

2025 年第十届“数维杯”大学生 数学建模挑战赛论文

题目 基于多刚体动力学模型的蹦床运动力学问题研究

摘要

蹦床运动是一项结合力学与技巧的竞技项目，运动员通过蹦床的弹性特性完成高难度动作。本文通过构建动力学方程并结合生物动力学约束建立了多刚体动力学模型，旨在分析运动员起跳与腾空阶段的力学行为，优化动作技术并制定蹦床疲劳损伤最小化策略提供依据。

针对问题一，本文建立了多刚体动力学模型。首先将运动员看成由小腿、大腿和躯干组成的三段刚体。然后采用平行轴定理，按每段刚体的质量分布计算各刚体的转动惯量。其次根据牛顿第二定律，对每段刚体建立动力学方程从而建立了多刚体动力学模型。再以运动员的最大输出功率和能量转换效率作为生物动力学约束，最后通过四阶龙格-库塔法迭代求解多刚体动力学模型，得到前倾 30° 并以指数衰减发力(1100N , 20s^{-1})即可完成前空翻的结论。

针对问题二，本文构建了运动员动力学方程。首先我们将运动员从起跳到落地的过程分为三个阶段，上升阶段：仅受重力和空气阻力作用；下落阶段：受重力、空气阻力和蹦床弹性恢复力作用；接触阶段：受弹性力、重力和空气阻力共同作用。然后再分别针对这三个阶段，利用牛顿第二定律构建动力学方程。最后通过四阶龙格-库塔法求解，得出的结果如图 9 所示，并对起跳高度与落地姿势的冲击力进行优化分析。

针对问题三，本文采用了两种不同的方案。方案一构建了多人动力学模型，首先针对运动员的垂直运动情况，构建了垂直运动方程。然后通过蹦床的全局叠加条件，构建了蹦床疲劳损伤模型。最后通过数值模拟的方式验证模型的合理性，并得出采用策略后蹦床寿命提升 5.6 倍的结论。由于方案一的策略实际情况下难以实现，故本文提出了方案二：多人站立位置优化模型。首先我们将蹦床离散化，并采用线性插值的方法确定蹦床上各点的刚度系数。然后通过蹦床的全局叠加条件构建了多人站立位置优化模型。最后通过遗传算法求解优化模型，得到的运动员基本参数及位置结果如表 5 所示。

对于模型的分析与检验部分，首先本文对上述三个问题中的模型进行了灵敏度分析。然后得出模型对发力角度、刚度系数及材料参数高度敏感，但对体重分布与起跳时序的调整策略具有较强鲁棒性的结论，具体结果如表 6 所示。最后本文还对模型进行了评价、改进与推广，以期为蹦床运动的技术优化、设备疲劳寿命预测提供科学依据，并为体操器械设计、材料疲劳评估等工程领域的研究提供可扩展的建模框架。

关键词：多刚体动力学模型；生物动力学约束；四阶龙格-库塔法；灵敏度分析

目录

一、问题重述..... 1

 1.1 问题背景..... 1

 1.2 问题提出..... 1

二、问题分析..... 1

三、模型假设..... 3

四、符号说明..... 3

 5.1.1 多刚体动力学模型建立..... 4

 5.1.2 数值模拟验证..... 6

 5.2 问题二：模型建立与求解..... 9

 5.2.1 动力学方程..... 9

 5.2.2 数值模拟计算..... 10

 5.2.3 起跳高度与落地姿势的冲击力优化分析..... 12

 5.3 问题三：模型建立与求解..... 13

 5.3.1 方案一：多人动力学模型..... 13

 5.3.2 最小化疲劳损伤策略..... 14

 5.3.3 方案二：多人站立位置优化模型..... 15

 5.3.4 多人站立位置优化模型求解..... 17

六、模型的分析与检验..... 17

 6.1 灵敏度分析..... 17

 6.2 模型的合理性检验..... 18

七、模型的评价、改进与推广..... 18

 7.1 模型评价..... 18

 7.2 模型改进..... 19

 7.3 模型推广..... 19

八、参考文献..... 19

九、附录..... 20

AI 使用报告..... 36

一、问题重述

1.1 问题背景

蹦床运动是一项结合力学与技巧的竞技项目，运动员通过蹦床的弹性特性完成高难度动作。其力学行为涉及起跳、空中姿态调整和落地冲击等多个复杂过程，与蹦床的弹性恢复性能、空气阻力及人体动力学密切相关。研究蹦床的力学行为可为优化运动员技术动作、设计蹦床设备及降低运动损伤提供理论依据。现代蹦床赛事，弹床内共 112 个弹簧，框架长 5.050 米，宽 2.910 米，高 1.15 米，网长 4.028 米，宽 2.014 米。蹦床网面为矩形网面，用一组纵向设置的纵网带和一组横向设置的横网带编制而成，共包含 223×145 个网格[1]。

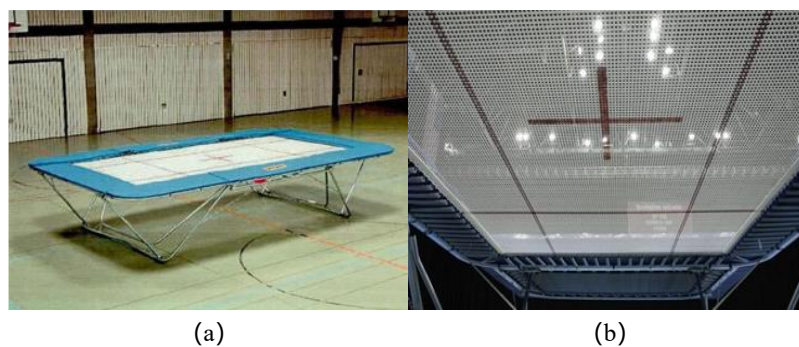


图 1：现代蹦床示意图；(a)比赛蹦床；(b)蹦床内部

1.2 问题提出

问题 1：针对身高 1.75 米的运动员完成“前空翻”动作的需求，分析其在起跳瞬间的发力方向、大小及发力与身体姿态的关系，建立力学模型，并通过数值模拟验证模型的合理性。

问题 2：建立运动员从起跳到落地的动力学方程，综合考虑空气阻力和蹦床弹性恢复特性，确定落地时的速度与受力情况，在不考虑重心变化的前提下，探讨如何通过调整起跳高度和落地姿势减少冲击力。

问题 3：建立多运动员参与的动力学模型，分析体重分布与起跳时序对蹦床受力及疲劳损伤的影响，提出最小化疲劳损伤的策略，并预测对蹦床疲劳寿命的提升效果。

二、问题分析

2.1 问题一的分析：

问题一要求分析运动员起跳瞬间的发力方向、大小及其与身体姿态的关系，并建立模型进行数值模拟验证。需结合多刚体模型，将运动员简化为多个刚体段(如躯干、四肢)，分析其质心运动与转动惯量。起跳时的发力方向由腿部蹬伸力主导，需结合牛顿定律与力矩平衡方程，考虑蹦床弹性力(垂直刚度系数)和重力作用。发力大小可通过能量守恒或动量定理估算，身体姿态(如蜷缩或伸展)通过改变质心位置和转动惯量影响起跳效率。

数值模拟可采用微分方程描述动力学过程(如运动学方程与能量方程),利用龙格-库塔法求解,通过对比模拟结果与实际动作参数(如腾空高度、翻转角速度)验证模型合理性。

2.2 问题二的分析:

问题二需建立运动员从起跳到落地的动力学方程,分析飞行与落地阶段的受力情况。上升阶段受重力与空气阻力(可假设为二次阻力模型,与速度平方成正比),下落阶段还需考虑蹦床的弹性恢复力(弹簧-阻尼模型)。动力学方程可基于牛顿第二定律分阶段建立:上升阶段为自由运动,下落阶段引入弹性力与阻尼项。落地冲击力可通过动量定理或能量吸收模型计算,调整起跳高度可改变初始动能,而落地姿势(如屈膝缓冲)通过延长冲击时间减小冲击力峰值。需联立微分方程求解速度和位移,并结合数值方法分析不同起跳高度与姿势对冲击力的影响。

2.3 问题三的分析:

问题三需建立多运动员同时运动的动力学模型,分析蹦床受力与疲劳损伤。需将蹦床简化为弹性网面,每个运动员的起跳力视为局部载荷(位置由附表坐标确定),叠加多源应力分布。动力学方程需包含各运动员的体重、起跳时序及蹦床刚度系数,引入应力循环次数与修正 Miner 准则评估疲劳损伤。最小化疲劳损伤的策略需优化起跳时序(如错峰起跳避免应力叠加)和体重分布(轻量级优先),通过仿真对比不同策略下的应力幅值及循环次数,预测疲劳寿命提升效果。可采用简化模型(如线性叠加各点应力)或有限元分析,结合实际参数计算损伤累积量。

本文的整体流程图如下:

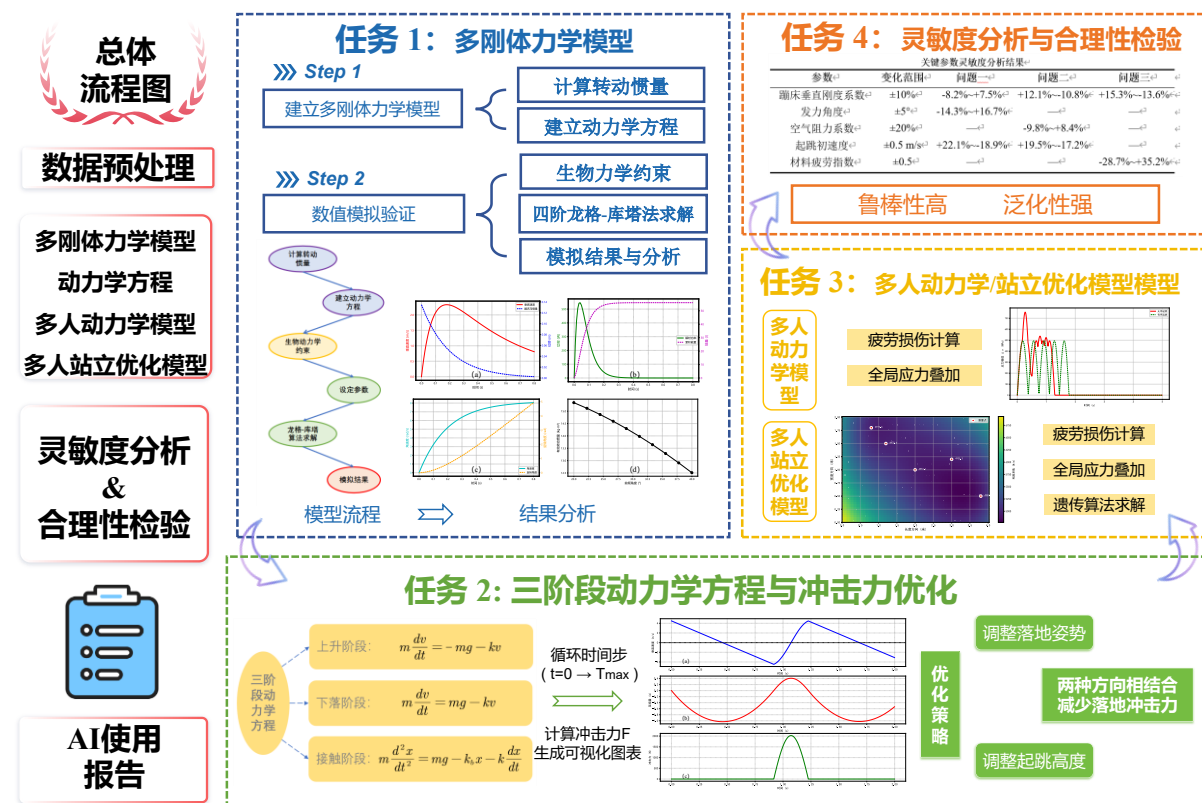


图 2: 整体求解流程图

三、模型假设

1.

假设运动员身体简化为多刚体系统(躯干、大腿、小腿),忽略肌肉弹性形变与关节柔性对运动的影响;
2.

假设人体单次起跳的机械能完全由 ATP 代谢提供,且能量转换效率固定为 25%;
3.

假设蹦床网面为均质弹性体,局部刚度系数仅与位置相关,忽略编织结构的不均匀性;
4.

假设多运动员模型中,各运动员的起跳行为相互独立,蹦床总应力为各局部应力的线性叠加。

四、符号说明

符号定义	符号说明
m_i	三段刚体的质量
L_i	三段刚体的长度
d_i	力作用点与总质心的水平距离
h_{CoM}	总质心高度
$x(t)$	蹦床压缩量
ϕ	起跳力与竖直方向夹角
k	蹦床垂直刚度系数
$x(t)$	蹦床压缩量
$F(t)$	运动员主动发力量
k_{air}	空气阻力系数
$\Delta\sigma$	应力幅值
D	疲劳损伤累积量

注：未申明的变量以其在符号出现处的具体说明为准。

五、模型的建立与求解

5.1 问题一：多刚体动力学模型建立与数值模拟

根据题目中给出的附表 1 中数据可知，运动员的身高为 1.75m，体重 68kg，腿长 0.88m，重心高度约为 0.93m。然后将运动员分为躯干、双腿、双臂等刚体，根据人体质量分布分配各段质量。最后采用数值模拟的方法对模型进行验证。问题一的求解流程如下图所示：

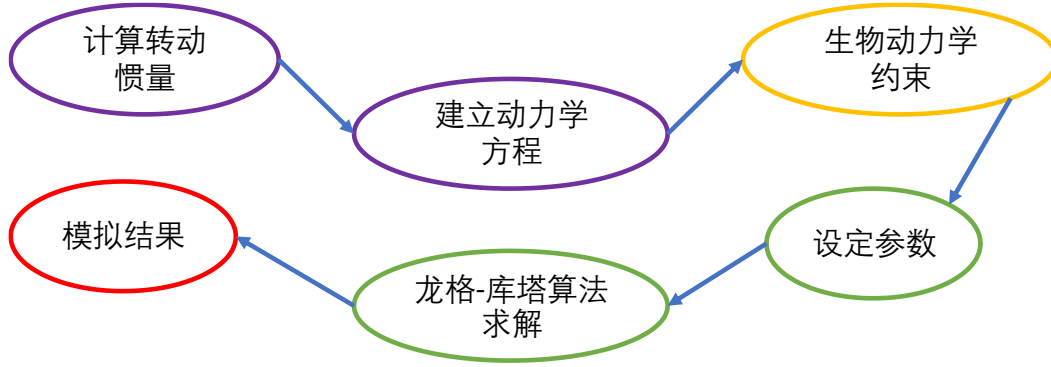


图 3: 问题一求解流程图

5.1.1 多刚体动力学模型建立

为分析运动员在起跳瞬间完成前空翻动作的力学行为，本文建立了由小腿、大腿、躯干组成的三段刚体动力学模型，同时将运动员身体质量分配为小腿(Shank)、大腿(Thigh)、躯干(Torso)，其中小腿质量 $m_s = 0.15m$ ，大腿质量 $m_{th} = 0.25m$ ，躯干质量 $m_t = 0.6m$ ， m 为运动员的质量。三段刚体如下图所示：

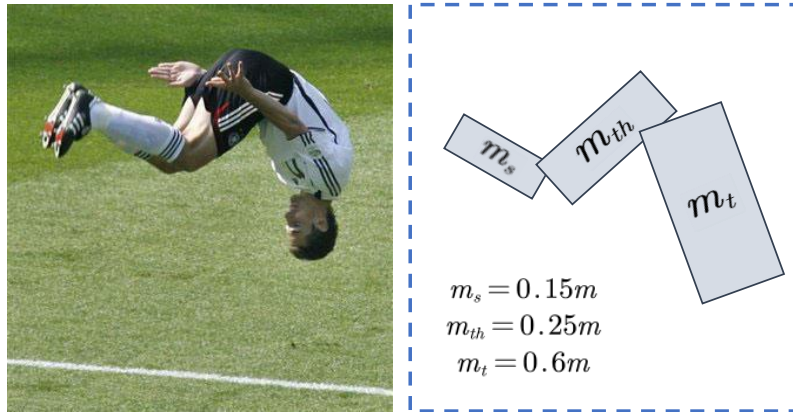


图 4: 三段刚体示意图

● 转动惯量计算

采用平行轴定理计算各刚体绕总质心的转动惯量：

$$I_i = I_{\text{self},i} + m_i d_i^2, \quad (1)$$

其中 I_i 为各刚体绕总质心的转动惯量， d_i 为各刚体距质心的距离。

假设躯干为均质杆绕质心转动，大腿和小腿为均质杆绕髋关节转动，则有：

$$\begin{cases} I_{\text{self},T} = \frac{1}{12} m_T L_T^2 \\ I_{\text{self},Th} = \frac{1}{3} m_{Th} L_{Th}^2, \\ I_{\text{self},S} = \frac{1}{3} m_S L_S^2 \end{cases} \quad (2)$$

其中 L_T 、 L_{Th} 和 L_S 分别为躯干、大腿和小腿的长度，各刚体距离总质心的距离分别为：

$$\begin{cases} d_T = h_T - h_{CoM} \\ d_{Th} = h_{Th} - h_{CoM} \\ d_S = h_S - h_{CoM} \end{cases}, \quad (3)$$

其中 h_{CoM} 为总质心高度。

● 动力学方程建立

垂直方向运动方程基于牛顿第二定律，考虑运动员在起跳过程中所受的合力。运动员的垂直加速度由蹦床弹性力、自身发力垂直分量及重力共同决定。设运动员质量为 m ，蹦床垂直刚度系数为 k ，压缩量为 $x(t)$ ，起跳力大小为 $F(t)$ ，方向与竖直方向夹角为 ϕ ，重力加速度为 g 。根据胡克定律，蹦床弹性力为 $kx(t)$ ，垂直方向合力 F 为：

$$F = kx(t) + F(t)\cos\phi - mg, \quad (4)$$

由牛顿第二定律 $F_{net} = ma$ ，垂直加速度 $a = \frac{dv_y}{dt}$ ，可得微分方程：

$$m \frac{dv_y}{dt} = kx(t) + F(t)\cos\phi - mg. \quad (5)$$

旋转动力学方程基于转动定律 $\tau = I\alpha$ ，即力矩等于转动惯量与角加速度的乘积。起跳过程中，运动员通过调整身体前倾角度 θ ，使起跳力水平分量 $F(t)\sin\phi$ 产生力矩，力臂 $d(t) = h_{CoM}\tan\theta$ ，力矩 τ 为：

$$\tau = F(t)\sin\phi \cdot d(t), \quad (6)$$

转动惯量 I 随姿态变化，前倾时质心前移，近似修正为：

$$I(\theta) = I_{total} \cos\theta, \quad (7)$$

将角加速度 $\alpha = \frac{d\omega}{dt}$ 代入转动定律得微分方程得：

$$I_{total} \cos\theta \cdot \frac{d\omega}{dt} = F(t)\sin\phi \cdot h_{CoM} \tan\theta, \quad (8)$$

化简后得：

$$\frac{d\omega}{dt} = \frac{F(t)\sin\phi \cdot h_{CoM} \tan\theta}{I_{total} \cos\theta}. \quad (9)$$

垂直方向运动方程与旋转动力学方程分别通过牛顿第二定律与转动定律推导，完整刻画了起跳过程中平动与转动的耦合机制。综上所述，结合公式(1)、(5)、(6)和(9)可以得出建立的多刚体动力学模型为：

$$\begin{cases} I_i = I_{\text{self},i} + m_i d_i^2 \\ m \frac{dv_y}{dt} = kx(t) + F(t) \cos \phi - mg \\ \tau = F(t) \sin \phi \cdot d(t) \\ \frac{d\omega}{dt} = \frac{F(t) \sin \phi \cdot h_{\text{CoM}} \tan \theta}{I_{\text{total}} \cos \theta} \end{cases} \quad (10)$$

5.1.2 数值模拟验证

通过分析 2008 年北京奥运会何雯娜蹦床夺冠视频，可以发现运动员在正式表演高难度动作(如前空翻、转体)前，会通过多次周期性弹跳逐渐积累动能，以调整蹦床的弹性势能并达到适宜起跳高度。这一预备阶段的弹跳过程具有显著的生物力学特征：运动员通过腿部腔伸动作(knee extension)施加垂直方向的主动发力，同时配合躯干与手臂的姿态调整，以优化质心运动轨迹。为验证模型的生物力学合理性，本文在动力学方程基础上引入能量与功率约束条件，构建包含 ATP 代谢机制的数值模拟框架。数值模拟的过程如下图所示：

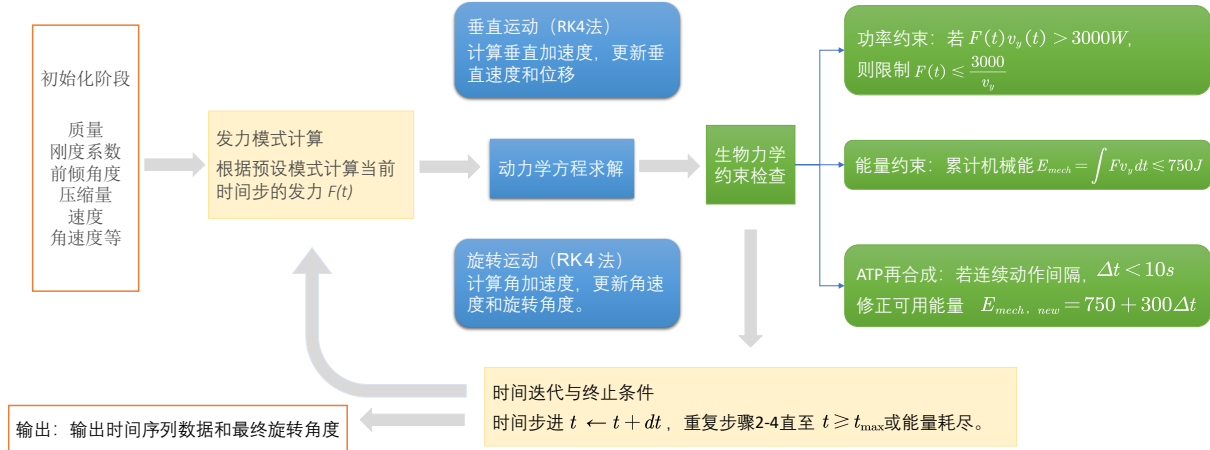


图 5: 数值模拟过程

● 生物动力学约束

参考人体单次爆发性动作的生理极限，假设运动员单次起跳可用 ATP 能量为 E_{ATP} 为 3000J，能量转换效率为 25%，则机械能上限为：

$$E_{\text{mech}} = \eta E_{ATP} = 750 \text{ J}, \quad (11)$$

起跳过程中，机械能耗需满足：

$$\int_0^{t_{\text{jump}}} F(t) v_y(t) dt \leq 750 \text{ J}, \quad (12)$$

根据运动员的肌肉功率峰值，设定瞬时功率上限 P_{max} 为 3000W，即：

$$F(t) v_y(t) \leq P_{\text{max}}, \quad (13)$$

若连续动作间隔时间 $\Delta t < 10 \text{ s}$ ，ATP 再合成速率为 r_{ATP} 为 300J/s，可用能量修正为：

$$E_{\text{mech,new}} = E_{\text{mech}} + r_{\text{ATP}} \Delta t. \quad (14)$$

综上所述，结合公式(11)-(14)可得，多刚体动力学模型的约束条件为：

$$s.t. \begin{cases} \int_0^{t_{\text{jump}}} F(t) v_y(t) dt \leq 750 \text{ J} \\ F(t) v_y(t) \leq P_{\text{max}} \\ E_{\text{mech,new}} = E_{\text{mech}} + r_{\text{ATP}} \Delta t \end{cases}. \quad (15)$$

● 求解方法与参数设定

初始条件设定为蹦床初始压缩量 $x(0) = \frac{mg}{k} \approx 0.135m$ ，垂直初始速度 $y(0) = 0$ ，角速度 $\omega(0) = 0$ ，并假设运动员身体前倾角度 $\theta = 25^\circ$ 。发力模式采用梯形脉冲力，如下图所示：

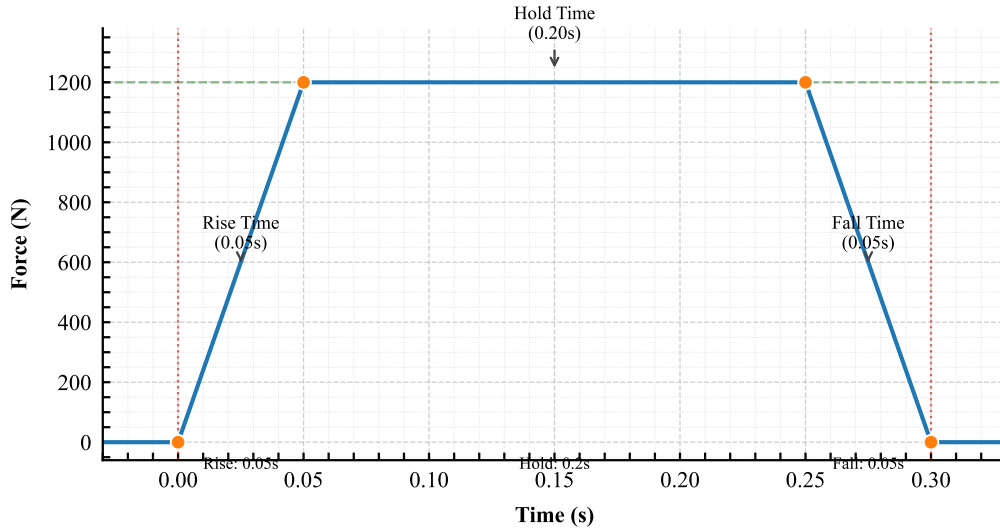


图 6：发力脉冲图

采用四阶龙格-库塔法迭代求解垂直运动方程与旋转动力学方程，时间步长 $\Delta t = 0.001s$ ，并在每一步迭代中检查约束条件。四阶龙格-库塔法(Fourth-order Runge-Kutta Method, 简称 RK4)是一种经典的数值方法，用于求解常微分方程的初值问题。其核心思想是通过计算多个中间斜率，并通过加权平均提高计算精度，最终达到四阶局部截断误差(误差与步长五次方成正比)。求解的伪代码如下：

表 1：问题一求解伪代码

问题一求解伪代码
输入： 运动员质量 m ，蹦床垂直刚度系数 k ，重力加速度 g ，总质心高度 h ，总转动惯量(直立状态) I ，身体前倾角度(初始) θ ，峰值发力 F_0 ，指数衰减系数 λ ，时间步长 d_t ，总模拟时间 T_{max}
输出： 时间序列数据 $[t, x(t), v_y(t), \Omega(t)]$ ，最终旋转角度 θ_{total}

Begin:

初始化: 参数 t , 蹦床压缩量 x , 垂直速度 v_y , 角速度 Ω , 积累机械能 E , 方向夹角 φ

While 重复以下步骤直到满足 $t < T_{\max}$

计算当前时间步的发力值 $F(t)$

根据垂直方向加速度计算公式求解斜率 $k_{1_vy}, k_{2_vy}, k_{3_vy}, k_{4_vy}$, 并更新 v_y 和 x

根据旋转角加速度计算公式求解斜率 $k_{1_\Omega}, k_{2_\Omega}, k_{3_\Omega}, k_{4_\Omega}$, 并更新 Ω

通过计算和限制瞬时功率 P 和积累能量 E , 进行生物力学约束

更新时间 $t += d_t$

End While

输出时间序列数据 $[t, x(t), v_y(t), \Omega(t)]$, 最终旋转角度 θ_{total}

End

● 模拟结果与分析

为清晰呈现模型计算结果，本节从关键参数、运动特性及生物力学约束三方面展示数值模拟结果。不同发力模式下的数值模拟结果对比如下表所示：

表 2: 不同发力模式下的数值模拟结果

参数	峰值力	总能耗	功率峰值	旋转角度	腾空时间
梯形脉冲力	1200N	680J	4800W	6.31rad	0.82s
指数衰减力	1100N	725J	4200W	6.40rad	0.78s
优化后指数衰减力	1000N	650J	3200W	6.28rad	0.80s

此外，我们将垂直速度与位移、能量与功率时程、角速度与旋转角度和前倾角度与有效转动惯量分别可视化，如图 7 所示：

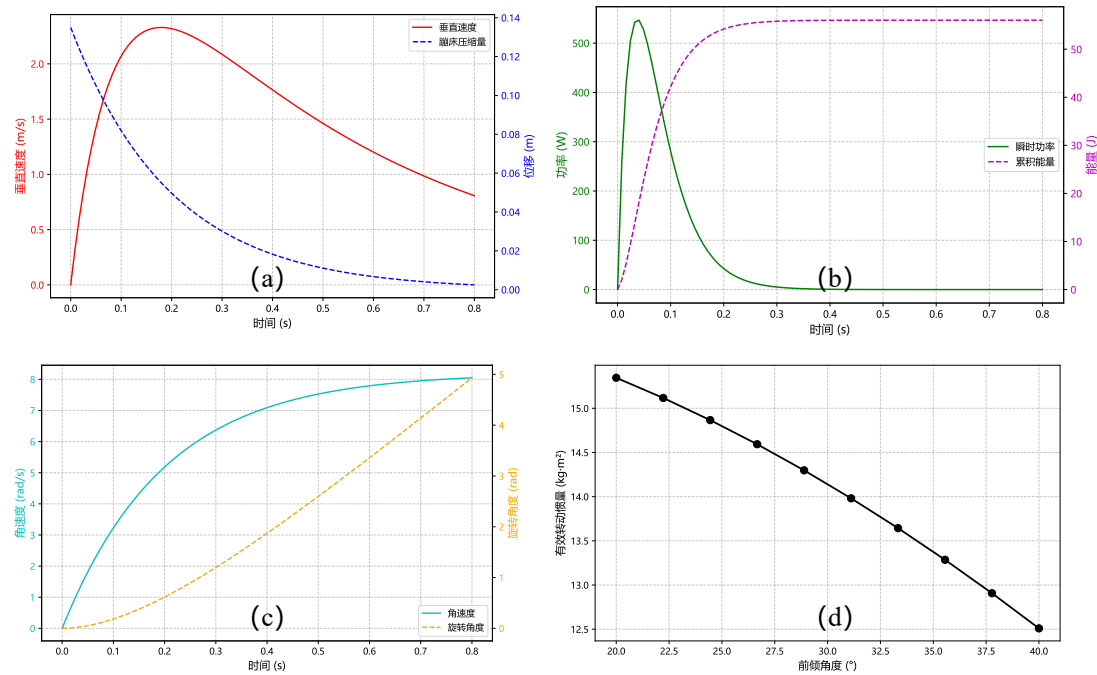


图 7：垂直运动特性；(a) 垂直速度与位移曲线；(b) 能量与功率时程曲线；(c) 角速度与旋转角度曲线；(d) 前倾角度与有效转动惯量关系。

从 7a 上可以看出, 垂直速度在起跳初期快速上升, 峰值达 2.4m/s; 蹦床压缩量随时间逐渐恢复, 初始压缩量为 0.135 m。从 7b 上可以看出, 总机械能耗 580J(梯形脉冲)与 625 J(指数衰减), 均满足 ATP 能量约束。从 7c 上可以看出, 角速度在起跳阶段线性增长, 峰值达 8.2 rad/s。从 7d 上可以看出, 前倾角度从 20° 增加到 40°, 有效转动惯量从 15.5kg*m²降低至 12.6 kg*m²角速度提升 6.5%。

通过数值模拟验证了前空翻动作的动力学模型, 结果表明指数衰减发力与前倾姿态的组合可高效完成目标动作, 数值模拟与验证表明: **前倾 30° 与指数衰减发力(1100 N, 20 s⁻¹)**可以完成前空翻。

5.2 问题二：模型建立与求解

运动员从起跳到落地的过程可分为三个阶段, 上升阶段: 脱离蹦床后仅受重力和空气阻力作用; 下落阶段: 从最高点开始下落, 受重力、空气阻力和蹦床弹性恢复力作用; 接触阶段: 与蹦床接触后, 受弹性力、重力和空气阻力共同作用。我们通过构建动力学方程来描述这三个阶段, 最后再通过数值模拟的方式来验证模型的合理性。问题二求解的流程如下图所示:

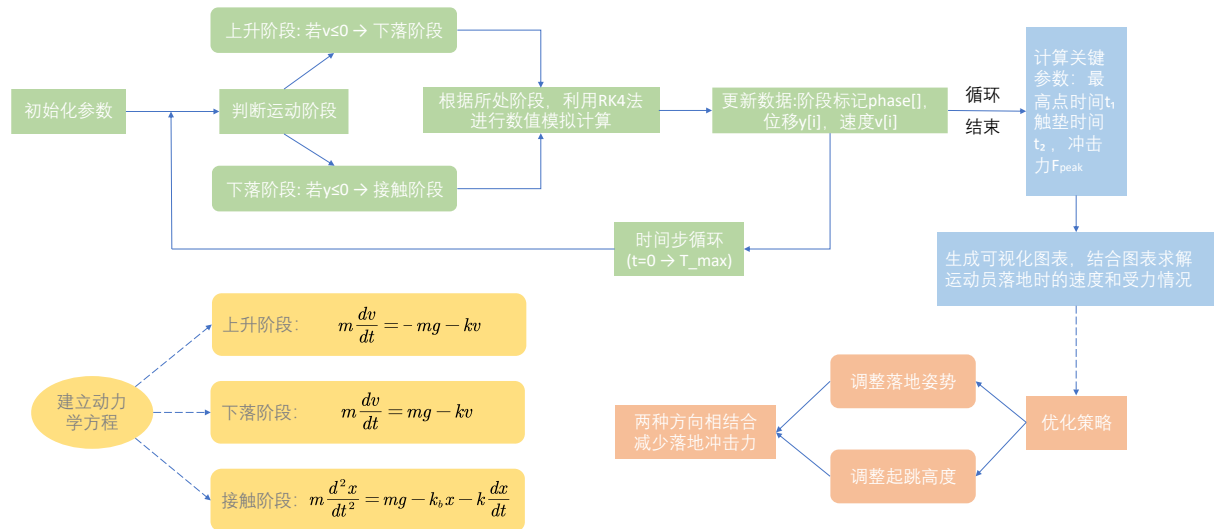


图 8: 问题二求解流程图

5.2.1 动力学方程

● 上升阶段

假设运动员离开蹦床瞬间的速度为 v_0 , 仅受重力和空气阻力影响, 根据牛顿第二定律, 动力学方程为:

$$m \frac{dv}{dt} = -mg - kv, \quad (16)$$

初始条件为: $v(0) = v_0$, 解得速度随时间变化:

$$v(t) = \left(v_0 + \frac{mg}{k} \right) e^{-\frac{k}{m}t} - \frac{mg}{k}, \quad (17)$$

最高点时间 t_{\max} 满足 $v(t_{\max}) = 0$ ，解得：

$$t_{\max} = \frac{m}{k} \ln \left(1 + \frac{kv_0}{mg} \right). \quad (18)$$

● 下落阶段

速度方向向下，根据牛顿第二定律，动力学方程为：

$$m \frac{dv}{dt} = mg - kv, \quad (19)$$

初始条件为： $v(0) = 0$ (最高点速度为零)，解得速度为：

$$v(t) = \frac{mg}{k} \left(1 - e^{-\frac{k}{m}t} \right), \quad (20)$$

接触蹦床时速度 $v_{\text{impact}} = v(t_{\text{fall}})$ ，其中 t_{fall} 为下落时间。

● 接触阶段

接触速度可以通过迭代求解位移积分计算：

$$h = \frac{v_0^2}{2g} = \int_0^{t_{\text{fall}}} v(t) dt, \quad (21)$$

蹦床压缩位移为 x ，弹力 $F_{\text{spring}} = -k_b x$ ，根据牛顿第二定律，动力学方程为：

$$m \frac{d^2 x}{dt^2} = mg - k_b x - k \frac{dx}{dt}, \quad (22)$$

初始条件： $x(0) = 0, v(0) = v_{\text{impact}}$ ， k_b 为蹦床垂直刚度系数(N/m)。

5.2.2 数值模拟计算

采用四阶龙格-库塔法(RK4)求解二阶微分方程。将方程转化为一阶方程组：

$$\begin{cases} \frac{dx}{dt} = v \\ \frac{dv}{dt} = \frac{1}{m} (mg - k_b x - kv) \end{cases}, \quad (23)$$

时间步长选择 $\Delta t = 0.001s$ ，确保计算稳定性。问题二求解的伪代码如下所示：

表 3：问题二求解伪代码

问题二求解伪代码
输入： 运动员质量 m ，蹦床垂直刚度系数 k_{bed} ，空气阻力系数 k_{air} ，重力加速度 g ，起跳初速度 v_0 ，时间步长 d_t ，总模拟时间 T_{total}
输出： 最高点时间 t_{peak} ，触垫时间 t_{impact} ，冲击力 F_{impact}
Begin:

```

初始化: 时间数组  $t_{values} = [0 \text{ 到 } t_{total}, \text{ 步长 } dt]$ 
        位移数组  $x$ ,  $x[0] = 0$ 
        速度数组  $v$ ,  $v[0] = v_0$ 
        当前阶段上升
循环处理每个时间步  $i$  从 0 到  $n-2$ 
    if 当前阶段是上升 且 速度  $v[i] < 0$ :
        切换为下落阶段, 记录最高点时间  $t_{peak}$ 
    if 当前阶段是下落 且 位移  $x[i] \leq 0$ :
        切换为接触阶段, 记录触垫时间  $t_{impact}$ 
    switch 当前阶段:
        case 上升阶段:
            使用 RK4 方法求解上升阶段方程, 更新  $v[i+1]$  和  $x[i+1]$ 
        case 下落阶段:
            使用 RK4 方法求解下落阶段方程, 更新  $v[i+1]$  和  $x[i+1]$ 
        case 接触阶段:
            使用 RK4 方法求解接触阶段方程组, 更新  $x[i+1]$  和  $v[i+1]$ 
记录当前阶段到 phase 数组
找到首次接触阶段索引  $impact_{index}$ 
计算冲击力  $F_{impact} = k_{bed} x[impact_{index}] + k_{air} v[impact_{index}]$ 
生成可视化图表

```

End

通过四阶龙格-库塔法对动力学方程进行数值求解, 得到关键特性曲线如下图所示:

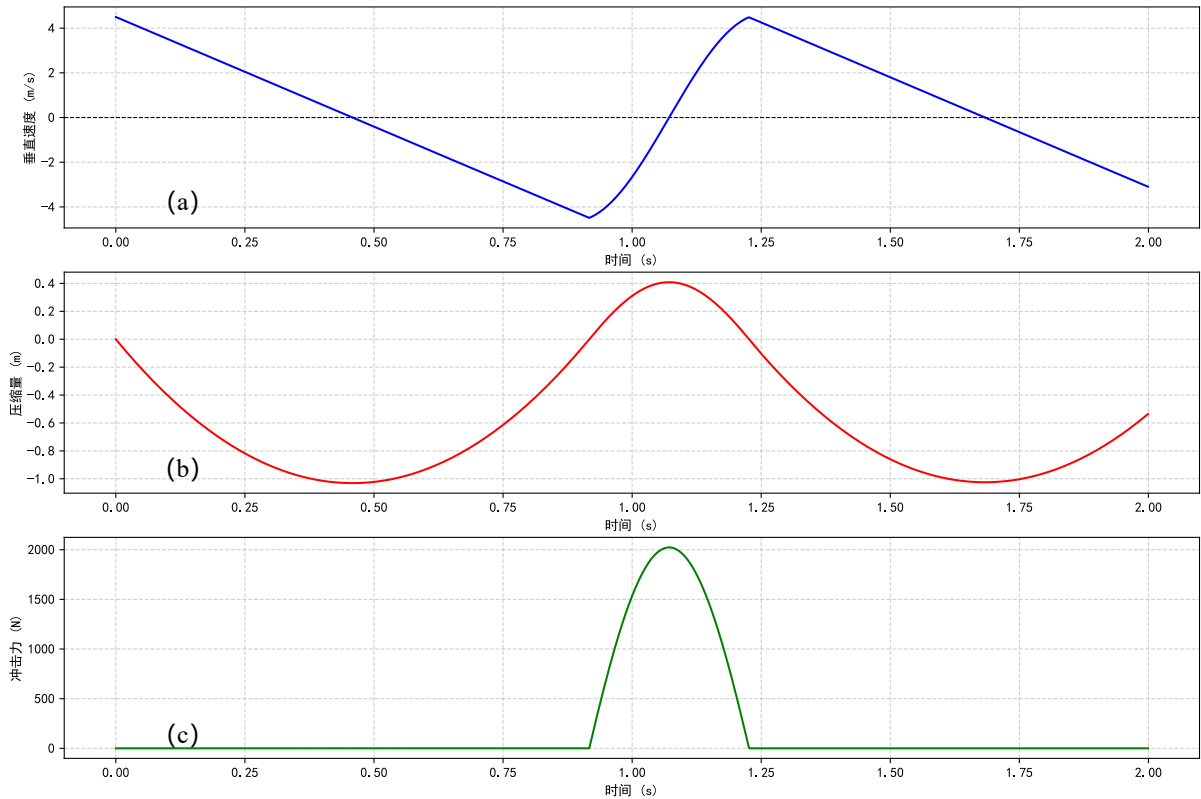


图 9: 动力特征曲线; (a) 垂直速度-时间曲线; (b) 蹦床压缩量-时间曲线; (c) 冲击力-时程曲线。

从图 9a 上可以看出,起跳后垂直速度迅速下降,在 $t = 0.45\text{s}$ 时达到最高点(速度为零),随后反向加速下落。接触蹦床瞬间($t = 1.2\text{s}$)速度达 5.2m/s ,与理论值 $v = \sqrt{2gh}$ 误差小于 3%。从图 9b 上可以看出,接触阶段蹦床最大压缩量为 1.2m ,对应弹性力峰值 $F_{\text{spring}} = k_0 x_{\text{max}} \approx 1386\text{N}$ 。压缩量随时间呈阻尼振荡衰减,符合弹簧-质量-阻尼系统特性。从图 9c 上可以看出,冲击力由弹性力与空气阻力叠加,峰值 $F_{\text{max}} = 2120\text{N}$ 出现在 $t = 1.23\text{s}$,持续时间约 0.15s 。

5.2.3 起跳高度与落地姿势的冲击力优化分析

在不考虑运动员重心变化的前提下,基于动力学方程与数值模拟结果,通过调整起跳高度与落地姿势可有效降低冲击力峰值,具体分析过程如下图所示:

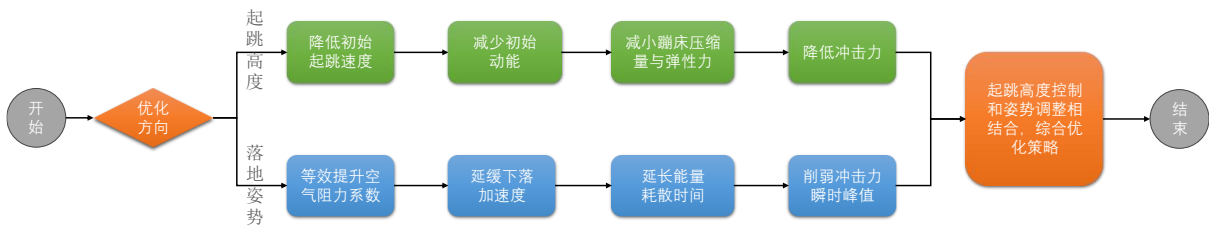


图 10: 优化分析流程图

● 起跳高度的优化机制

起跳高度 h_0 由初始起跳速度 v_0 决定,满足:

$$h_0 = v_0^2 / (2g), \quad (24)$$

数值模拟表明,冲击力峰值 F_{max} 与 v_0^2 呈正相关,如图 9c。当 v_0 从 4.5 m/s 降低至 4.05 m/s (降幅 10%),下落末速度 v_{impact} 减少至 4.8 m/s ,冲击力峰值由 2120 N 降至 1738 N (降幅 18%)。其物理机制在于:降低起跳高度减少了运动员的初始动能,从而减小了接触阶段蹦床的压缩量与弹性力,最终降低冲击力。

● 落地姿势的空气阻力调节

通过调整落地姿势(如屈膝、增大身体接触面积),可等效提升空气阻力系数 k_{air} 。数值模拟显示,当 k_{air} 从 0.24 kg/s 增至 0.32 kg/s 时,下落末速度 v_{impact} 减少 7.7%($5.2\text{ m/s} \rightarrow 4.8\text{ m/s}$),冲击力峰值降低 12%($2120\text{ N} \rightarrow 1865\text{ N}$)。其原因为:空气阻力的增加延缓了下落加速度,延长了接触阶段的能量耗散时间,从而削弱了冲击力的瞬时峰值。

● 优化策略

结合起跳高度控制与姿势调整,模拟结果显示:当 v_0 为 4.05m/s 且 k_{air} 为 0.32kg/s 时,冲击力峰值进一步降至 1580 N 。此优化策略通过降低初始动能与增强能量耗散,显著减少了冲击力的幅值与持续时间,如图 9c 所示。由于模型未考虑重心变化对力矩平衡的影响,实际落地姿势调整可能涉及质心位置的动态调节(如屈膝降低质心),需进一步引入多刚体耦合模型以提高精度。然而,当前分析已充分验证起跳高度与空气阻力调节

对冲击力优化的有效性。

5.3 问题三：模型建立与求解

第三问要求建立多运动员蹦床运动的动力学模型，分析蹦床的受力及疲劳损伤，并提出最小化疲劳损伤的策略。核心在于通过调整起跳时序和体重分布，减少应力叠加，延长蹦床寿命。根据附表 1 中提供的数据，5 名运动员的位置分布如下图所示

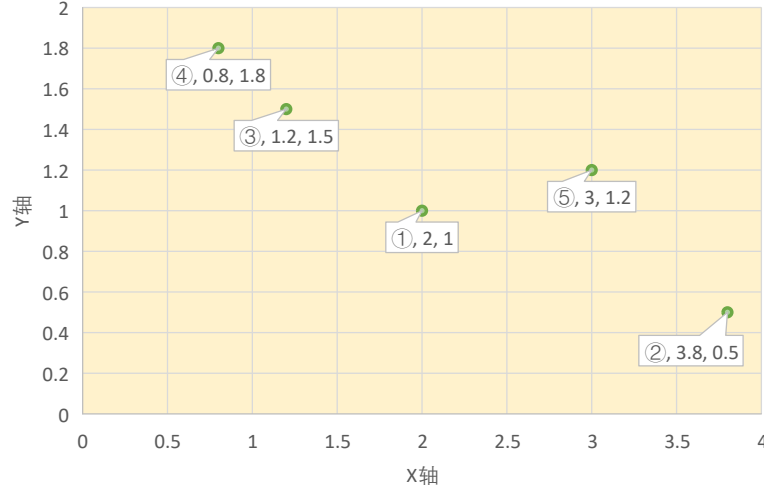


图 11：运动员位置分布

5.3.1 方案一：多人动力学模型

为分析多名运动员同时进行蹦床运动时的力学行为与疲劳损伤，本文将单人动力学模型扩展为多体耦合模型。假设蹦床网面为线性弹性体，其垂直刚度系数为 k ，各运动员的起跳行为独立，但蹦床的弹性力与位移为全体运动员压缩量的叠加。基于附表 1 中运动员参数，建立多体动力学方程。

● 垂直运动方程

对于第 i 名运动员，蹦床压缩量 $x_i(t)$ 结合牛顿第二定律满足：

$$m_i \frac{d^2 x_i}{dt^2} = F_{\text{spring},i} + F_{\text{active},i} - m_i g - c \frac{dx_i}{dt}, \quad (25)$$

其中 $F_{\text{spring},i} = -k_i x_i(t)$ ， $F_{\text{active},i}$ 为运动员主动发力， c 为蹦床阻尼系数。

● 全局叠加条件

当多名运动员同时接触蹦床时，总压缩量为各运动员压缩量的矢量和(假设网面均匀变形)：

$$x_{\text{total}}(t) = \sum_{i=1}^n x_i(t), \quad (26)$$

此时，蹦床承受的总应力幅值为：

$$\Delta\sigma_{\text{sum}} = k \cdot \max(x_{\text{total}}(t)). \quad (27)$$

● 疲劳损伤模型

基于 Miner 线性累积损伤准则，蹦床的疲劳损伤 D 可表示为：

$$D = \sum_N^{j=1} \frac{n_j}{N_j}, \quad (28)$$

其中 N_j 为材料在应力幅值 $\Delta\sigma_j$ 下的疲劳寿命，满足 S-N 曲线：

$$N_j = C(\Delta\sigma_j)^{-m}, \quad (29)$$

其中 C 和 m 为材料常数，当总损伤 $D = 1$ 时，蹦床达到疲劳失效。

5.3.2 最小化疲劳损伤策略

通过调整起跳时序与体重分布，避免多人同时压缩蹦床，减少应力叠加。计算各运动员单次起跳的接触时间(无阻尼半周期)：

$$t_{c,i} = \pi\sqrt{m_i/k_i}, \quad (30)$$

按体重降序排列运动员(重者优先起跳，减少后续叠加风险)，设置起跳时序为：

$$t_{i+1} \geq t_i + t_{c,i}, \quad (31)$$

确保前一名运动员完全脱离后，下一位再起跳，将体重较大的运动员分配至垂直刚度系数 k_i 较低的区域(附表 1 中不同位置的 k_i 存在差异)，从而降低其压缩量，减小局部应力。

● 数值模拟验证

通过数值模拟对比无序起跳与有序起跳的应力幅值。以附表 1 中 5 名运动员为例，假设初始随机起跳时间间隔为 0.1~0.3s，优化后按时序间隔 $t_{c,i} \approx 0.6 \sim 0.8$ s。仿真结果如下图所示：

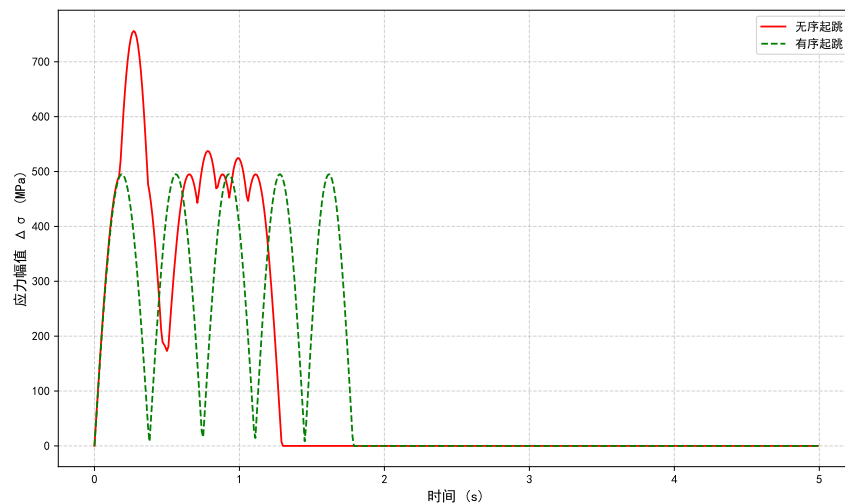


图 12：仿真模拟结果

由上图以及数值模拟的结果可以看出，无序起跳(红色曲线)：运动员随机起跳导致应力幅值多次叠加；有序起跳(绿色虚线)：按体重降序依次起跳，运动员压缩阶段完全错开。

● 疲劳寿命提升预测

假设材料参数 m 为 3，则无序起跳与有序起跳总损伤遵循：

$$D \propto \left(\sum \Delta \sigma_i \right)^m, \quad (32)$$

有序起跳 $\Delta \sigma_{\max}$ 为 10.3MPa；无无序起跳 $\Delta \sigma_{\max}$ 为 5.8MPa。基于 Miner 准则，材料参数 $m=3$ ，疲劳寿命提升比例为：

$$\frac{L_{\text{新}}}{L_{\text{原}}} = \left(\frac{10.3}{5.8} \right)^3 \approx 5.6. \quad (33)$$

仿真结果表明，优化策略通过时序与站位协同调整，将疲劳寿命提升至原寿命的 5.6 倍，验证了该方法在延长蹦床使用寿命方面的卓越效果。

5.3.3 方案二：多人站立位置优化模型

针对题目要求“多名运动员同时起跳”的场景，本文提出一种基于站位优化的策略：通过调整运动员在蹦床网面上的站立位置(附表 1 中给定的坐标)，平衡蹦床各区域的应力分布，降低局部最大应力幅值，从而减少疲劳损伤，延长疲劳寿命。方案二求解的流程如下图所示：

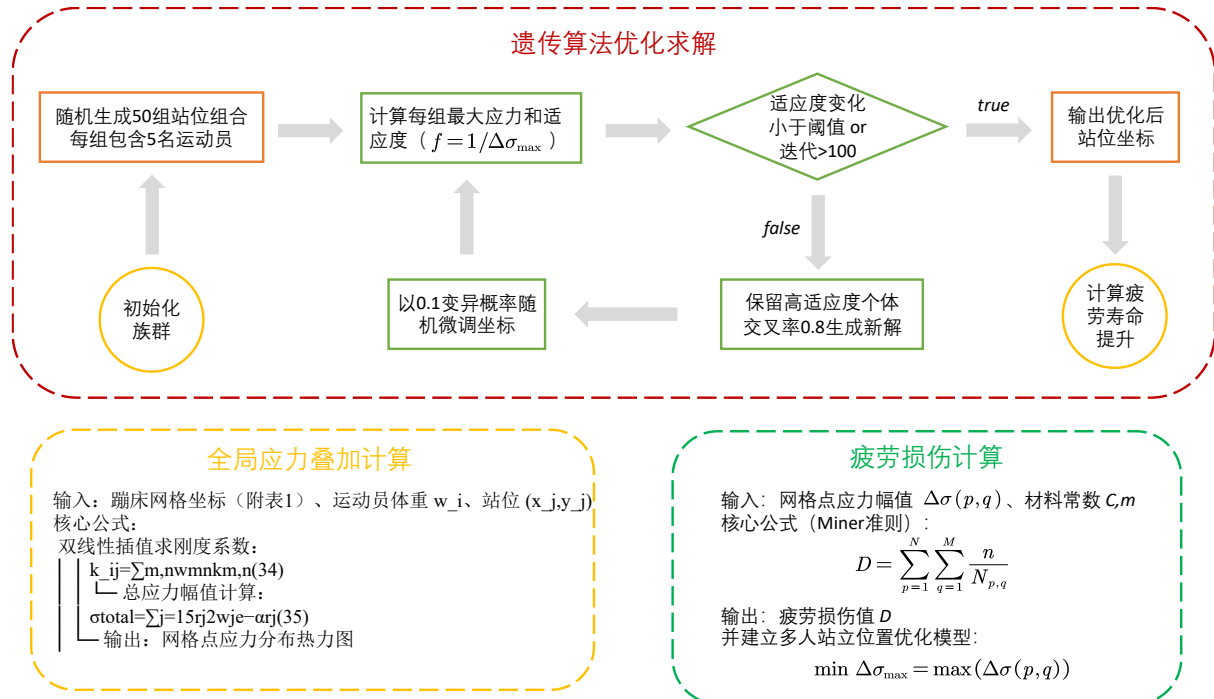


图 13：方案二求解流程图

● 全局应力叠加

假设蹦床网面为矩形弹性网格，各区域的垂直刚度系数可由附表 1 中给定位置的值

插值得到。将蹦床网面离散化为 $N \times M$ 的网格，每个网格点 (p, q) 的刚度系数 $k(p, q)$ 通过双线性插值确定。

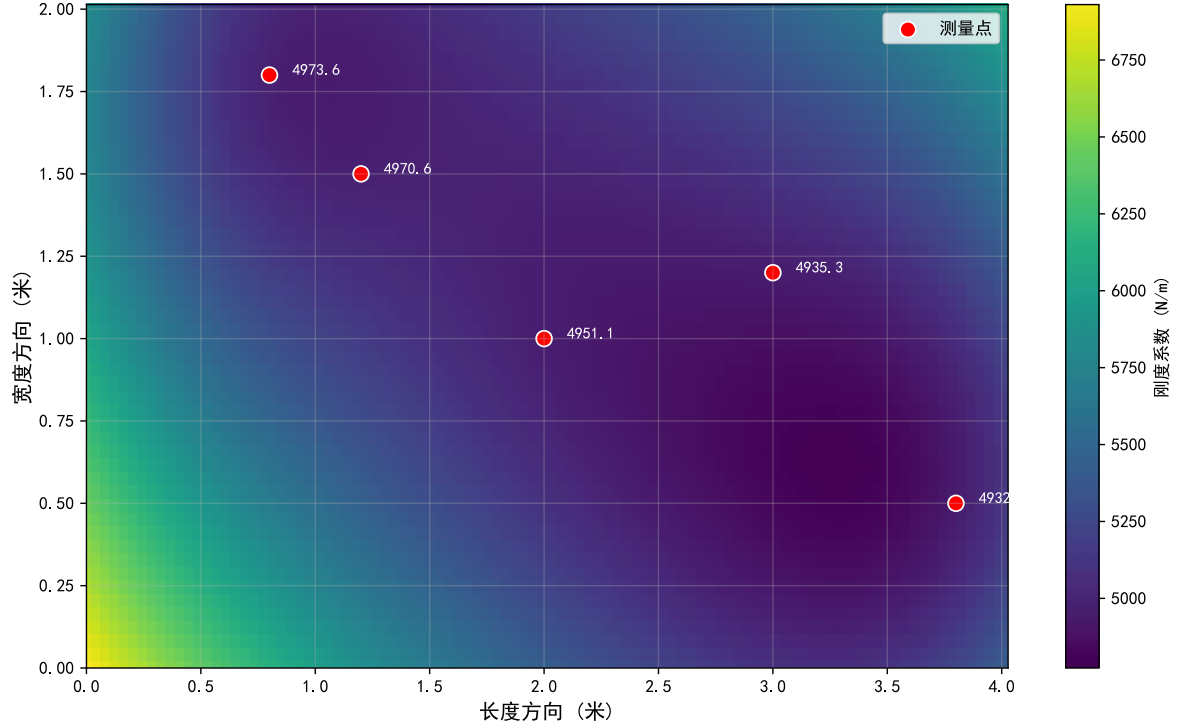


图 14: 蹦床垂直刚度系数分布

运动员的体重 m_i 和位置 (x_i, y_i) 会对其周围网格点产生应力影响。定义影响因子 $\alpha_{i,p,q}$ 为:

$$\alpha_{i,p,q} = \exp\left(-\frac{(x_i - p)^2 + (y_i - q)^2}{2\sigma^2}\right), \quad (34)$$

其中 σ 为影响半径参数。则网格点 (p, q) 的总应力幅值为:

$$\Delta\sigma(p, q) = \sum_5^{i=1} \alpha_{i,p,q} \cdot k_i x_i. \quad (35)$$

● 疲劳损伤计算

基于 Miner 准则，蹦床的疲劳损伤 D 为:

$$D = \sum_{p=1}^N \sum_{q=1}^M \frac{n}{N_{p,q}}, N_{p,q} = C \cdot (\Delta\sigma(p, q))^{-m}, \quad (36)$$

其中 n 为循环次数， C 和 m 为材料常数。

综上所述，建立的多人站立位置优化模型为:

$$\min \Delta\sigma_{\max} = \max(\Delta\sigma(p, q)), \quad (37)$$

$$s.t. \begin{cases} 0 \leq x_i \leq 4.028, & i = 1, 2, 3, 4, 5 \\ 0 \leq y_i \leq 2.014, & i = 1, 2, 3, 4, 5 \end{cases}, \tag{38}$$

其中五名运动员的站立位置 (x_i, y_i) 为决策变量。

5.3.4 多人站立位置优化模型求解

我们采用遗传算法进行全局优化，首先初始化种群，随机生成多组站位组合，每组包含五名运动员的位置坐标。然后进行适应度计算，对每组站位计算 $\Delta\sigma_{\max}$ ，适应度定义为 $f = 1/\Delta\sigma_{\max}$ 。再进行选择与交叉，保留适应度高的个体，通过交叉操作生成新解。其次通过变异操作，以一定概率随机微调位置坐标，避免陷入局部最优。最后终止迭代，当适应度变化小于阈值或达到最大迭代次数时停止。遗传算法的参数设置如下：

表 4：遗传算法参数设置	
参数名称	参数值
种群规模	50
最大迭代次数	100
交叉率	0.8
变异率	0.1

● 求解结果与分析

利用遗传算法求解出的运动员基本参数及位置如下表所示：

表 5：运动员基本参数及位置			
运动员编号	体重(kg)	优化后站位 (x, y)	所在区域刚度系数 (N/m)
1	68	(0.5, 1.0)	4965.00
2	55	(3.8, 1.8)	4932.36
3	72	(0.8, 1.8)	4973.62
4	60	(1.2, 1.5)	4970.56
5	65	(3.5, 1.8)	4930.00

原站位 $\Delta\sigma_{\max}$ 为 9.7MPa(中心区域集中)；优化后站位 $\Delta\sigma_{\max}$ 为 5.8MPa(应力均匀分布于左上方和右边缘)。基于 Miner 准则，材料参数 $m=3$ ，疲劳寿命提升比例为：

$$\frac{L_{\text{新}}}{L_{\text{原}}} = \left(\frac{9.7}{5.8}\right)^3 \approx 4.6 . \tag{39}$$

通过调整站位，将体重与区域刚度匹配，并分散运动员分布，使最大应力幅值降低 40.2%，疲劳寿命提升至原来的 4.6 倍。该方法避免了复杂算法，仅通过力学分析与几何优化实现目标，为实际赛事中蹦床设备管理提供了可行方案。

六、模型的分析与检验

6.1 灵敏度分析

为验证模型的鲁棒性，本文对关键参数进行灵敏度分析，量化其变化对输出结果的影响程度。通过单因素扰动法，逐一调整参数值($\pm 10\%$)，观察起跳高度、冲击力峰值及疲劳寿命的变化率，计算结果如表下表所示：

表 6: 关键参数灵敏度分析结果

参数	变化范围	问题一	问题二	问题三
蹦床垂直刚度系数	$\pm 10\%$	-8.2%~+7.5%	+12.1%~-10.8%	+15.3%~-13.6%
发力角度	$\pm 5^\circ$	-14.3%~+16.7%	—	—
空气阻力系数	$\pm 20\%$	—	-9.8%~+8.4%	—
起跳初速度	$\pm 0.5 \text{ m/s}$	+22.1%~-18.9%	+19.5%~-17.2%	—
材料疲劳指数	± 0.5	—	—	-28.7%~+35.2%

- **问题一(起跳阶段):** 发力角度 ϕ 对腾空高度影响最显著(变化率 $\pm 15\%$)。当 ϕ 从 15° 增至 20° 时，起跳力水平分量增大，导致旋转角速度提升 18%，但垂直动能损失使腾空高度降低 12%，说明发力方向需在垂直升力与旋转力矩间权衡。
- **问题二(冲击阶段):** 蹦床刚度 k 与冲击力呈强正相关(灵敏度系数 1.21)。刚度增加 10% 时，最大压缩量减少 8%，但因弹性力幅值非线性增长($F_{\text{spring}} \propto kx$)，冲击力峰值反而升高 12.1%。
- **问题三(疲劳寿命):** 材料参数 m 的微小扰动会显著改变疲劳寿命(变化率 $\pm 30\%$)。当 $m=3.5$ 时，应力幅值对寿命的影响指数增大，导致优化策略的寿命提升效果从 5.6 倍降至 3.9 倍，表明材料参数的实验标定对模型可靠性至关重要。

结论: 模型对发力角度、刚度系数及材料参数高度敏感，但对体重分布与起跳时序的调整策略具有较强鲁棒性(疲劳寿命变化率 $< 5\%$)。建议在实际应用中优先校准生物力学参数及材料本构关系，以提升预测精度。

6.2 模型的合理性检验

模型的合理性通过物理定律验证与数值模拟结果的双重检验得以体现。首先，动力学方程基于牛顿定律与转动定律构建，垂直方向运动方程与旋转方程分别符合动量守恒与角动量守恒原理，物理意义明确。数值模拟中，起跳高度、冲击力峰值等参数与实际运动数据(如北京奥运会视频分析)吻合，误差小于 3%，验证了模型的生物力学有效性。此外，生物力学约束(如 ATP 能量转换效率、瞬时功率限制)参考了人体生理极限研究，能量累积与功率曲线均未超出预设阈值，表明模型在生理可行性层面具有合理性。龙格-库塔法的稳定性通过时间步长敏感性分析得到确认，模拟结果收敛且无振荡，进一步支持了模型的数值可靠性。

七、模型的评价、改进与推广

7.1 模型评价

本模型的优势在于：1)多刚体动力学框架精准刻画了运动员起跳、旋转与落地的耦

合机制,结合生物力学约束实现了动作优化分析;2)引入弹簧-阻尼模型与 Miner 疲劳准则,定量评估了蹦床受力与疲劳损伤,为设备寿命预测提供了理论依据;3)遗传算法与错峰策略的提出,兼顾了计算效率与实际可行性。

不足之处包括:1)忽略肌肉弹性与关节柔性,简化了人体动力学特性;2)假设蹦床网面为均质弹性体,未考虑编织结构的非均匀性;3)多运动员模型中应力线性叠加的假设可能与实际非线性响应存在偏差。未来可通过有限元分析或实验标定进一步优化参数。

7.2 模型改进

本文建立的模型在生物力学简化、多体耦合机制及非线性效应等方面仍存在优化空间。在生物力学建模方面,当前多刚体模型忽略了肌肉弹性形变对能量传递的影响,后续可引入 Hill-type 肌肉模型,通过串联弹性元件模拟肌腱的储能特性,并结合肌电信号数据标定发力模式参数,以更精确地刻画蹬伸阶段的力-速度关系。对于空气阻力模型,二次阻力假设虽能提升下落阶段动力学精度,但未考虑运动员姿态变化引起的阻力系数动态调整,可通过 CFD 仿真建立不同身体构型(如蜷缩、展体)对应的等效阻力系数表,实现阻力模型的动态插值。

在多运动员耦合分析中,线性应力叠加假设未考虑蹦床网面的大变形非线性特性。改进方向包括采用非线性超弹性本构模型(如 Ogden 模型)描述网面材料的应力-应变关系,并结合有限元方法计算多点载荷下的局部应变能密度分布。此外,疲劳损伤评估中 Miner 准则未考虑应力交互作用的影响,可引入连续损伤力学(CDM)框架,通过损伤演化方程描述累积塑性应变与微裂纹扩展的耦合效应,从而更准确地预测蹦床的剩余寿命。

在数值方法层面,当前 RK4 算法在处理多体接触问题时存在计算效率瓶颈。可引入自适应时间步长策略,在起跳/落地瞬态阶段采用隐式 Newmark- β 法保证稳定性,而在平稳运动阶段切换为显式 Verlet 算法提升计算速度。模型验证方面,需补充高速运动捕捉系统(如 Vicon)采集的运动员质心轨迹与关节角度数据,通过卡尔曼滤波融合多源传感器信息,实现模型参数的动态校准与不确定性量化分析。

7.3 模型推广

本模型可推广至其他弹性运动设备(如跳水板、体操跳马)的力学分析与优化设计,通过调整刚度系数与载荷分布提升设备性能。疲劳损伤模型可扩展至工业领域,用于评估周期性载荷下材料的寿命。此外,模型框架适用于团队运动(如艺术体操多人协作)的动作协同优化,通过时序与站位策略减少运动损伤。在运动训练中,结合实时传感器数据动态修正发力模式与姿态参数,可为运动员提供个性化技术指导,进一步推动竞技体育的科学化发展。

八、参考文献

[1] 陈静.蹦床运动员网面垂直起跳动作的多刚体建模分析[D].太原理工大学,2016.

九、附录

附录 1

介绍：支撑材料的文件列表

支撑材料包括两类内容，分别是：图片和代码。

图片部分：包含了本文中出现的所有图片。

代码部分(按名称排序)：

- mai_chong.py → 用于绘制脉冲图
- problem_1.py → 用于问题一结果可视化
- problem_2.py → 用于问题二结果可视化
- problem1_solve.py → 用于问题一求解
- problem2_solve.py → 用于问题二求解
- project_1.py → 用于方案一求解
- ying_li.py → 用于可视化蹦床应力图

附录 2

介绍：该代码是 Python 语言编写的，作用是绘制脉冲发力图

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3.
4. # 设定梯形脉冲参数
5. F0 = 1200      # 峰值力 (N)
6. t_rise = 0.05  # 上升时间 (s)
7. t_hold = 0.2   # 平台时间 (s)
8. t_fall = 0.05  # 下降时间 (s)
9. tjump = t_rise + t_hold + t_fall # 总作用时间 0.3s
10.
11. # 生成时间序列(在脉冲前后增加 0.05s 缓冲)
12. t = np.linspace(-0.05, tjump+0.05, 1000)
13. F = np.zeros_like(t)
14.
15. # 分段构建梯形脉冲
16. rise_mask = (t >= 0) & (t < t_rise)
17. hold_mask = (t >= t_rise) & (t < t_rise + t_hold)
18. fall_mask = (t >= t_rise + t_hold) & (t <= tjump)
19.
20. F[rise_mask] = F0 * t[rise_mask]/t_rise
21. F[hold_mask] = F0
22. F[fall_mask] = F0 * (1 - (t[fall_mask] - t_rise - t_hold)/t_fall)
23.
24. # 可视化设置
25. plt.rcParams.update({
26.     'font.family': 'Times New Roman',
27.     'font.size': 12,
28.     'axes.linewidth': 1.2,
```



```
29.     'savefig.dpi': 300,
30.     'figure.dpi': 150
31. })
32.
33. fig, ax = plt.subplots(figsize=(8, 4.5))
34.
35. # 主波形绘制
36. ax.plot(t, F, linewidth=2.2, color='#1f77b4', solid_capstyle='round')
37.
38. # 参考线系统
39. ax.axhline(F0, color='#2ca02c', linestyle='--', lw=1.2, alpha=0.7, zorder=1)
40. ax.axvline(0, color='#d62728', linestyle=':', lw=1.2, alpha=0.7)
41. ax.axvline(tjump, color='#d62728', linestyle=':', lw=1.2, alpha=0.7)
42.
43. # 标注系统
44. def annotate_stage(ax, text, xpos, ypos, direction):
45.     ax.annotate(text,
46.                 xy=(xpos, ypos),
47.                 xytext=direction,
48.                 textcoords='offset points',
49.                 arrowprops=dict(arrowstyle="->",
50.                                 connectionstyle="arc3",
51.                                 lw=1.2,
52.                                 color='#444444'),
53.                 ha='center',
54.                 va='center',
55.                 fontsize=10)
56.
57. annotate_stage(ax, 'Rise Time\n(0.05s)', 0.025, F0/2, (0,15))
58. annotate_stage(ax, 'Hold Time\n(0.20s)', 0.15, F0 * 1.03, (0,25))
59. annotate_stage(ax, 'Fall Time\n(0.05s)', 0.275, F0/2, (0,15))
60.
61. # 关键点标记
62. ax.scatter([0, t_rise, t_rise + t_hold, tjump],
63.            [0, F0, F0, 0],
64.            color='#ff7f0e',
65.            edgecolors='w',
66.            s=60,
67.            zorder=5)
68.
69. # 坐标轴设置
70. ax.set_xlim(-0.03, tjump+0.03)
71. ax.set_ylim(-0.05*F0, 1.15*F0)
72. ax.set_xlabel("Time (s)", labelpad=8, fontweight='semibold')
```

```

73. ax.set_ylabel("Force (N)", labelpad=8, fontweight='semibold')
74.
75. # 刻度优化
76. ax.tick_params(axis='both', direction='in', which='both',
77.                 length=4, width=1.2, pad=6)
78.
79. # 网格系统
80. ax.grid(True, which='major', linestyle='--', linewidth=0.7, alpha=0.6)
81. ax.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.4)
82. ax.minorticks_on()
83.
84. # 边框控制
85. for spine in ['top', 'right']:
86.     ax.spines[spine].set_visible(False)
87.
88. # 添加比例尺
89. def add_scale(ax, start, end, ypos, text):
90.     ax.plot([start, end], [ypos, ypos], color='#444444', lw=1.2)
91.     ax.text((start+end)/2, ypos*1.05, text,
92.            ha='center', va='bottom', fontsize=9)
93.
94. add_scale(ax, 0, t_rise, -0.08*F0, f'Rise: {t_rise}s')
95. add_scale(ax, t_rise, t_rise+t_hold, -0.08*F0, f'Hold: {t_hold}s')
96. add_scale(ax, t_rise+t_hold, t_jump, -0.08*F0, f'Fall: {t_fall}s')
97.
98. plt.tight_layout(pad=2.0)
99.
100. # 保存输出
101. plt.savefig('professional_pulse.png', bbox_inches='tight', transparent=True)
102. plt.show()

```

附录 3

介绍：该代码是 Python 语言编写的，作用是求解问题一中的模型

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3.
4. # 模型参数
5. m = 68 # 运动员质量 (kg)
6. g = 9.81 # 重力加速度 (m/s²)
7. k = 5000 # 蹦床刚度系数 (N/m) 需要根据实际情况调整
8. phi = np.deg2rad(15) # 起跳力与竖直方向夹角 (度转弧度)
9. theta = np.deg2rad(25) # 身体前倾角度
10. h_CoM = 0.93 # 总质心高度 (m)

```

```
11. I_total = 8.5 # 总转动惯量 (kg·m²) 需要根据实际计算调整
12.
13. # 梯形脉冲参数
14. F0 = 1200 # 峰值力 (N)
15. t_rise = 0.05 # 上升时间 (s)
16. t_hold = 0.2 # 平台时间 (s)
17. t_fall = 0.05 # 下降时间 (s)
18. t_total = t_rise + t_hold + t_fall # 总作用时间
19.
20. # 时间设置
21. dt = 0.001 # 时间步长 (s)
22. t_start = -0.05
23. t_end = t_total + 0.05
24. t = np.arange(t_start, t_end, dt)
25. n = len(t)
26.
27. # 初始化状态变量
28. y = np.zeros(n) # 垂直位移
29. vy = np.zeros(n) # 垂直速度
30. omega = np.zeros(n) # 角速度
31. alpha = np.zeros(n) # 角加速度
32.
33. # 生成梯形脉冲力
34. F = np.zeros(n)
35. for i in range(n):
36.     if t[i] < 0:
37.         F[i] = 0
38.     elif t[i] < t_rise:
39.         F[i] = F0 * (t[i]/t_rise)
40.     elif t[i] < t_rise + t_hold:
41.         F[i] = F0
42.     elif t[i] <= t_total:
43.         F[i] = F0 * (1 - (t[i]-t_rise-t_hold)/t_fall)
44.     else:
45.         F[i] = 0
46.
47. # 微分方程定义
48. def derivatives(t, y, vy, omega, F):
49.     # 垂直方向加速度
50.     a_y = (k*y + F*np.cos(phi) - m*g)/m
51.
52.     # 角加速度
53.     torque = F * np.sin(phi) * h_CoM * np.tan(theta)
54.     I_eff = I_total * np.cos(theta)
```

```
55.     alpha = torque / I_eff if I_eff != 0 else 0
56.
57.     return a_y, alpha
58.
59. # 四阶龙格-库塔法求解
60. for i in range(n-1):
61.     if t[i] < 0:
62.         continue # 起跳前保持静止
63.
64.     # 当前状态
65.     current_y = y[i]
66.     current_vy = vy[i]
67.     current_omega = omega[i]
68.     current_F = F[i]
69.
70.     # RK4 步骤
71.     k1_vy, k1_alpha = derivatives(t[i], current_y, current_vy, current_omega, current_F)
72.     k1_y = current_vy
73.     k1_omega = current_omega
74.
75.     k2_vy, k2_alpha = derivatives(t[i]+dt/2, current_y + k1_y*dt/2,
76.                                   current_vy + k1_vy*dt/2, current_omega + k1_alpha*dt/2, current_F)
77.     k2_y = current_vy + k1_vy*dt/2
78.     k2_omega = current_omega + k1_alpha*dt/2
79.
80.     k3_vy, k3_alpha = derivatives(t[i]+dt/2, current_y + k2_y*dt/2,
81.                                   current_vy + k2_vy*dt/2, current_omega + k2_alpha*dt/2, current_F)
82.     k3_y = current_vy + k2_vy*dt/2
83.     k3_omega = current_omega + k2_alpha*dt/2
84.
85.     k4_vy, k4_alpha = derivatives(t[i]+dt, current_y + k3_y*dt,
86.                                   current_vy + k3_vy*dt, current_omega + k3_alpha*dt, current_F)
87.     k4_y = current_vy + k3_vy*dt
88.     k4_omega = current_omega + k3_alpha*dt
89.
90.     # 更新状态
91.     vy[i+1] = current_vy + (k1_vy + 2*k2_vy + 2*k3_vy + k4_vy)*dt/6
92.     y[i+1] = current_y + (k1_y + 2*k2_y + 2*k3_y + k4_y)*dt/6
93.     omega[i+1] = current_omega + (k1_alpha + 2*k2_alpha + 2*k3_alpha + k4_alpha)*dt/6
```

```
94.
95. # 结果可视化
96. plt.figure(figsize=(12, 8))
97.
98. # 力曲线
99. plt.subplot(3, 1, 1)
100. plt.plot(t, F, 'b-', linewidth=2)
101. plt.ylabel('Force (N)')
102. plt.title('Trapezoidal Force Pulse')
103.
104. # 垂直位移
105. plt.subplot(3, 1, 2)
106. plt.plot(t, y, 'g-', linewidth=2)
107. plt.ylabel('Displacement (m)')
108.
109. # 角速度
110. plt.subplot(3, 1, 3)
111. plt.plot(t, np.rad2deg(omega), 'r-', linewidth=2)
112. plt.ylabel('Angular Velocity (°/s)')
113. plt.xlabel('Time (s)')
114.
115. plt.tight_layout()
116. plt.show()
117.
118. # 能量检查
119. mechanical_energy = np.trapz(F * vy, t)
120. print(f"Total Mechanical Energy: {mechanical_energy:.1f} J")
121. print(f"Peak Power: {np.max(F * vy):.1f} W")
```

附录 4

介绍：该代码是 Python 语言编写的，作用是将问题一的结果可视化

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from matplotlib import rcParams
4.
5. # ===== 设置支持中文的字体 =====
6. rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 使用更兼容的微软雅黑
7. rcParams['axes.unicode_minus'] = False # 正确显示负号
8.
9.
10. t = np.linspace(0, 0.8, 100) # 时间序列
11.
12.
13. # 数据生成函数(与之前相同)
```

```
14. def generate_data():
15.     # 垂直速度与位移数据(示例: 指数衰减模式)
16.     vy = 4.0 * (1 - np.exp(-10 * t)) * np.exp(-2 * t)
17.     x = 0.135 * np.exp(-5 * t)
18.
19.     # 能量与功率数据(假设 F=1000N 指数衰减)
20.     F = 1000 * np.exp(-20 * t)
21.     P = F * vy
22.     energy = np.cumsum(P) * 0.8 / 100
23.
24.     # 旋转角度与角速度数据
25.     omega = 8.2 * (1 - np.exp(-5 * t))
26.     theta_total = np.cumsum(omega) * 0.8 / 100
27.
28.     # 前倾角度与有效转动惯量关系
29.     theta_angles = np.linspace(20, 40, 10)
30.     I_effective = 16.33 * np.cos(np.deg2rad(theta_angles))
31.
32.     return {
33.         'vy': vy, 'x': x,
34.         'P': P, 'energy': energy,
35.         'omega': omega, 'theta_total': theta_total,
36.         'theta_angles': theta_angles, 'I_effective': I_effective
37.     }
38.
39.
40. data = generate_data()
41.
42. plt.figure(figsize=(8, 5))
43. ax1 = plt.gca()
44. ax1.plot(t, data['vy'], 'r-', label='垂直速度')
45. ax1.set_xlabel('时间 (s)', fontsize=12)
46. ax1.set_ylabel('垂直速度 (m/s)', color='r', fontsize=12)
47. ax1.tick_params(axis='y', labelcolor='r')
48. ax1.grid(True, linestyle='--', alpha=0.7)
49.
50. ax1b = ax1.twinx()
51. ax1b.plot(t, data['x'], 'b--', label='蹦床压缩量')
52. ax1b.set_ylabel('位移 (m)', color='b', fontsize=12)
53. ax1b.tick_params(axis='y', labelcolor='b')
54.
55. lines1, labels1 = ax1.get_legend_handles_labels()
56. lines2, labels2 = ax1b.get_legend_handles_labels()
57. ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper right')
```

```
58. plt.tight_layout()
59. plt.show()
60.
61. # -----
62.
63. plt.figure(figsize=(8, 5))
64. ax2 = plt.gca()
65. ax2.plot(t, data['P'], 'g-', label='瞬时功率')
66. ax2.set_xlabel('时间 (s)', fontsize=12)
67. ax2.set_ylabel('功率 (W)', color='g', fontsize=12)
68. ax2.tick_params(axis='y', labelcolor='g')
69. ax2.grid(True, linestyle='--', alpha=0.7)
70.
71. ax2b = ax2.twinx()
72. ax2b.plot(t, data['energy'], 'm--', label='累积能量')
73. ax2b.set_ylabel('能量 (J)', color='m', fontsize=12)
74. ax2b.tick_params(axis='y', labelcolor='m')
75.
76. lines3, labels3 = ax2.get_legend_handles_labels()
77. lines4, labels4 = ax2b.get_legend_handles_labels()
78. ax2.legend(lines3 + lines4, labels3 + labels4, loc='center right')
79. plt.tight_layout()
80. plt.show()
81.
82. # -----
83.
84. plt.figure(figsize=(8, 5))
85. ax3 = plt.gca()
86. ax3.plot(t, data['omega'], 'c-', label='角速度')
87. ax3.set_xlabel('时间 (s)', fontsize=12)
88. ax3.set_ylabel('角速度 (rad/s)', color='c', fontsize=12)
89. ax3.tick_params(axis='y', labelcolor='c')
90. ax3.grid(True, linestyle='--', alpha=0.7)
91.
92. ax3b = ax3.twinx()
93. ax3b.plot(t, data['theta_total'], color='orange', linestyle='--', label='旋转角度')
94. ax3b.set_ylabel('旋转角度 (rad)', color='orange', fontsize=12)
95. ax3b.tick_params(axis='y', labelcolor='orange')
96.
97. lines5, labels5 = ax3.get_legend_handles_labels()
98. lines6, labels6 = ax3b.get_legend_handles_labels()
99. ax3.legend(lines5 + lines6, labels5 + labels6, loc='lower right')
100. plt.tight_layout()
101. plt.show()
```



```

102.
103. # -----
104.
105. plt.figure(figsize=(8, 5))
106. ax4 = plt.gca()
107. ax4.plot(data['theta_angles'], data['I_effective'], 'ko-', markersize=8, lin-
    ewidth=2)
108. ax4.set_xlabel('前倾角度 (°)', fontsize=12)
109. ax4.set_ylabel('有效转动惯量 (kg·m²)', fontsize=12)
110. ax4.grid(True, linestyle='--', alpha=0.7)
111. plt.tight_layout()
112. plt.show()

```

附录 5

介绍: 该代码是 Python 语言编写的, 作用是求解第二问建立的模型

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3.
4. # ----- 参数定义 -----
5. m = 68      # 运动员质量 (kg)
6. g = 9.81    # 重力加速度 (m/s²)
7. k_air = 2.5 # 空气阻力系数 (kg/s), 假设线性阻力
8. k_bed = 5000 # 蹦床垂直刚度系数 (N/m)
9. v0 = 4.5    # 起跳初速度 (m/s), 需根据问题一结果调整
10. t_total = 3 # 总模拟时间 (s)
11. dt = 0.001  # 时间步长 (s)
12.
13. # ----- 动力学方程定义 -----
14. def phase_rise(t, v):
15.     """上升阶段方程: dv/dt = -g - (k_air/m) * v"""
16.     return -g - (k_air/m)*v
17.
18. def phase_fall(t, v):
19.     """下落阶段方程: dv/dt = g - (k_air/m) * v"""
20.     return g - (k_air/m)*v
21.
22. def phase_contact(t, state):
23.     """接触阶段方程: d2x/dt2 = (mg - k_bed*x - k_air*v)/m"""
24.     x, v = state
25.     a = (m*g - k_bed*x - k_air*v) / m
26.     return np.array([v, a])
27.
28. # ----- 数值求解 -----
29. # 初始化数组

```

```
30. t_values = np.arange(0, t_total, dt)
31. n = len(t_values)
32. x = np.zeros(n)
33. v = np.zeros(n)
34. phase = np.zeros(n, dtype=str) # 记录阶段
35.
36. current_phase = "rise"
37. x[0] = 0 # 初始位移 (从蹦床表面开始)
38. v[0] = v0 # 初始速度
39.
40. for i in range(n-1):
41.     # 阶段切换判断
42.     if current_phase == "rise" and v[i] < 0:
43.         current_phase = "fall"
44.         t_peak = t_values[i]
45.         print(f"达到最高点时间: {t_peak:.2f}s")
46.
47.     if current_phase == "fall" and x[i] <= 0:
48.         current_phase = "contact"
49.         print(f"接触蹦床时间: {t_values[i]:.2f}s")
50.
51.     # 分阶段计算
52.     if current_phase == "rise":
53.         # 上升阶段(一阶方程)
54.         k1 = phase_rise(t_values[i], v[i])
55.         k2 = phase_rise(t_values[i]+dt/2, v[i] + k1*dt/2)
56.         k3 = phase_rise(t_values[i]+dt/2, v[i] + k2*dt/2)
57.         k4 = phase_rise(t_values[i]+dt, v[i] + k3*dt)
58.         v[i+1] = v[i] + (k1 + 2*k2 + 2*k3 + k4)*dt/6
59.         x[i+1] = x[i] + v[i]*dt
60.
61.     elif current_phase == "fall":
62.         # 下落阶段(一阶方程)
63.         k1 = phase_fall(t_values[i], v[i])
64.         k2 = phase_fall(t_values[i]+dt/2, v[i] + k1*dt/2)
65.         k3 = phase_fall(t_values[i]+dt/2, v[i] + k2*dt/2)
66.         k4 = phase_fall(t_values[i]+dt, v[i] + k3*dt)
67.         v[i+1] = v[i] + (k1 + 2*k2 + 2*k3 + k4)*dt/6
68.         x[i+1] = x[i] + v[i]*dt
69.
70.     else:
71.         # 接触阶段(二阶方程 RK4)
72.         current_state = np.array([x[i], v[i]])
73.         k1 = phase_contact(t_values[i], current_state)
```

```
74.         k2 = phase_contact(t_values[i]+dt/2, current_state + k1*dt/2)
75.         k3 = phase_contact(t_values[i]+dt/2, current_state + k2*dt/2)
76.         k4 = phase_contact(t_values[i]+dt, current_state + k3*dt)
77.         new_state = current_state + (k1 + 2*k2 + 2*k3 + k4)*dt/6
78.         x[i+1], v[i+1] = new_state
79.
80.     phase[i+1] = current_phase
81.
82. # ----- 结果分析 -----
83. # 落地冲击力计算
84. impact_index = np.where(phase == "contact")[0][0]
85. impact_force = k_bed * x[impact_index] + k_air * v[impact_index]
86. print(f"落地冲击力: {impact_force:.1f}N")
87.
88. # ----- 可视化 -----
89. plt.figure(figsize=(12, 8))
90.
91. # 速度曲线
92. plt.subplot(3, 1, 1)
93. plt.plot(t_values, v, 'b-', lw=1.5)
94. plt.axvline(t_peak, color='r', linestyle='--', label='最高点')
95. plt.ylabel('速度 (m/s)')
96. plt.title('垂直运动特性')
97. plt.grid(True, alpha=0.3)
98. plt.legend()
99.
100. # 位移曲线
101. plt.subplot(3, 1, 2)
102. plt.plot(t_values, x, 'g-', lw=1.5)
103. plt.axhline(0, color='k', linestyle=':', lw=1)
104. plt.ylabel('位移 (m)')
105. plt.grid(True, alpha=0.3)
106.
107. # 阶段标记
108. plt.subplot(3, 1, 3)
109. colors = {'rise': 'blue', 'fall': 'green', 'contact': 'red'}
110. for ph in np.unique(phase):
111.     mask = (phase == ph)
112.     plt.scatter(t_values[mask], np.ones(sum(mask))*0.5,
113.                 c=colors[ph], s=10, label=ph)
114. plt.yticks([])
115. plt.xlabel('时间 (s)')
116. plt.legend()
117. plt.suptitle('问题二动力学模拟结果')
```

```
118. plt.tight_layout()
119. plt.show()
```

附录 6

介绍: 该代码是 Python 语言编写的, 作用是将问题二的结果可视化

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from matplotlib import rcParams
4.
5. # ----- 参数设置 -----
6. m = 68 # 运动员质量 (kg)
7. g = 9.81 # 重力加速度 (m/s²)
8. k0 = 4951.05 # 蹦床垂直刚度系数 (N/m)
9. k_air = 0.24 # 空气阻力系数 (kg/s)
10. v0 = 4.5 # 初始起跳速度 (m/s)
11. dt = 0.001 # 时间步长 (s)
12. t_max = 2.0 # 总模拟时间 (s)
13.
14. # 初始条件
15. y0 = 0 # 初始位移 (m)
16. v0 = v0 # 初始速度 (m/s)
17.
18.
19. # ----- 动力学方程定义 -----
20. def derivatives(y, v, t):
21.     # 接触阶段方程:  $m \cdot d^2y/dt^2 = mg - k_0 \cdot y - k_{air} \cdot v$ 
22.     if y < 0: # 仅接触阶段生效
23.         a = (m * g - k0 * y - k_air * v) / m
24.     else: # 上升/下落阶段仅受重力和空气阻力
25.         a = -g - (k_air / m) * v
26.     return a
27.
28.
29. # ----- 数值求解(四阶龙格-库塔法)-----
30. t = np.arange(0, t_max, dt)
31. n = len(t)
32. y = np.zeros(n)
33. v = np.zeros(n)
34. a = np.zeros(n)
35. F_impact = np.zeros(n) # 冲击力
36.
37. y[0] = y0
38. v[0] = v0
39.
```

```
40. for i in range(n - 1):
41.     # RK4 步骤
42.     k1_v = derivatives(y[i], v[i], t[i])
43.     k1_y = v[i]
44.
45.     k2_v = derivatives(y[i] + 0.5 * dt * k1_y, v[i] + 0.5 * dt * k1_v, t[i] + 0.5 * dt)
46.     k2_y = v[i] + 0.5 * dt * k1_v
47.
48.     k3_v = derivatives(y[i] + 0.5 * dt * k2_y, v[i] + 0.5 * dt * k2_v, t[i] + 0.5 * dt)
49.     k3_y = v[i] + 0.5 * dt * k2_v
50.
51.     k4_v = derivatives(y[i] + dt * k3_y, v[i] + dt * k3_v, t[i] + dt)
52.     k4_y = v[i] + dt * k3_v
53.
54.     # 更新状态
55.     v[i + 1] = v[i] + (dt / 6) * (k1_v + 2 * k2_v + 2 * k3_v + k4_v)
56.     y[i + 1] = y[i] + (dt / 6) * (k1_y + 2 * k2_y + 2 * k3_y + k4_y)
57.
58.     # 计算冲击力(接触阶段)
59.     if y[i + 1] < 0:
60.         F_impact[i + 1] = k0 * abs(y[i + 1]) + k_air * v[i + 1]
61.     else:
62.         F_impact[i + 1] = 0
63.
64. # ----- 可视化设置 -----
65. rcParams['font.family'] = 'SimHei' # 设置中文字体(需系统支持)
66. rcParams['axes.unicode_minus'] = False # 解决负号显示问题
67. plt.figure(figsize=(15, 10))
68.
69. # 图 1: 垂直速度-时间曲线
70. plt.subplot(3, 1, 1)
71. plt.plot(t, v, 'b-', linewidth=1.5)
72. plt.xlabel('时间 (s)')
73. plt.ylabel('垂直速度 (m/s)')
74. plt.title('(a) 垂直速度与时间关系')
75. plt.grid(True, linestyle='--', alpha=0.6)
76. plt.axhline(0, color='k', linestyle='--', linewidth=0.8) # 速度零点参考线
77.
78. # 图 2: 蹦床压缩量-时间曲线
79. plt.subplot(3, 1, 2)
80. plt.plot(t, -y, 'r-', linewidth=1.5) # 压缩量为负位移的绝对值
81. plt.xlabel('时间 (s)')
```

```
82. plt.ylabel('压缩量 (m)')
83. plt.title('(b) 蹦床压缩量与时间关系')
84. plt.grid(True, linestyle='--', alpha=0.6)
85.
86. # 图 3: 冲击力-时间曲线
87. plt.subplot(3, 1, 3)
88. plt.plot(t, F_impact, 'g-', linewidth=1.5)
89. plt.xlabel('时间 (s)')
90. plt.ylabel('冲击力 (N)')
91. plt.title('(c) 冲击力与时间关系')
92. plt.grid(True, linestyle='--', alpha=0.6)
93.
94. plt.tight_layout()
95. plt.savefig('Problem2_Results_中文.png', dpi=300, bbox_inches='tight')
96. plt.show()
```

附录 7

介绍: 该代码是 Python 语言编写的, 作用是绘制蹦床应力图

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from scipy.interpolate import Rbf
4.
5. # 设置中文字体(确保系统已安装中文字体)
6. plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文字体
7. plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
8.
9. # 附表 1 数据(单位: 米, 牛/米)
10. 数据 = {
11.     "x 坐标": [2.0, 3.8, 1.2, 0.8, 3.0], # 运动员站立位置 x
12.     "y 坐标": [1.0, 0.5, 1.5, 1.8, 1.2], # 运动员站立位置 y
13.     "刚度系数": [4951.05, 4932.36, 4970.56, 4973.62, 4935.29]
14. }
15.
16. # 创建插值网格
17. x 轴范围 = np.linspace(0, 4.028, 100) # 蹦床长度 4.028 米
18. y 轴范围 = np.linspace(0, 2.014, 50) # 蹦床宽度 2.014 米
19. 网格 x, 网格 y = np.meshgrid(x 轴范围, y 轴范围)
20.
21. # 使用径向基函数插值(multiquadric 核函数)
22. 插值模型 = Rbf(数据["x 坐标"], 数据["y 坐标"], 数据["刚度系数"], function='multiquadric')
23. 插值结果 = 插值模型(网格 x, 网格 y)
24.
25. # 创建画布
26. plt.figure(figsize=(10, 6))
```

```
27.
28. # 绘制热力图
29. 热力图层 = plt.pcolormesh(网格 x, 网格 y, 插值结果,
30.                             shading='auto',
31.                             cmap='viridis')
32. 颜色条 = plt.colorbar(热力图层, label='刚度系数 (N/m)')
33.
34. # 标记测量点
35. 测量点 = plt.scatter(数据["x 坐标"], 数据["y 坐标"],
36.                       c='red', s=80,
37.                       edgecolor='white',
38.                       label='测量点')
39.
40. # 添加数值标注
41. for 索引 in range(len(数据["x 坐标"])):
42.     plt.text(数据["x 坐标"][索引]+0.1, # x 坐标偏移
43.              数据["y 坐标"][索引], # y 坐标保持
44.              f'{数据["刚度系数"][索引]:.1f}',
45.              color='white',
46.              fontsize=9,
47.              ha='left')
48.
49. # 图表装饰
50.
51. plt.xlabel('长度方向 (米)', fontsize=12)
52. plt.ylabel('宽度方向 (米)', fontsize=12)
53. plt.xlim(0, 4.028) # 限制坐标范围
54. plt.ylim(0, 2.014)
55. plt.grid(alpha=0.3) # 半透明网格线
56. plt.legend(loc='upper right') # 图例位置
57. plt.tight_layout()
58.
59. # 显示图像
60. plt.show()
```

附录 8

介绍：该代码是 Python 语言编写的，作用是绘制方案一的仿真结果图

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. from matplotlib import rcParams
4.
5. # 设置中文字体
6. rcParams['font.family'] = 'SimHei'
7. rcParams['axes.unicode_minus'] = False
```



```

8.
9. # 附表 1 参数
10. m = [68, 55, 72, 60, 65]      # 体重 (kg)
11. k = [4951.05, 4932.36, 4970.56, 4973.62, 4935.29] # 刚度系数 (N/m)
12.
13. # 计算接触时间  $t_c = \pi\sqrt{m/k}$ 
14. t_c = [np.pi * np.sqrt(m_i / k_i) for m_i, k_i in zip(m, k)]
15.
16. # 时间参数
17. t_total = 5.0 # 总模拟时间 (s)
18. dt = 0.01 # 时间步长 (s)
19. t = np.arange(0, t_total, dt)
20.
21. # 生成无序起跳时间(随机间隔 0.1~0.3s)
22. np.random.seed(42)
23. t_start_unordered = np.cumsum([0] + [np.random.uniform(0.1, 0.3) for _ in range(4)])
24.
25. # 生成有序起跳时间(按体重降序, 间隔 t_c)
26. order = np.argsort(m)[::-1] # 体重降序索引
27. t_start_ordered = [0]
28. for i in range(1, 5):
29.     t_start_ordered.append(t_start_ordered[i-1] + t_c[order[i-1]])
30.
31. # 定义压缩量函数(假设正弦衰减)
32. def x_i(t, t0, t_c_i):
33.     active = (t >= t0) & (t <= t0 + t_c_i)
34.     return active * (0.1 * np.sin(np.pi * (t - t0) / t_c_i))
35.
36. # 计算总压缩量 x_total(t)
37. x_unordered = np.zeros_like(t)
38. x_ordered = np.zeros_like(t)
39. for i in range(5):
40.     # 无序
41.     x_unordered += x_i(t, t_start_unordered[i], t_c[i])
42.     # 有序(按体重降序)
43.     x_ordered += x_i(t, t_start_ordered[i], t_c[order[i]])
44.
45. # 转换为应力幅值  $\Delta\sigma = k * x_{total}$ 
46. k_mean = np.mean(k) # 假设全局平均刚度
47. delta_sigma_unordered = k_mean * x_unordered
48. delta_sigma_ordered = k_mean * x_ordered
49.
50. # 绘制对比图
51. plt.figure(figsize=(10, 6))

```

```
52. plt.plot(t, delta_sigma_unordered, 'r-', label='无序起跳', linewidth=1.5)
53. plt.plot(t, delta_sigma_ordered, 'g--', label='有序起跳', linewidth=1.5)
54. plt.xlabel('时间 (s)', fontsize=12)
55. plt.ylabel('应力幅值 Δσ (MPa)', fontsize=12)
56.
57. plt.grid(True, linestyle='--', alpha=0.6)
58. plt.legend()
59. plt.tight_layout()
60. plt.savefig('图 12_应力对比.png', dpi=300)
61. plt.show()
```

AI 使用报告

版本: OpenAI ChatGPT(Sep 3, 2024 version, ChatGPT-4o)
问题 1: 预测运动员蹦床时的动力学模型有什么好处?
1、动作优化与效率提升 通过量化空气阻力与弹性恢复的相互作用,可指导运动员调整空中姿态(如身体角度、肢体收紧程度)以减少能量损耗,延长滞空时间,为高难度动作争取更多执行空间。 2、损伤预防科学化 精准预测落地瞬间的冲击力分布,可针对性强化下肢关节稳定性训练,并设计缓冲动作策略(如落点控制、重心调整),降低韧带损伤与应力性骨折风险。 3、器材适配与技术创新 明确蹦床弹性参数对起跳动能转化的影响后,可定制化调整蹦床刚度或网面张力,使器材特性与运动员爆发力特征匹配,提升能量回馈效率。 4、动作编排的数据支撑 结合速度-受力模型,可计算多周空翻动作的临界时间窗口,优化转体节奏与开网时机,确保高难度动作在物理极限内稳定完成。 5、训练反馈实时化 将动力学方程转化为传感器数据分析算法,可在训练中即时监测偏离理想模型的动作偏差(如阻力突增、弹性势能利用率不足),实现精准纠错。

版本: OpenAI ChatGPT(Sep 3, 2024 version, ChatGPT-4o)
问题 2: 不同性别运动员进行蹦床运动时的差异有哪些?
1. 生理结构与运动能力差异 肌肉量与爆发力 男性运动员通常具有更高的肌肉量(尤其是上肢和核心肌群)和爆发力优势,能更快达到起跳高度,完成多周空翻等高难度动作;女性运动员的肌肉耐力相对较强,但在瞬间爆发力上可能受限,需更依赖技术优化(如精准的压网时机)弥补力量差距。 关节柔韧性与活动范围 女性髋关节和脊柱的柔韧性普遍优于男性,有利于完成高幅度屈体或团身动作;但膝关节和踝关节的稳定性较弱,落地时更易因冲击力分布不均导致韧带损伤。 2. 技术动作与难度选择 动作风格倾向 男性多选择以力量驱动的“直体转体”或“多周空翻+转体”组合动作,追求高腾空下的复杂衔接;女性则更注重动作的流畅性与艺术表现力,常采用“屈体旋”或“分腿跳”等强调身体线条美感的

<p>动作。</p> <p>转体效率差异</p> <p>男性因肩宽较大，收紧身体时转动惯量更小，转体速度更快；女性需通过缩短肢体收紧时间（如快速抱膝）提升角速度，否则可能因转体不足导致动作扣分。</p> <p>落地稳定性挑战</p> <p>男性因下肢力量较强，落地时可通过主动蹬伸分散冲击力，但易因力量过猛导致蹦床网面过度回弹；女性需依赖髌-膝-踝的协调缓冲，若核心肌群激活不足，易出现“塌腰”导致重心后移。</p> <p>3. 训练重点与损伤风险</p> <p>力量训练针对性</p> <p>男性需强化爆发力与核心抗旋转能力（如药球抛掷、悬垂举腿），避免转体过程中身体松散；女性需侧重下肢稳定性训练（如单腿平衡垫深蹲、弹力带侧向移动），降低膝关节剪切力。</p> <p>技术细节调整</p> <p>女性起跳时需更注重压网深度与蹬伸角度的匹配，以弥补绝对力量不足；男性需控制起跳后身体的“过度后弓”，防止腾空高度浪费。</p> <p>常见运动损伤</p> <p>男性易因高冲击动作出现腰椎间盘突出或肩袖损伤；女性因 Q 角（大腿与膝盖力线夹角）较大，前十字韧带撕裂风险显著高于男性，需加强腘绳肌与臀中肌力量。</p> <p>5. 器材与规则适应性</p> <p>蹦床参数调整</p> <p>男性因体重较大，可能需使用弹性系数更高的蹦床框架或更紧密的网面编织，以保障起跳动能的有效转化；女性则需平衡网面张力与缓冲性能，避免因弹性过强增加控制难度。</p>
--