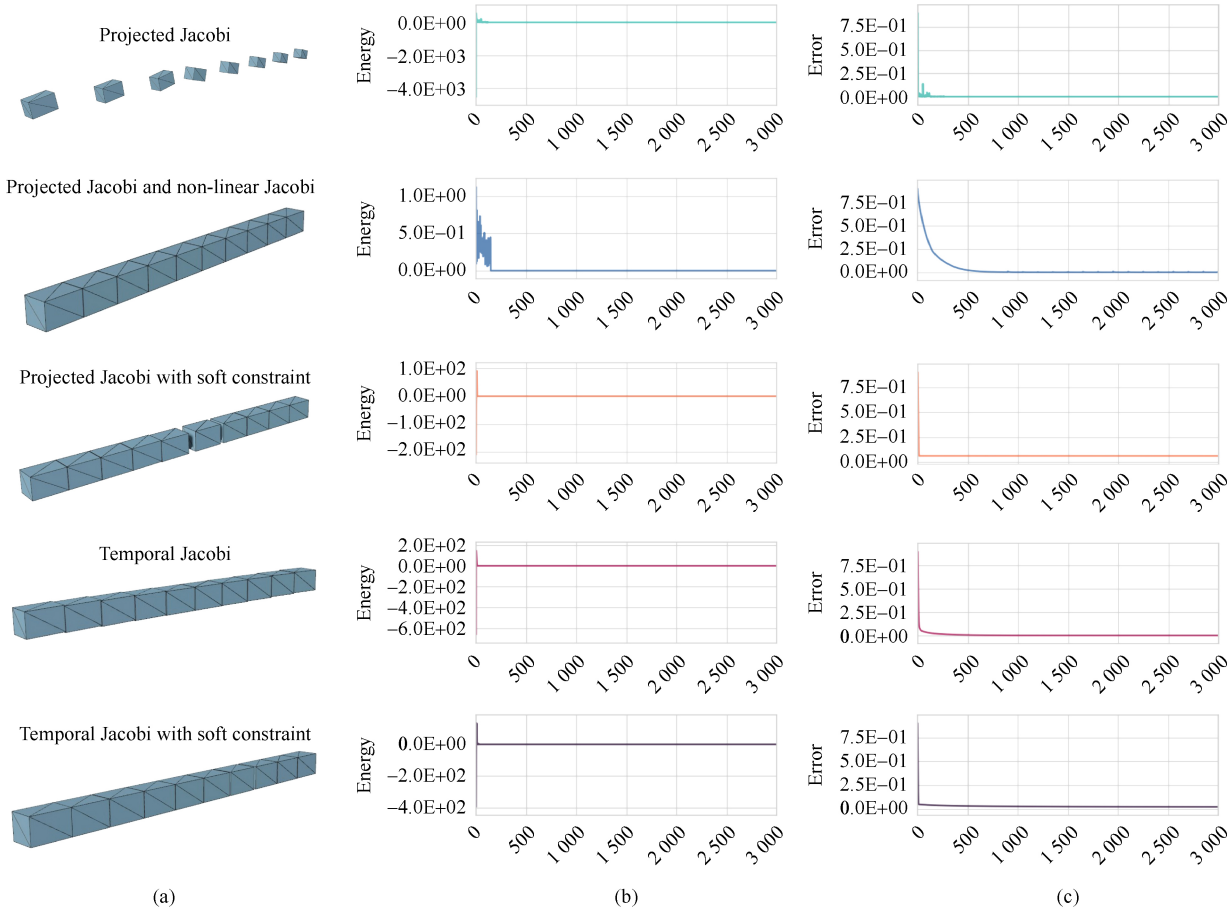

图 7 穿透场景((a) Overlap 场景表现；(b)求解前后系统能量变化量；(c)约束误差变化曲线)

Fig. 7 Overlap scene ((a) Overlap scene performance; (b) System energy change before and after solving; (c) Constraint error variation curve)

在该场景中，使用基本的 Baumgarte stabilization 策略的求解器(projected Jacobi，PJ)在施加分离力时表现出显著的刚性，容易出现过冲现象。而采用软约束(Soft constraint)方法添加弹簧阻尼系统则可以有效缓解这一问题，实现更为平滑和精确的物体分离。相比之下，PJNJ 求解器采用非线性迭代方法直接修正位置误差，尽管在单帧内实现了物体的分离，但由于修正过于迅速和直接，产生了视觉上的不自然感。除此之外，在较大的时间步长下，PJNJ 求解器出现了明显的抖动行为，进一步影响了模拟的稳定性和视觉效果。此外，引入子步骤策略的求解器通过减少每个时间步长的间隔来提高求解精度，从而在一定程度上缓解了 Baumgarte stabilization 方法带来的过冲问题。

4.3 性能测试

对于 4.2 节展示的各个场景，本文统计了不同求解器在进行相同迭代次数(300)下的求解效率，见表 2。

表 2 时间开销对比/ms

Table 2 Time consumption comparison/ms

算法	场景		
	Box stack	Ball and chain	Overlap
PJ	7.093 2	7.212 8	6.804 4
PJNJ	54.825 1	53.645 8	45.276 9
PJSoft	7.652 4	7.376 4	6.419 0
TJ	10.338 9	9.641 3	8.920 8
TJSoft	10.503 3	9.326 5	7.833 7

在 5 种求解器中，基本的迭代过程均采用投影雅各比迭代法来求解约束方程组。图 8 展示了 Overlap 场景第 1 帧的迭代残差随迭代次数的变化。其中不使用子步骤法的曲线收敛速率基本相近，这是由于在求解线性方程组的方法上均使用雅各比迭代法。而子步骤法实际上将 300 次迭代平均分到了每一个子时间步长里，且在子时间步长里实际求解的是不同的线性方程组，这也是 TJSoft 曲线每相隔一定迭代次数出现残差骤然增长的变化的原因。图 9 展示了不同求解器在 2 个关键步骤：构建约束方程以及单步迭代求解的平均时间开销。可以看出，求解器的时间开销的主要差异在于速度约束中雅各比矩阵的更新频率。PJ 和 PJSoft 求解器的求解速率相近，这是因为其在约束修正项构建上存在差异，但算法流程基本一致。PJNJ 在 3 个测试场景中求解速率最慢，这是由于在非线性雅各比迭代过程中，每次迭代后都需要更新雅各比矩阵，导致计算开销显著增加。相比之下，使用子步骤法的 TJ 和 TJSoft 求解器在每次子步骤中更新雅各比矩阵，虽然引入了一定的计算开销，但在一定程度上平衡了计算精度与时间复杂度之间的关系。

4.4　复杂性场景展示

本文构建了 2 个复杂性场景：①包含多种关节(铰链关节和固定关节)的吉普车过吊桥的仿真场景；②同样包含了多种关节(铰链关节和固定关节)并且包含大质量比碰撞的风车转动与堆积块碰撞的场景。时间步长均设置为 16 ms。图 10~11 展示了不同求解器在这 2 个场景中的表现。在图 10 场景中采用非线性雅各比迭代法的求解器难以维持稳定的关节约束，而采用子步骤策略与软约束的求解器均在一定程度上提高了关节约束的稳定性。在图 11 场景中采用非线性雅各比迭代法的求解器的刚体方块还是产生了不稳定的抖动行为，而采用软约束和子步骤法的求解器相比较基础的 Baumgarte 稳定策略的求解器在碰撞力大小的控制上更加稳定，一定程度上抑制了过冲现象。

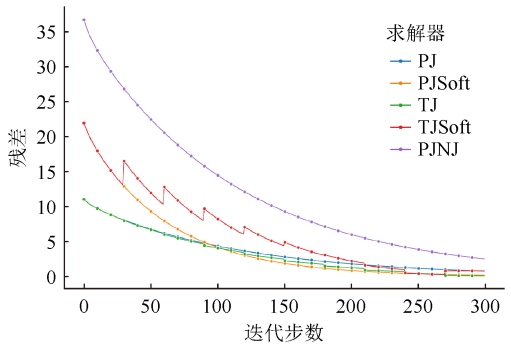

图 8　单帧迭代残差变化曲线

Fig. 8　Single-frame iterative residual change curve

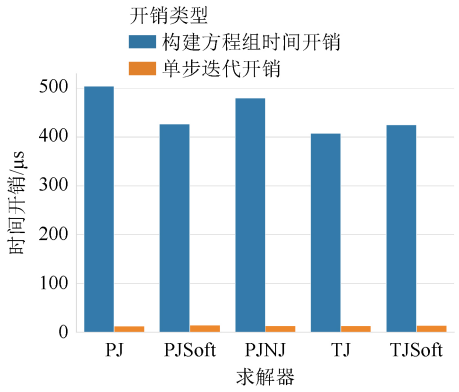

图 9　关键步骤时间开销

Fig. 9　Time consumption of key step

图 10　不同求解器在复杂场景——车过吊桥的表现

Fig. 10　Performance of different solvers in complex scenarios—cars crossing suspension bridges ((a) PJ; (b) PJNJ; (c) PJSoft; (d) TJ; (e) TJSoft)

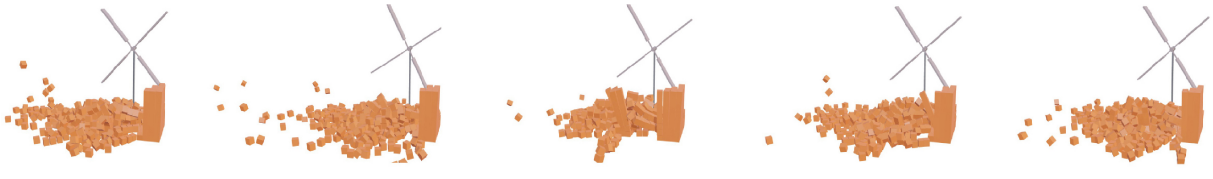

图 11　不同求解器在复杂场景——风车转动撞击堆积块的表现

Fig. 11　Performance of different solvers in complex scenarios—windmill rotation impacting stacked blocks ((a) PJ; (b) PJNJ; (c) PJSoft; (d) TJ; (e) TJSoft)

5 结束语

本文基于 GPU 实现了 5 种基于投影雅各比迭代法的约束求解器，分析了不同求解器在理论求解策略上的差异与特点，并通过多个实际场景展示了这些差异。总结而言，对于性能关键的场景，使用软约束法替代基础的 Baumgarte stabilization 法往往是更好的选择；而在精度要求更高或更复杂的大规模场景中，引入子步骤策略可以为求解器提供更优的表现，为实时物理模拟和交互式计算机图形学中的多体系统约束求解提供有价值的参考。

然而，本方法仍存在一些不足和缺陷。首先，尽管 GPU 加速提高了求解器的性能，但在处理极端复杂或大规模场景时，内存管理和数据传输效率仍然是瓶颈，可能导致性能下降。其次，当前的求解器在处理关节等约束时，其稳定性和精度仍需进一步提升。

对于未来的工作，将探索更多适用于 GPU 上的约束求解器，进一步利用 GPU 加速来探索实时模拟大规模刚体场景的可能性。

参考文献 (References)

[1] PARBERRY I. Introduction to game physics with Box2D[M]. Boca Raton: CRC Press, 2017: 51-78.

[2] BARAFF D. Coping with friction for non-penetrating rigid body simulation[J]. ACM SIGGRAPH Computer Graphics, 1991, 25(4): 31-41.

[3] TASORA A, NEGRUT D, ANITESCU M. A GPU-based implementation of a cone convex complementarity approach for simulating rigid body dynamics with frictional contact[C]//ASME 2008 International Mechanical Engineering Congress and Exposition. New York: ASME, 2008: 107-118.

[4] STEWART D E, TRINKLE J C. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction[J]. International Journal for Numerical Methods in Engineering, 1996, 39(15): 2673-2691.

[5] ANITESCU M, POTRA F A. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems[J]. Nonlinear Dynamics, 1997, 14(3): 231-247.

[6] CLINE M B, PAI D K. Post-stabilization for rigid body simulation with contact and constraints[C]//2003 IEEE International Conference on Robotics and Automation. New York: IEEE Press, 2003: 3744-3751.

[7] CATTO E. Iterative dynamics with temporal coherence[EB/OL]. [2024-06-23]. https://box2d.org/files/ErinCatto_IterativeDynamics_GDC2005.pdf.

[8] ERLEBEN K. Stable, robust, and versatile multibody dynamics animation[D]. Copenhagen: University of Copenhagen, 2004.

[9] MURTY K G, YU V F. Linear complementarity, linear and nonlinear programming[M]. Berlin: Heldermann, 1988: 1-11.

[10] HARADA T. Parallelizing the physics pipeline: physics simulations on the GPU[EB/OL]. [2024-06-23]. https://ubm-twvide001.s3.amazonaws.com/01/vault/gdc10/slides/Harada_Takahiro_PhysicsForProgvammers_GPUPhysics.pdf.

[11] TONGE R, WYATT B, NICHOLSON N. Physx GPU rigid bodies in batman: Arkham asylum[J]. Game Programming Gems, 2010, 8: 590-601.

[12] CATTO E. Modeling and solving constraints[EB/OL]. [2024-06-23]. https://box2d.org/files/ErinCatto_ModelingAndSolvingConstraints_GDC2009.pdf.

[13] COTTLE R W, PANG J S, STONE R E. The linear complementarity problem[M]. Philadelphia: Society for Industrial and Applied Mathematics, 2009: 1-10.

[14] COUMANS E. Bullet physics simulation[C]//ACM SIGGRAPH 2015 Courses. New York: ACM, 2015: 7.

[15] HARADA T. A parallel constraint solver for a rigid body simulation[C]//SIGGRAPH Asia 2011 Sketches. New York: ACM, 2011: 22.

[16] TONGE R, BENEVOLENSKI F, VOROSHILOV A. Mass splitting for jitter-free parallel rigid body simulation[J]. ACM Transactions on Graphics, 2012, 31(4): 105.

[17] MACKLIN M, ERLEBEN K, MÜLLER M, et al. Non-smooth newton methods for deformable multi-body dynamics[J]. ACM Transactions on Graphics, 2019, 38(5): 140.

[18] CROMER A. Stable solutions using the Euler approximation[J]. American Journal of Physics, 1981, 49(5): 455-459.

[19] CHAPPUIS D. Constraints derivation for rigid body simulation in 3D[EB/OL]. (2013-11-13). [2024-06-23]. https://danielchappuis.ch/download/ConstraintsDerivationRigidBody3D.pdf.

[20] BAUMGARTE J. Stabilization of constraints and integrals of motion in dynamical systems[J]. Computer Methods in Applied Mechanics and Engineering, 1972, 1(1): 1-16.

[21] CATTO E. Soft constraints: reinventing the spring[EB/OL]. [2024-06-23]. https://box2d.org/files/ErinCatto_SoftConstraints_GDC2011.pdf.

[22] MÜLLER M, MACKLIN M, CHENTANEZ N, et al. Detailed rigid body simulation with extended position based dynamics[J]. Computer Graphics Forum, 2020, 39(8): 101-112.

[23] MACKLIN M, STOREY K, LU M, et al. Small steps in physics simulation[C]//The 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York: ACM, 2019: 2.