arXiv:2603.00871v1 [cs.RO] 1 Mar 2026

Hippo：面向通用约束轨迹优化的高性能内点与基于投影的求解器

Haizhou Zhao[1,2] ,Ludovic Righetti[1] ,Majid Khadiv[2]

摘要——轨迹优化是现代基于模型的机器人控制与运动规划的核心。现有基于序列二次规划（SQP）或微分动态规划（DDP）的轨迹优化器，往往受限于计算效率低、建模灵活性不足，以及在需要硬约束的复杂任务中收敛性差。本文介绍*Hippo*[i]，这是一种求解器，可使用带有自适应障碍更新策略的内点法（IPM）处理不等式约束，并通过投影或 IPM 处理硬等式约束。通过大量数值基准测试，我们表明，*Hippo* 是现有最先进求解器的稳健且高效的替代方案，适用于需要高质量解的困难机器人轨迹优化问题，例如运动与操作。

I. 引言

直接多重射击（DMS）凭借其出色的稳健性和更快的收敛速度，在现代机器人轨迹优化中发挥着重要作用 [1]。多种方法利用最优控制问题（OCP）中的时间稀疏结构，通过黎卡提递推 [2], [3], [4] 或分块消元 [5], [6], 加速求解，这两种方法在本质上等价 [5]。尽管其中许多方法已成功用于模型预测控制（MPC），但由于收敛性差、计算速度慢，它们往往无法求解大规模或困难的 OCP。这些局限妨碍了它们在行为克隆（BC）或任务与运动规划（TAMP）的数据生成中的可靠应用。本文提出一种旨在解决这些问题的轨迹优化器。

A. 相关工作

常见的约束处理技术包括内点法（IPM） [7], [8], [9] 和增广拉格朗日法（ALM） [3], [10], [4], [5], [6]。两者都具有同伦参数（即 IPM 的障碍参数和 ALM 的惩罚参数），并通过迭代减小这些参数来求得原始最优控制问题的解。

内点法通过引入松弛变量，将不等式约束转化为松弛等式约束，并使用不断变化的障碍参数放宽KKT条件中的互补松弛。*CALIPSO* [11]和 *robotoc* [8] 实现了内点法的单调更新规则：当KKT残差低于阈值时，便减小障碍参数。不过，这种方法可能对初始障碍参数 [12] 的选择较为敏感。*acados* [7]是一个基于*HPIPM* [13]，（基于内点法的二次规划求解器）的序列二次规划求解器，它采用Mehrotra 预测-校正法（MP-C）自适应障碍更新策略 [14]。它将等式约束表示为边界为零的盒约束不等式。在每次序列二次规划迭代中，它都会调用HPIPM，以可接受的精度求解分阶段线性二次（LQ）子问题。近期，*piqp* [15] 将乘子近端法（类似于增广拉格朗日法）与采用MP-C的内点法相结合，并在*qpbenchmark* [16]中取得了最先进的性能。

增广拉格朗日法通过对约束违反进行惩罚，将受约束优化问题转化为无约束优化问题。具体而言，*aligator* [4] 和 *mim* 求解器 [5] 采用类似增广拉格朗日法的算法，并以半光滑牛顿形式实现，这是原始-对偶主动集策略（PDAS） [17]的一种变体。增广拉格朗日法的主要问题在于，由于需要显式检测主动集[9], , 它在求解受不等式约束的最优控制问题时收敛性较差，而且很容易陷入局部极小值。相比之下，内点法得益于障碍函数设计的自协调性，具有多项式收敛复杂度 [18]。增广拉格朗日法的另一个问题是惩罚参数调度方案，这仍是一个有待解决的问题 [19]。相反，内点法拥有系统化的自适应障碍更新策略，例如 MPC 和质量函数 [12]。

*acados*和*mim* 求解器 的两阶段精确最小化框架，会在内循环中近似求解线性二次子问题。对于类似增广拉格朗日法的方法，这有时会有所帮助，因为它通过仅对值函数雅可比矩阵进行廉价递归，提高了找到主动集的可能性。对于内点法，这类内循环的计算开销可能很大，因为松弛变量和乘子不断变化，无法重复利用海森矩阵分解。此外，对于非线性优化，这种框架还引入了额外的求解器参数，用于控制内循环的终止，例如子问题容差。这些参数可能高度依赖具体问题，需要仔细地手动调整。

IPM 和增广拉格朗日法在其逐阶段 KKT 系统中具有共同的稀疏性，因此可以采用类似的流程求解：在递归阶段，仅计算原始步；随后在递归后阶段，并行重构对偶步 [8], [5]。在 [4],中，有人指出，这种拆分由于需要求逆对角系数而存在数值条件性较差的问题，而这些系数在准收敛时可能达到极端值。相反， [4] 选择使用适用于半正定 KKT 线性系统的 Cholesky LDLT 分解，但代价是计算速度较慢，因为 LDLT 分解以及更高维度的矩阵乘法具有较高的复杂度。为了进一步加快优化速度，*ocs2* [20]通过将 OCP 的因果时间结构拆分为多个片段，并行化顺序递归，使

[1]纽约大学坦登工程学院，美国，haizhou.zhao@nyu.edu，ludovic.righetti@nyu.edu[2]慕尼黑工业大学机器人与机器智能研究所（MIRMI），德国majid.khadiv@tum.de[i] GitHub 链接：https://github.com/HaizhouZ/moto

表 I 求解器属性

求解器	并行化	Eq.[a]	不等式约束	动力学
Hippo（本文方法）	yes	投影[b]/内点法	IPM	隐式
fatrop [26]	no	投影	IPM	显式
acados [7]	no	IPM	IPM	隐式
aligator [4], [6]	yes	ALM	ALM	隐式
mim-求解器 [5]	yes	ALM	ALM	显式

[a]等式约束/不等式约束 - 等式约束/不等式约束处理。

[b]基于投影的处理。

得每个片段上的递归都可以利用上一次迭代的信息开始。这对应于经典的时间并行积分，用于求解初值问题[21]。尽管速度有所提升，但这种不一致性可能会减慢收敛速度，甚至导致发散。*aligator*[6] 利用相对于协状态（即断点处的动力学乘子）的分段灵敏度来拆分轨迹。尽管其解是精确的，不会影响收敛，但额外灵敏度的计算和分解会使速度提升不太理想。

除专用最优控制问题求解器外，*IPOPT* [22] 还被认为是一种通用且稳健的非线性规划（NLP）求解器，并被广泛用于求解复杂的多接触轨迹优化问题 [23], [24], [25]。与需要两阶段精确极小化的*acados* 相比，*IPOPT*在每次序列二次规划迭代中计算原始-对偶牛顿步，并采用强大的全局化策略，包括可行性恢复阶段和滤子线搜索。尽管具有稳健性，*IPOPT*并未利用最优控制问题中的时间结构，这使其速度较慢，并限制了其在机器人学中的应用。最近，有研究提出*fatrop*[26] ，旨在模仿*ipopt* ，同时利用最优控制问题的KKT 块稀疏性。然而，它仅适用于显式动力学，且缺乏有效的并行性。上述求解器的总结见表 I。

B. 主要贡献

本文提出了一种正则化内点法和基于投影的约束序列二次规划的简化求解器实现，即*Hippo*。它支持通用的最优控制问题表述，包括隐式动力学和跨时间步纯状态约束。与*IPOPT*类似，在每次序列二次规划迭代中，*Hippo*直接求解用于原始-对偶牛顿步的近似二次规划，并通过投影或内点法处理硬等式约束。受 [15], [12],启发，我们采用MP-C 方案，并对试探互补残差设置保护机制。我们证明，*Hippo* 能够在困难的轨迹优化问题上实现稳健收敛。此外，凭借最少的调参参数以及简单而有效的全局化策略，*Hippo* 无需对求解器参数进行细致的手动调节。我们与最先进的求解器进行了系统比较，结果表明，*Hippo* 在收敛速度和可求解问题数量方面均具有优越性。

II. 约束最优控制

记号：本文中，粗体字符表示向量/矩阵值，普通字符表示标量。我们使用 $F_{\mathbf{a}}$ 表示$\frac{\partial F}{\partial \mathbf{a}}$，并使用 $F_{\mathbf{ab}}$ 表示 $\frac{\partial^2 F}{\partial \mathbf{a}\,\partial \mathbf{ab}}$。$\delta\mathbf{a}$ 表示$\mathbf{a}$的变化。$(\cdot)$用作符号的占位符。

本文旨在针对给定的初始状态$\mathbf{x}$求解以下一般形式的最优控制问题（OCP）

$$\min_{\substack{\mathbf{x}_{1\cdots N}\\ \mathbf{u}_{0\cdots N-1}}} \quad l_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} l_k(\mathbf{x}_k, \mathbf{u}_k), \tag{1a}$$

$$\text{s.t.} \quad \forall k \in [0, N-1]: \ \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}) = \mathbf{0}, \tag{1b}$$

$$\mathbf{c}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \tag{1c}$$

$$\boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k) \preceq \mathbf{0}, \tag{1d}$$

$$\mathbf{s}(\mathbf{x}_k, \mathbf{x}_{k+1}) = \mathbf{0}, \tag{1e}$$

$$\boldsymbol{\phi}(\mathbf{x}_k, \mathbf{x}_{k+1}) \preceq \mathbf{0}, \tag{1f}$$

其中， N 为射击节点的数量，$\mathbf{x}_k$，$\mathbf{u}_k$ 分别为第 k阶段的状态和输入，$\mathbf{x}_N$ 为终端状态，l_N，l_k分别为终端代价和状态-输入运行代价；$\mathbf{f}$，$\mathbf{c}$和 $\mathbf{s}$分别表示隐式离散动力学（适用于一般积分器）、堆叠的状态-输入约束和仅状态约束。$\boldsymbol{\psi}$ 表示状态-输入不等式约束，$\boldsymbol{\phi}$ 表示仅状态不等式约束。

为简化符号，在本文其余部分中我们省略下标 k ，并使用三元组（$\mathbf{x}$，$\mathbf{u}$，$\mathbf{y}$）表示当前状态、输入以及第 k阶段的下一状态，即，对于 $k \in [0, N-1]$，$\mathbf{y}_k \equiv \mathbf{x}^{k+1}$；对于$_N$，$\mathbf{y}_{N-1} \equiv \mathbf{x}$。此外，除非另有说明，以下将使用逐阶段符号:

$\mathbf{h} = [\mathbf{f}^\top, \mathbf{s}^\top, \mathbf{c}^\top]^\top$	stacked equality constraints
$\boldsymbol{\lambda} = [\boldsymbol{\lambda}_f^\top, \boldsymbol{\lambda}_c^\top, \boldsymbol{\lambda}_s^\top]^\top$	multipliers of $\mathbf{h}$
$\mathbf{w} = [\mathbf{x}^\top, \mathbf{u}^\top, \mathbf{y}^\top]^\top$	stacked primal variables
$\mathbf{g} = [\boldsymbol{\phi}^\top, \boldsymbol{\psi}^\top]^\top$	stacked inequality constraints
$\boldsymbol{\nu} = [\boldsymbol{\nu}_{\boldsymbol{\phi}}^\top, \boldsymbol{\nu}_{\boldsymbol{\psi}}^\top]^\top$	multipliers of $\mathbf{g}$

本节介绍高效的基于投影的黎卡提递推，用于求解不含不等式的等式约束最优控制问题。随后在第 III 节中，我们说明如何用内点法处理不等式。基于投影的递归的整体流程如算法 1 所示。

A. 黎卡提递推

1) 中间阶段：贝尔曼最优性原理为每个阶段 $k \in [0, N-1]$定义价值函数 V:

$$V(\mathbf{x}) = \min_{\mathbf{u},\mathbf{y}} \ l(\mathbf{x}, \mathbf{u}) + V(\mathbf{y}), \ \text{s.t. } \mathbf{h}(\mathbf{w}) = \mathbf{0} \tag{2}$$

其本身可视为一个约束优化问题。与先前工作 [4], 类似，我们将式 (2) 的逐阶段拉格朗日函数 Q引入为

$$Q(\mathbf{w}, \boldsymbol{\lambda}) = l(\mathbf{x}, \mathbf{u}) + V(\mathbf{y}) + \boldsymbol{\lambda}^\top \mathbf{h}(\mathbf{w}) \tag{3}$$

在本工作中，我们采用类似高斯-牛顿法的 LQ 近似来处理 (3)， [4], 即：1）对所有约束进行线性近似；2）对代价和价值函数 V进行二次近似。相应的 KKT 条件可写成以下递归形式

$$\mathcal{K}\delta\boldsymbol{\xi} = -\mathbf{k}_0 - \mathbf{K}_0\delta\mathbf{x}, \tag{4a}$$

$$\mathcal{K} = \begin{bmatrix} Q_{\mathbf{uu}} & \mathbf{0} & \mathbf{h}_{\mathbf{u}}^\top \\ \mathbf{0} & Q_{\mathbf{yy}} & \mathbf{h}_{\mathbf{y}}^\top \\ \mathbf{h}_{\mathbf{u}} & \mathbf{h}_{\mathbf{y}} & \mathbf{0} \end{bmatrix}, \mathbf{k}_0 = \begin{bmatrix} Q_{\mathbf{u}}^\top \\ Q_{\mathbf{y}}^\top \\ \mathbf{h} \end{bmatrix}, \mathbf{K}_0 = \begin{bmatrix} Q_{\mathbf{ux}} \\ Q_{\mathbf{yx}} \\ \mathbf{h}_{\mathbf{x}} \end{bmatrix}. \tag{4b}$$

其中 $\boldsymbol{\xi} = [\mathbf{u}^\top,\ \mathbf{y}^\top,\ \boldsymbol{\lambda}^\top]^\top$。求解 (4)，其解为

$$\delta\boldsymbol{\xi} = \underbrace{-\mathcal{K}^{-1}\mathbf{k}_0}_{\mathbf{k}_{\boldsymbol{\xi}}} \underbrace{-\mathcal{K}^{-1}\mathbf{K}_0}_{\mathbf{K}_{\boldsymbol{\xi}}}\delta\mathbf{x} \tag{5}$$

其中$\mathbf{k}_{\boldsymbol{\xi}}$ 与$\mathbf{K}_{\boldsymbol{\xi}}$ 分别是 $\delta\boldsymbol{\xi}$的零阶和一阶灵敏度。特别地，$\mathbf{K}_{\boldsymbol{\xi}} = \delta\boldsymbol{\xi}/\delta\mathbf{x} \equiv \partial\boldsymbol{\xi}/\partial\mathbf{x}$。注意，在后文中，$\mathbf{k}/\mathbf{K}_{(\cdot)},_{(\cdot)} \neq 0$ 指$(\cdot)$的零阶/一阶敏感度。

2) 终端节点：终端状态的价值函数$\mathbf{x}_N$即为终端代价，即 $V_N(\mathbf{x}_N) \equiv l_N(\mathbf{x}_N)$。由于$\mathbf{y}_k \equiv \mathbf{x}^{k+1}$，可以看出 $V^k{}_{\mathbf{yy}} \equiv V^{k+1}{}_{\mathbf{xx}} V^{N-1}{}_{\mathbf{yy}} \equiv V^N{}_{\mathbf{xx}}$，$V$的雅可比矩阵亦然。随后可反向执行黎卡提递推：从终端节点 $V_\mathbf{N}$的导数出发，依次求解(4)以得到 $\delta\boldsymbol{\xi}$ 灵敏度；前一阶段的$V(\mathbf{y})$导数通过微分链式法则进行更新

$$\begin{aligned} V^{k-1}_{\mathbf{y}} &= V^k_{\mathbf{x}} = Q^k_{\mathbf{x}} + (\mathbf{Q}^\top_{\boldsymbol{\xi}}\mathbf{K}_{\boldsymbol{\xi}})^k = Q^k_{\mathbf{x}} + (\mathbf{k}_0^\top\mathbf{K}_{\boldsymbol{\xi}})^k \\ V^{k-1}_{\mathbf{yy}} &= V^k_{\mathbf{xx}} = Q^k_{\mathbf{xx}} + (\mathbf{Q}^\top_{\boldsymbol{\xi}\mathbf{x}}\mathbf{K}_{\boldsymbol{\xi}})^k = Q^k_{\mathbf{xx}} + (\mathbf{K}_0^\top\mathbf{K}_{\boldsymbol{\xi}})^k \end{aligned} \tag{6}$$

B. 基于投影的分解

根据 $\mathbf{h}$ 和 $\boldsymbol{\lambda}$ 的定义，KKT系统 (4) 可展开为

$$\begin{bmatrix} Q_{\mathbf{uu}} & \mathbf{0} & \mathbf{f}_{\mathbf{u}}^\top & \mathbf{0} & \mathbf{c}_{\mathbf{u}}^\top \\ \mathbf{0} & Q_{\mathbf{yy}} & \mathbf{f}_{\mathbf{y}}^\top & \mathbf{s}_{\mathbf{y}}^\top & \mathbf{0} \\ \mathbf{f}_{\mathbf{u}} & \mathbf{f}_{\mathbf{y}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{s}_{\mathbf{y}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{c}_{\mathbf{u}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u} \\ \delta\mathbf{y} \\ \delta\boldsymbol{\lambda}_f \\ \delta\boldsymbol{\lambda}_s \\ \delta\boldsymbol{\lambda}_c \end{bmatrix} = - \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{y}_0 \\ \mathbf{f}_0 \\ \mathbf{s}_0 \\ \mathbf{c}_0 \end{bmatrix}. \tag{7}$$

其中，右端项是 $\mathbf{k}_0$ 或 $\mathbf{K}_0$ 中的通用项。可以使用零空间投影来求解上述鞍点问题。对于机器人学中的常见积分器动力学，例如通用显式/隐式 Euler/RK4 积分器，$\mathbf{f}_\mathbf{y}$ 通常具有可逆性。因此，假设 $\mathbf{f}_\mathbf{y}$ 可逆，随后可引入以下内容，以便对 (7) 进行后续因式分解：

$$\mathbf{F}_0 := \mathbf{f}_\mathbf{y}^{-1}\mathbf{f}_0, \mathbf{F}_\mathbf{u} := \mathbf{f}_\mathbf{y}^{-1}\mathbf{f}_\mathbf{u}, \tag{8}$$

其中，左侧等式约束雅可比矩阵 $\mathbf{h}_{\mathbf{u},\mathbf{y}} := [\mathbf{h}_\mathbf{u}, \mathbf{h}_\mathbf{y}]$（(7) 的左下角）的零空间基 $[\mathbf{Z}_\mathbf{u}^\top, \mathbf{Z}_\mathbf{y}^\top]^\top$ 可推导如下：

$$\mathbf{Z}_\mathbf{u} = \mathcal{N}\left(\begin{bmatrix} -\mathbf{s}_\mathbf{y}\mathbf{F}_\mathbf{u} \\ \mathbf{c}_\mathbf{u} \end{bmatrix}\right), \mathbf{Z}_\mathbf{y} = -\mathbf{F}_\mathbf{u}\mathbf{Z}_\mathbf{u}. \tag{9}$$

其中，$\mathbf{Z}_\mathbf{u}, \mathbf{Z}_\mathbf{y}$ 分别是对应于约束雅可比矩阵 $\mathbf{u},\mathbf{y}$ 的零空间基 $\mathcal{N}$。随后，我们可以将 (7) 中的原始解 $\delta\mathbf{u}$ $\delta\mathbf{y}$ 拆分为

$$\begin{bmatrix} \delta\mathbf{u} \\ \delta\mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_\mathbf{u} \\ \mathbf{Z}_\mathbf{y} \end{bmatrix} \delta\mathbf{z} - \begin{bmatrix} \delta\hat{\mathbf{u}} \\ \delta\hat{\mathbf{y}} \end{bmatrix} \tag{10}$$

其中，$\delta\mathbf{z}$ 是（$\mathbf{u},\mathbf{y}$）在$\mathbf{h}$的零空间中对式（7）的解，且 $[\delta\hat{\mathbf{u}}^\top, \delta\hat{\mathbf{y}}^\top]^\top$：$\mathbf{h}_{\mathbf{u},\mathbf{y}}[\delta\hat{\mathbf{u}}^\top, \delta\hat{\mathbf{y}}^\top]^\top = \mathbf{h}_0$是 KKT 原始可行性条件的伪逆解。类似地，[27], 将式（10）代入式（7），并将所得结果左乘 $[\mathbf{Z}_u^\top, \mathbf{Z}_y^\top]$后，可以将式（7）投影到 $\mathbf{h}$ 的零空间中，得到

$$Q_{\mathbf{zz}}\delta\mathbf{z} = -\mathbf{z}_0, where \tag{11a}$$
$$Q_{\mathbf{zz}} := \mathbf{Z}_\mathbf{u}^\top Q_{\mathbf{uu}}\mathbf{Z}_\mathbf{u} + \mathbf{Z}_\mathbf{y}^\top Q_{\mathbf{yy}}\mathbf{Z}_y \tag{11b}$$
$$\mathbf{z}_0 := \mathbf{Z}_\mathbf{u}^\top\bar{\mathbf{u}}_0 + \mathbf{Z}_\mathbf{y}^\top\bar{\mathbf{y}}_0 \tag{11c}$$
$$\bar{\mathbf{u}}_0 := \mathbf{u}_0 - Q_{\mathbf{uu}}\delta\hat{\mathbf{u}} \tag{11d}$$
$$\bar{\mathbf{y}}_0 := \mathbf{y}_0 - Q_{\mathbf{yy}}\delta\hat{\mathbf{y}} \tag{11e}$$

该约化系统适用于各种受约束动力学，尤其是涉及接触动力学的情况。$\hat{}$ 当不涉及 $\mathbf{s},\mathbf{c}$ 时，$\mathbf{Z}_\mathbf{u} \equiv \mathbf{I}$ 和 $\mathbf{Z}_\mathbf{y} \equiv -\mathbf{F}_\mathbf{u}$, $\delta(\cdot) = 0$，构成经典黎卡提递推，例如 [2] 中所述。

通过对 $\delta\mathbf{z}$ 求解（11），我们可以根据（10）恢复完整的原始步，并在代入 $\delta\mathbf{u}$ $\delta\mathbf{y}$ 后，通过求解 KKT驻点条件得到对偶步：

$$\mathbf{h}_{\mathbf{u},\mathbf{y}}^\top\delta\boldsymbol{\lambda} = -\begin{bmatrix} \mathbf{u}_0 + Q_{\mathbf{uu}}\delta\mathbf{u} \\ \mathbf{y}_0 + Q_{\mathbf{yy}}\delta\mathbf{y} \end{bmatrix}. \tag{12}$$

备注1 （秩亏约束雅可比矩阵）我们使用 *Eigen [28]* 全主元 *LU* 分解来计算（9）以及原始步和对偶步的伪逆解。该方法基于阈值判定非零主元，因此能够相对稳健地应对较差的数值条件。实际应用中，尤其是在存在接触的情况下，可以使用 *[29]* 中这类经过专门正则化的分解器，以获得有界近似解和零空间投影算子。当约束雅可比矩阵高度秩亏时，正则化可能不足以获得有效的牛顿步，求解器也可能收敛到不可行驻点。在这种情况下，应采用全局化策略来摆脱此类情形 *[22]*。

备注2 （递归零空间投影）（9）中零空间基的推导受到经典的基于层级二次规划的全身控制器启发 *[30]*。如果投影后的$\mathbf{h}_u$ 的某些部分可逆，例如存在接触的提升逆动力学情形，则可以进一步扩展 $\bar{}$，此时可以使用德拉瑟斯矩阵的逆进一步推导解析零空间投影算子。

C. 价值函数导数传播

直接通过式 (6) 更新 V 导数，需要计算完整的原始变量和对偶变量一阶灵敏度，导致递归阶段的计算成本高昂。然而，利用投影可以将残差-敏感度乘积（RSP）$\mathbf{k}_0^\top\mathbf{K}_{\boldsymbol{\xi}}$，$\mathbf{K}_0^\top\mathbf{K}_{\boldsymbol{\xi}}$ 转换为计算效率更高的形式。

令上标 l 表示与 RSP 左侧相关的量，r 表示与$\mathbf{K}_{\boldsymbol{\xi}}$ 相关的量。例如，$\mathbf{u}_0^l$ 是$\mathbf{k}_0$ 中针对 $V_\mathbf{x}$的 $Q_\mathbf{u}^\top$；$\delta\hat{\mathbf{u}}^r, \delta\hat{\mathbf{y}}^r$ 是右端项为$\mathbf{K}_0$时的伪逆解。根据 (12) 和$\mathbf{h}_{\mathbf{u},\mathbf{y}}[\delta\hat{\mathbf{u}}^\top, \delta\hat{\mathbf{y}}^\top]^\top = \mathbf{h}_0$,，RSP 的对偶部分可改写为

$$\mathbf{h}_0^{l,\top}\delta\boldsymbol{\lambda}^r = -\begin{bmatrix} \delta\hat{\mathbf{u}}^l \\ \delta\hat{\mathbf{y}}^l \end{bmatrix}^\top \begin{bmatrix} \mathbf{u}_0^r + Q_{\mathbf{uu}}\delta\mathbf{u}^r \\ \mathbf{y}_0^r + Q_{\mathbf{yy}}\delta\mathbf{y}^r \end{bmatrix} \tag{13}$$

算法1： SQP的单次迭代

数据： $\mathbf{x}$的初始轨迹，$\mathbf{u}$，代价函数 l_k，l_N，动力学$\mathbf{f}$、约束$\mathbf{c}$，$\mathbf{s}$ ψ，ϕ及其乘子

针对每个节点并行执行 ▷预求解步骤
- 更新最优控制问题的线性二次近似。
- ˆ零空间分解以计算 (9)，$\delta(\cdot)$ 中的 (10) 以及 (11)、(14) 中可预先计算的部分。
- 内点法 (19) 对雅可比矩阵/海森矩阵的修改。

对于 $k = N-1$ **到** 0，**执行** ▷反向递推
- $V_{\mathbf{y}} \leftarrow V^{k+1}{}_{\mathbf{x}}$，$V_{\mathbf{yy}} \leftarrow V^{k+1}_{\mathbf{xx}}$。
- $Q_{\mathbf{y}} \mathrel{+}= V_{\mathbf{y}}$，$Q_{\mathbf{yy}} \mathrel{+}= V_{\mathbf{yy}}$。
- 分别更新 $Q_{\mathbf{zz}}$ 和 $\bar{}\mathbf{y}_0$，用于 $\mathbf{k}_0$,$\mathbf{K}_0$ 。
- 求解零空间 KKT 系统 (11)，得到 $\mathbf{K_z}$。
- 使用 (14) 更新 $V^k_{\mathbf{xx}}$、$V^k{}_{\mathbf{x}}$。

对每个节点并行执行 ▷求解后步骤
- 计算 $\mathbf{k_z}$，$\mathbf{k_y}$，$\mathbf{K_y}$。

对于 $k = 0$ **到** $N-1$，**执行**
- $\delta\mathbf{x}_{k+1} \leftarrow \delta\mathbf{y}_k = \mathbf{k}^k_{\mathbf{y}} + \mathbf{K}^k_{\mathbf{y}}\delta\mathbf{x}^k$。

并行处理每个节点 ▷滚动计算后步骤
- $\delta\mathbf{z} = \mathbf{k_z} + \mathbf{K_z}\delta\mathbf{x}$。
- $\delta\mathbf{u} = \mathbf{Z_u}\delta\mathbf{z} - \delta\hat{}\mathbf{u}$.
- 更新 $\boldsymbol{\lambda}$ （如有必要）。

将其与 RSP 的原始部分合并，并根据 (10) 将$\delta\mathbf{u}^r, \delta\mathbf{y}^r$ 拆分为零空间步骤和伪逆步骤后，对于当前阶段，我们有 $\forall(\cdot) \in \{\mathbf{x},\mathbf{xx}\}$,

$$V_{(\cdot)} = Q_{(\cdot)} + \mathbf{z}_0^{l,\top}\delta\mathbf{z}^r - \bar{\mathbf{u}}_0^{l,\top}\delta\hat{\mathbf{u}}^r - \bar{\mathbf{y}}_0^{l,\top}\delta\hat{\mathbf{y}}^r - \delta\hat{\mathbf{u}}^{l,\top}\mathbf{u}_0^r - \delta\hat{\mathbf{y}}^{l,\top}\mathbf{y}_0^r. \tag{14}$$

它是前一阶段的 $V_{\mathbf{y/yy}}$，如式 (6) 所示。这样，在递归过程中，我们只需计算零空间中的灵敏度。请注意，与值函数导数无关的部分可以预先并行计算。

D. 线性前向展开

为构造所有决策变量的完整牛顿步，首先对 $\delta\mathbf{y}$ 使用 $\mathbf{k_y}$，$\mathbf{K_y}$进行线性前向展开。其余原始步和对偶步随后利用更新后的 $\delta\mathbf{x}$ （来自 $\delta\mathbf{y}$）并行构造。请注意，此处未采用微分动态规划中广泛使用的非线性展开，因为线性展开能够重构真实的牛顿步，并具有更可靠的理论收敛保证 [5]。

III. 保持稀疏性的约束处理

本节介绍一种保持稀疏性、可并行化的正则化内点法实现，用于处理不等式约束和秩亏等式约束。

A. 正则化原始-对偶内点法

将等式约束 OCP 扩展到包含不等式约束 $\mathbf{g}$ 的完整问题 (1)，可得到新的˜拉格朗日函数 Q 及其相应的 LQ 近似 KKT 条件

$$\tilde{Q}(\mathbf{w},\boldsymbol{\lambda},\boldsymbol{\nu}) := Q(\mathbf{w},\boldsymbol{\lambda}) + \boldsymbol{\nu}^\top\mathbf{g}(\mathbf{w}), \tag{15a}$$

$$\begin{bmatrix}\tilde{\mathcal{K}} & \mathbf{g}_{\boldsymbol{\xi}}^\top \\ \mathbf{g}_{\boldsymbol{\xi}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\xi}\\ \delta\boldsymbol{\nu}\end{bmatrix} = -\begin{bmatrix}\tilde{\mathbf{k}}_0\\ \mathbf{g}\end{bmatrix} - \begin{bmatrix}\tilde{\mathbf{K}}_0\\ \mathbf{g_x}\end{bmatrix}, \tag{15b}$$

$$\mathbf{h} + \mathbf{h_w}\delta\mathbf{w} = \mathbf{0}, \tag{15c}$$

$$\mathbf{g} + \mathbf{g_w}\delta\mathbf{w} \le \mathbf{0}, \tag{15d}$$

$$\boldsymbol{\nu} \odot \mathbf{g} = \mathbf{0}, \tag{15e}$$

˜ ˜ ˜其中 $\boldsymbol{\nu}$是$\mathbf{g}$对应的堆叠不等式乘子，$\mathcal{K}$、$\mathbf{k}_0$,$\mathbf{K}_0$˜的定义与 (4) 类似，但对应于 Q；$\odot$ 表示逐元素乘积。注意，$\mathbf{g_\lambda} = \mathbf{0}$。(15) 可通过原始-对偶内点法转化为等式约束问题：在 (15d) 中引入松弛变量$\mathbf{t}$，并通过障碍参数 μ放宽互补松弛条件 (15e)，即扰动 *KKT* 条件

$$\mathbf{r}_g := \mathbf{g} + \mathbf{t}, \qquad \mathbf{r}_s := \boldsymbol{\nu}\odot\mathbf{t} - \mu\mathbf{1}, \tag{16a}$$

$$\mathbf{r}_g + \mathbf{g_w}\delta\mathbf{w} = \mathbf{0}, \quad \mathbf{r}_s = \mathbf{0}, \qquad \boldsymbol{\nu},\mathbf{t} \succeq \mathbf{0}, \tag{16b}$$

由于 $\boldsymbol{\nu}\odot\mathbf{t}$，这是一个双线性系统。对 $\boldsymbol{\nu}$,$\mathbf{t}$ 进行线性化后，可得到如下新的 KKT系统

$$\begin{bmatrix}\tilde{\mathcal{K}} & \mathbf{g}_{\boldsymbol{\xi}}\top & \mathbf{0}\\ \mathbf{g}_{\boldsymbol{\xi}} & -\rho\mathbf{I} & \mathbf{I}\\ \mathbf{0} & \mathbf{T} & \mathbf{N}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\xi}\\ \delta\boldsymbol{\nu}\\ \delta\mathbf{t}\end{bmatrix} = -\begin{bmatrix}\tilde{\mathbf{k}}_0\\ \mathbf{r}_g\\ \mathbf{r}_s\end{bmatrix} - \begin{bmatrix}\tilde{\mathbf{K}}_0\\ \mathbf{g_x}\\ \mathbf{0}\end{bmatrix}\delta\mathbf{x}, \tag{17}$$

其中 $\mathbf{T}$ = diag($\mathbf{t}$)，$\mathbf{N}$ = diag($\boldsymbol{\nu}$)，ρ 为乘子正则化参数，用于避免$\mathbf{t} \to 0$ 时出现病态条件。随后，我们可以利用式（17）的结构，将求解过程拆分为一个 $\boldsymbol{\xi}$-步和一个 ($\boldsymbol{\nu}$,$\mathbf{t}$)-步：

$$\delta\boldsymbol{\xi} = \hat{\tilde{\mathcal{K}}}^{-1}(-\hat{\tilde{\mathbf{k}}}_0 - \hat{\tilde{\mathbf{K}}}_0\delta\mathbf{x}) \tag{18a}$$

$$\delta\boldsymbol{\nu} = \mathbf{T}_\rho^{-1}[-\mathbf{r}_s - \mathbf{N}(\mathbf{r}_g + \mathbf{g_w}\delta\mathbf{w})] \tag{18b}$$

$$\delta\mathbf{t} = -\mathbf{r}_g - \mathbf{g_w}\delta\mathbf{w} + \rho\delta\boldsymbol{\nu} \tag{18c}$$

其中˜(·)表示因修改雅可比矩阵和海森矩阵而产生的修改量，这些修改量以增量值表示（$\mathcal{W}$:= {$\mathbf{x}$，$\mathbf{u}$，$\mathbf{y}$}）：

$$\forall(\cdot) \in \mathcal{W} : \Delta\tilde{Q}_{(\cdot)} = [\mathbf{T}_\rho^{-1}\mathbf{N}\mathbf{g} + \mu\mathbf{T}_\rho^{-1}]^\top\mathbf{g}_{(\cdot)}, \tag{19a}$$

$$\forall(\cdot) \in \mathcal{W}\times\mathcal{W} : \Delta\tilde{Q}_{(\cdot)} = \mathbf{g}_{(\cdot)}^\top\mathbf{T}_\rho^{-1}\mathbf{N}\mathbf{g}_{(\cdot)}. \tag{19b}$$

由于$\mathbf{T}_\rho^{-1}\mathbf{N}$ 具有对角性质，且$\mathbf{g}$的定义如此，$Q_{\mathbf{uy}}$ 仍将为零。因此，(7)中的稀疏性得以保留。采用(18)中的拆分后，在顺序向前展开之后，可以并行计算不同射击节点的($\boldsymbol{\nu}$，$\mathbf{t}$)步，而无需直接求解整个KKT系统(17)。

定理 1 假设$\mathbf{t}$ $\boldsymbol{\nu}$非负，则正则化项可保证海森矩阵修改(19)有界。此外，假设$\mathbf{r}_s \approx \mathbf{0}$，雅可比矩阵修改(19a)在数值上不会发生爆炸。

证明： 由$\mathbf{t}$ $\boldsymbol{\nu}$的非负性可推知其有界：

$$\mathbf{T}_\rho^{-1}\mathbf{N} = (\mathbf{T}\mathbf{N}^{-1} + \rho\mathbf{I})^{-1} \prec \rho^{-1}\mathbf{I} \tag{20}$$

其中，$\preceq$表示逐元素小于等于。根据$\mathbf{r}_s \approx \mathbf{0}$，我们有$\mathbf{TN} \approx \mu\mathbf{I}$。雅可比矩阵修改 (19a) 的幅值有界：

$$||\mathbf{T}_\rho^{-1}\mathbf{N}\mathbf{g}+\mu\mathbf{T}_\rho^{-1}|| \approx ||\mathbf{T}_\rho^{-1}\mathbf{N}(\mathbf{g}+\mathbf{t})|| \leq \rho^{-1}||\mathbf{g}+\mathbf{t}||. \quad (21)$$

■

海森矩阵的有界性避免了原始朴素内点法的数值病态问题。只需付出极低的 ρ （在我们的设置中，默认值为 1e-8），便能大幅提高数值鲁棒性。定理 1 表明，当不等式约束变为活跃约束时，内点法的收敛特性将类似于传统的基于惩罚的方法处理等式约束的情形。

B. 自适应障碍策略

我们实现了 [15] 中的 MP-C 变体，以自适应地调整障碍参数 μ。在预测步中，对 (7) 进行分解，并使用 $\mu = 0$ 计算原始步和内点法变量。随后计算新的 μ：

$$\sigma = \text{clip}\left(\frac{\sum(\boldsymbol{\nu}+\delta\boldsymbol{\nu})^\top(\mathbf{t}+\delta\mathbf{t})}{\sum\boldsymbol{\nu}^\top\mathbf{t}}, 0, 1\right)^3, \mu \leftarrow \sigma\frac{\sum\boldsymbol{\nu}^\top\mathbf{t}}{n_{\text{ipm}}}, \quad (22)$$

其中，σ 是取值范围为 $[0,1]$ 的中心化参数，n_{ipm} 是内点法约束的总维度。在校正步中，使用更新后的 μ 和校正项 $\delta\boldsymbol{\nu}\odot\delta\mathbf{t}$ 修改拉格朗日雅可比矩阵。随后再次执行黎卡提递推，利用预测步中先前的 KKT 分解更新值函数雅可比矩阵 $V_{\mathbf{x}}$。

注意，在预测步和校正步中，都必须计算步长上界，以确保松弛变量和乘子为正：

$$\alpha^{\max} = \max\{\alpha \in [0,1] | \mathbf{t} + \alpha\delta\mathbf{t} \geq (1-\tau)\mathbf{t}\}, \quad (23a)$$

$$\alpha_{\boldsymbol{\nu}}^{\max} = \max\{\alpha_{\boldsymbol{\nu}} \in [0,1] | \boldsymbol{\nu} + \alpha_{\boldsymbol{\nu}}\delta\boldsymbol{\nu} \geq (1-\tau)\boldsymbol{\nu}\}, \quad (23b)$$

其中，$\alpha^{\max}$、$\alpha_{\boldsymbol{\nu}}^{\max}$分别是不等式对偶步长上界和原始步长上界（包括$\mathbf{t}$）（$\tau = 0.995$ 在我们的设置中）。我们还实现了一种保护机制，仅当试探互补性不增加时，才接受校正项 [12]。

C. 迭代改进

当采用 IPM 的 SQP 接近收敛时，其较差的数值条件可能导致驻点条件中的误差相对较大，并减缓其向高精度解的收敛速度。如有必要，可以采用迭代改进，通过将右端项$\mathbf{k}_0$ 设为驻定性残差，并执行类似于校正步的递归，来减小误差。

D. 固定步长回溯线搜索

与 IPOPT [22], 类似，我们的求解器对原始步和等式对偶步采用 $\alpha \leq \alpha^{\max}$，仅对不等式对偶步采用 $\alpha_{\boldsymbol{\nu}} \leq \alpha_{\boldsymbol{\nu}}^{\max}$。我们采用简单的固定步长回溯线搜索：当原始残差和对偶残差均未取得进展时，以固定步长 $\Delta\alpha$ 减小 α，最多尝试 n_{ls} 次：$= \alpha^{\max}/n_{\text{ls}}$。如果求解器找不到任何下降步时，为了摆脱这种局部驻点，求解器会：1）将式（23）中的 $\alpha^{\max}$ 缩放至不高于预先设定的最小值 $\alpha^{\min}$；2）重置 $\mu = 1$，并重新初始化内点法松弛变量和乘子。实践中，我们发现 $\alpha^{\min} = 0.01$ 是一个稳健的选择。

表 II 求解器及其特性比较

求解器	内循环[a]	参数更新方案
Hippo/fatrop	迭代改进（线性）	每次外循环
acados	hpipm 二次规划循环（quad）	每次外循环重置
aligator(p)[b]	迭代改进（lin）	如果原始残差为 $>$ tol
mim 求解器	ADMM 步（线性）	每固定次数的更新

[a]'lin'表示仅更新零阶灵敏度。'quad'表示计算线性二次子问题的新牛顿步。[b]aligator 的串行版本没有针对序列二次规划步的内循环

E. 基于内点法的等式约束处理

当等式约束可能如备注1所述存在秩亏时，基于投影的处理可能导致收敛方面的严重困难。与*acados*类似，可以将此类等式约束转换为边界为零的盒约束，即$\mathbf{h} + \mathbf{t}_p = \mathbf{0}, \mathbf{t}_n - \mathbf{h} = \mathbf{0}, \mathbf{t}_p, \mathbf{t}_n \succeq 0$，其中$\mathbf{t}_p, \mathbf{t}_n$ 表示双边松弛变量，内点法会同时将它们推至零以确保可行性。

F. 实现

该优化器采用 C++ 开发，基于*Eigen*[28]和*BLASFEO* [31]。此外，它还提供了一个代码生成流程，可基于*CasADi* [32]和*Pinocchio* [33]进行高度优化的函数导数计算。

IV. 基准测试与讨论

在本节中，我们将*Hippo* 与先前工作*fatrop* [26]（commit 9a290fd）、*acados* [7]（commit 6111f01）、*aligator* [4], [6]（commit af2826）、*mim* 求解器 [5]（commit bd8852a）在两个代表性示例上进行基准测试，以展示我们求解器的鲁棒性：1）UR5到达任务，用于测试其处理秩亏硬约束的能力；2）Go2运动，用于测试其在优化驱动的任务与运动规划中常用的多接触场景下、间歇接触条件下的收敛性 [34], [23], [25]。需要注意的是，这些设置不同于常见的模型预测控制设置，后者通常更简单，具有良好的热启动，且不要求精确求解（例如强调每个二次规划计时的实时迭代方案 [5],[7]）。在所有实验中，我们仔细调节了*aligator* 和*mim* 求解器 的参数，使其能够求解尽可能多的问题。表II提供了各求解器的更多细节。若 L_∞KKT残差（包括约束违反、驻定性和互补性）低于绝对容差，则将该测试标记为“成功”。为确保公平比较，所有被测求解器均使用线性前向展开和线搜索；问题建模采用前向全身动力学。对于每项测试，所有求解器的代价与初始化均保持统一。硬件平台为AMD Ryzen 9 7945HX。所有求解器均使用-*march=native*编译。

表 III UR5 到达基准测试的求解器参数

求解器	迭代次数	内层迭代次数	内层容差	参数[a]
Hippo	100	2	1e-10（绝对）	1
acados	100	50	1e-3（绝对误差）	1
fatrop	100	5	1e-8（绝对误差）	1
aligator（复数）	400	0	变化	0.1
mim_求解器(1)/(2)	400	5(15)/20(60)[b]	1e-3（绝对值）	0.1

[a]内点法的障碍参数和增广拉格朗日法/交替方向乘子法的惩罚参数。
[b](..) 表示用于更新惩罚参数的固定内层步数。

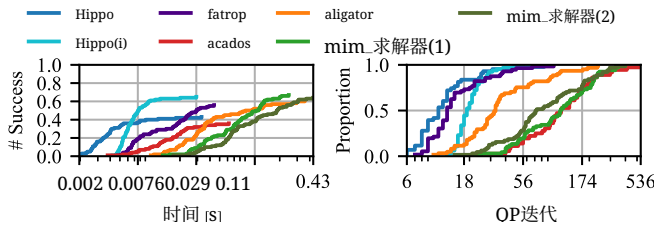


图 1. UR5 随机 SE(3) 到达基准测试结果。左图显示成功次数与墙钟时间的关系。右图显示测试的 QP 迭代次数分布。

Hippo(i) 表示按第III-E节处理非动力学等式约束的测试。需要注意的是，由于*fatrop*的实现限制，后文报告的其总墙钟时间将是仅求解器计时与函数评估计时/线程数之和，以便进行比较。然而，在实际中，由于求解器缺乏并行化且函数评估效率低下，*fatrop*的墙钟时间会高得多。

A. UR5 随机到达

在本实验中，我们采用[4],中的 UR5 伸手任务，即求解 100 个随机目标末端执行器 $SE(3)$在机器人基座坐标系原点周围半球内的放置问题：

$$\mathbf{M}^{\text{des}} \sim \mathcal{U}([-0.6, 0.6]^2 \times [0, 0.6] \times SO(3)). \tag{24}$$

其中，$\mathcal{U}$ 表示均匀分布。每个目标都与一个从$\mathcal{U}[-1, 1]$中均匀采样的随机初始配置配对，如 [4]所示。目标作为终端状态的硬约束$\mathbf{m}$加入，即：

$$\mathbf{m} := \log[\mathbf{M}^{\text{des}} \ominus \mathbf{M}_N] = \mathbf{0}_6, \tag{25}$$

其中$\mathbf{M}_N \in SE(3)$ 是末端执行器的最终位姿，$\ominus$ 表示 $SE(3)$ 差，log 是从 $SE(3)$ 到 $\mathfrak{se}(3)$ 的对数映射。轨迹时域为 $N = 50$，时间步长为 20ms。包含状态和力矩限制。每项测试均使用 4 个线程运行。求解器设置见表 III，结果见图 1。终止绝对容差为 1e-3。我们使用了 *aligator* 的串行版本（标记为*(s)*），因为其并行版本（记为*aligator(p)*）存在数值问题。对于*fatrop*，我们使用其代码生成的*CasADiOptiStack* 接口的编译版本，以最大限度地提升性能。对于增广拉格朗日法基线，我们采用了较高的迭代上限 [4] ，因为它们通常需要更多迭代才能收敛。使用了两组参数

表 IV UR5 示例的统计数据

	Hippo	Hippo(i)	fatrop	acados	aligator	mim_求解器
#	43	65	56	36/81[a]	61	(1):67;(2):64
t[b]	0.33	0.32	0.86	0.13	1.02	(1):0.61;(2):1.1

[a]81 个解中，只有 36 个解的松弛值 (≤1e-3) 可接受。
[b]"#"：已求解问题的数量；"**t**"：每次二次规划的平均耗时（毫秒）

表 V 增广拉格朗日法求解器参数：GO2运动基准

求解器	SQP迭代次数	内层迭代次数	内层容差	参数[a]
aligator(p)	200	5	变化的	0.1
mim_求解器	200	100(50)[b]	1e-3（绝对值）	0.1

[a]内点法的障碍参数以及增广拉格朗日法/交替方向乘子法的惩罚参数
[b](..) 用于更新惩罚参数的固定步数

如表III 所示，使用了*mim* 求解器 。如图 1 和表 IV 所示，由于$\mathbf{m}$可能存在雅可比矩阵秩亏，*Hippo(i)* 解决的问题比 *Hippo* 更多，但迭代次数也更多。得益于*ipopt*的全局化方案，*fatrop* 解决的测试数量超过了*Hippo*，但仍少于 *Hippo(i)*。*acados* 只有在对松弛变量施加强惩罚时才能将原始残差降至容差以下，因此容易陷入局部不可行极小值。*aligator* 和*mim* 求解器 解决的问题数量与*Hippo(i)* 相近，但迭代次数更多。- 这表明，尽管所需迭代次数多于纯投影，我们基于正则化内点法的等式约束处理仍具有更好的鲁棒性和效率，因为 μ 必须逐步降至零。

值得注意的是，由于采用了通用零空间投影，*Hippo* 求解二次规划子问题的速度并非最快。相比之下，*fatrop* 以及*acados*的*HPIPM* 核心充分利用了*BLASFEO* 的高效性能；*aligator* 和*mim solver*利用*pinocchio* 计算接触力相对于运动学状态的导数，从而直接消除接触力。*hippo*的速度优势通常源于其较少的 QP 迭代次数：它能以更少的迭代收敛，且无需精确求解子问题。*fatrop*也表现出类似的速度优势。相反，*mim solver* 和*acados* 的每个序列二次规划步都可能需要大量内层迭代，却收效甚微，尤其是在子问题已经病态时，例如图 1 中*mim solver(1)/(2)*的数据。正如备注1中所述，对于适定性不良的最优控制问题，通过全局化采取原始步以摆脱局部不可行点，比精确求解线性二次子问题更有效。根据我们的实践，*acados* 和*mim solver* 对其内循环设置非常敏感。

B. Go2 两步运动

在本实验中，我们构建了一个通用的四足机器人运动问题：机器人朝两个随机采样的参考基座 x-y 位置移动，这些位置位于以机器人初始位置（即原点）为中心的矩形区域内。我们测试了两种步态（小跑和跳跃），并将它们分别作为显式的阶段接触约束加入最优控制问题。终止绝对容差为 1e-3。我们提供了两个难度级别的随机采样参考位置。令$\mathbf{r}_1$,$\mathbf{r}_2$ 分别表示第一步和最后一步之后的期望基座位置，

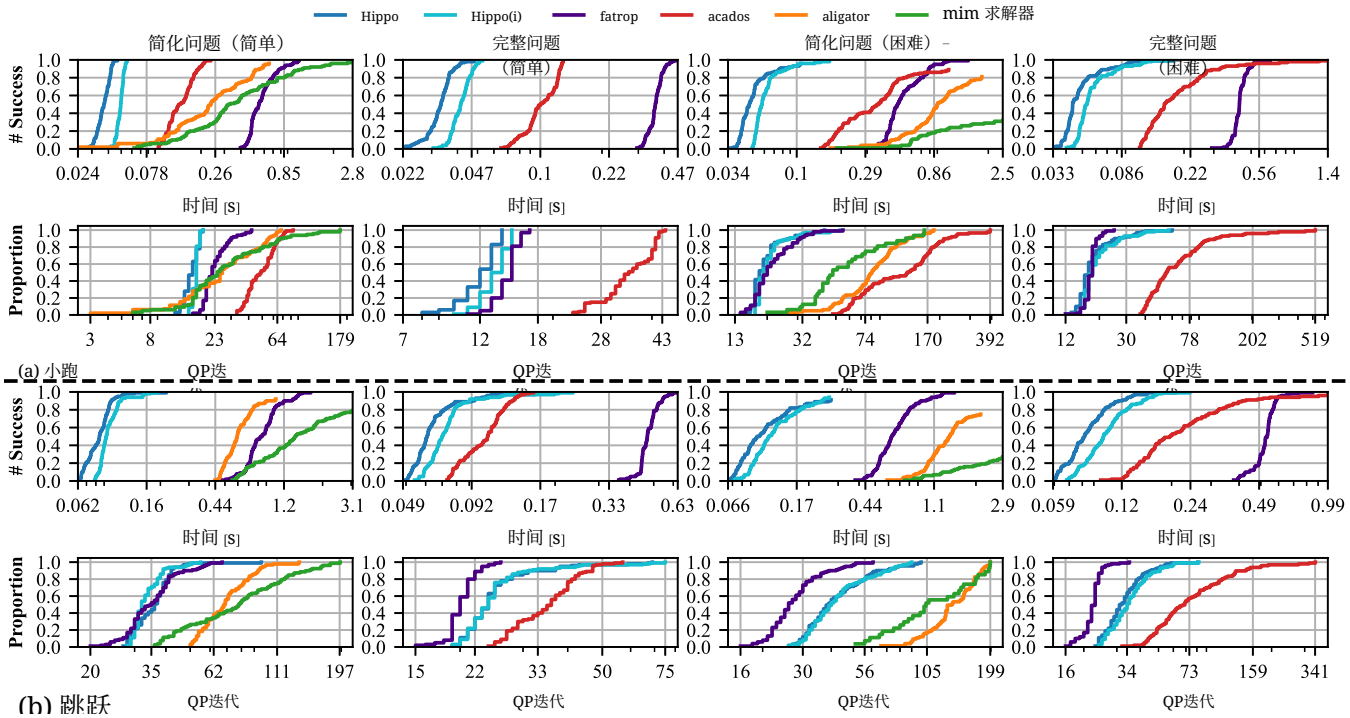


图 2. Go2 两步运动基准测试结果。每种设置均进行了 100 次随机试验。某种设置缺少求解器数据，表示该求解器在该设置下的所有测试均失败。

表 VI Go2 示例统计数据

	RS[a]		FS		RH		FH	
trot	#	t	#	t	#	t	#	t
Hippo	100	2.35	100	2.75	96	2.55	100	2.67
Hippo(i)	100	2.93	100	3.13	99	3.01	100	2.95
fatrop	100	24.9	100	25.0	100	24.9	100	23.9
acados	100	2.86	100	2.99	89	2.60	100	2.81
aligator	96	8.68	0	-	81	9.95	0	-
mim 求解器	98	13.37	0	-	32	14.96	0	-
跳跃	#	t	#	t	#	t	#	t
Hippo	100	2.35	100	2.57	91	2.46	100	2.57
Hippo(i)	100	2.89	100	2.93	94	2.86	100	2.93
fatrop	100	24.3	100	23.6	100	24.9	100	24.4
acados	0	-	100	2.97	0	-	97	2.86
aligator	92	8.58	0	-	75	8.97	0	-
mim 求解器	79	15.42	0	-	27	14.29	0	-

[a]R：简化版；S：简易版；F：完整版；H：困难版。'#'：已求解问题的数量。't'：每个二次规划的平均耗时（毫秒）。

简单版本从一个较小区域内采样$\mathbf{r}_2$，然后将其中点作为$\mathbf{r}_1$，，即，

$$\mathbf{r}_1 = \frac{\mathbf{r}_2}{2}, \mathbf{r}_2 \sim \mathcal{U}[-0.5, 0.5] \times \mathcal{U}[-0.5, 0.5]. \quad (26)$$

困难版从更大的区域采样，生成两个互不相关的$\mathbf{r}_1$,$\mathbf{r}_2$，，从而产生更大的运动，即：

$$\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}[-1, 1] \times \mathcal{U}[-1, 1]. \quad (27)$$

此示例的正则化代价为基座状态赋予了较高权重，强调对位置参考的跟踪。模型包含关节和力矩限制。与 [23] 类似，此处未提供足部参考。我们测试了两种约束设置：1）“完整”，包含摩擦锥和位置级接触约束 $[v_{c,x}, v_{c,y}, v_{c,z} + 100 z_c]^\top = \mathbf{0}$，其中$v_{c,\ (\cdot)}$为线性接触速度，$z_c$为足部高度；2）“简化”，不包含摩擦锥，仅包含零接触速度约束，同时对 z_c 施加惩罚。ALM 求解器的参数记录在表 V 中；IPM 求解器的参数与表 III 相同。我们尽可能使用直接的 *CasADiOptiStack* 接口测试了 *fatrop*，因为无法编译生成的超大代码文件。该场景的结果见图 2 和表 VI。

Hippo 和*Hippo(i)* 解决了几乎所有问题，唯独简化问题（困难）设置除外。失败源于第 III-D 节中简单的全局化方案，该方案在准收敛时会导致步长过小。相反，*fatrop*采用了系统化的*IPOPT* 全局化方法，解决了所有测试问题。*aca-dos* 解决的问题较少，并且完全未能通过简化跳跃测试。这些观察结果表明，与约束相比，采用惩罚项的简化最优控制问题容易出现退化，因此需要系统化的全局化方法。这解释了*Hippo* 性能下降以及*acados*的灾难性失败。此外，此类最优控制问题的局部极小值可能不可行，如第 IV-A 节表 IV 中*acados* 的结果所示，因此对于 [23], [34], [35] 等基于约束的规划而言，用处较小。在严格受约束的完整设置中，*Hippo/Hippo(i)* 展现出最快的速度，且鲁棒性与*fatrop*相当；即使在跳跃测试中，*fatrop* 的 QP 迭代次数更少，情况也是如此。这表明，我们的求解器在处理受约束的最优控制问题时，并行性和全局化方法是有效的。相比之下，尽管拥有更高的迭代次数上限，*aligator* 和*mim* 求解器 也只能解决部分简化测试问题，并且在所有完整设置中均告失败。这些结果再次印证了第 I-A 节和第 IV-A 节所述的结论：与增广拉格朗日法相比，内点法具有更优的鲁棒性和收敛性。

V. 结论

本文介绍了*Hippo*，一种简洁的内点法和基于投影的序列二次规划实现。*Hippo*在困难示例基准测试中展现出稳健的收敛性，能够与现有最先进的求解器相媲美，且无需耗费大量精力进行求解器参数调优。作为一种通用的约束轨迹优化器，*Hippo*可用于模仿学习的数据生成和优化驱动的任务与运动规划。未来工作中，可将更全面的全局化策略集成到求解器框架中，例如*IPOPT* 可行性恢复阶段和过滤线搜索 [22], ，以进一步增强其鲁棒性。

参考文献

[1] J. Albersmeyer 和 M. Diehl，《提升牛顿法及其在优化中的应用》，*SIAM* 优化期刊，第20卷第3期，第1655–1684页，2010年。[2] C. Mastalli、R. Budhiraja、W. Merkt、G. Saurel、B. Hammoud、M. Naveau、J. Carpentier、L. Righetti、S. Vijayakumar 和 N. Mansard，《Crocoddyl：一种高效且通用的多接触最优控制框架》，载于*IEEE* 国际机器人与自动化会议 *(ICRA)*，2020年。[3] J.-P. Sleiman、F. Farshidian 和 M. Hutter，《基于连续时间 DDP 的模型预测控制中的约束处理》，载于*2021*年 *IEEE* 国际机器人与自动化会议 *(ICRA)*，第8209–8215页，2021年。[4] W. Jallet、A. Bambade、E. Arlaud、S. El-Kazdadi、N. Mansard 和 J. Carpentier，《ProxDDP：近端约束轨迹优化》，*IEEE* 机器人学汇刊，第41卷，第2605–2624页，2025年。[5] A. Jordana、S. Kleff、A. Meduri、J. Carpentier、N. Mansard 和 L. Righetti，《用于模型预测控制的利用结构的序列二次规划》，*IEEE* 机器人学汇刊，第41卷，第4960–4974页，2025年。[6] W. Jallet、E. Dantec、E. Arlaud、N. Mansard 和 J. Carpentier，《用于实时约束模型预测控制的并行近端线性二次方法》，载于机器人科学与系统 *XX*，2024年6月。[7] R. Verschueren、G. Frison、D. Kouzoupis、J. Frey、N. van Duijkeren、A. Zanelli、B. Novoselnik、T. Albin、R. Quirynen 和 M. Diehl，《acados——用于快速嵌入式最优控制的模块化开源框架》，数学规划计算，2021年。[8] S. Katayama 和 T. Ohtsuka，《用于切换系统最优控制的利用结构的牛顿型方法》，国际控制期刊，第97卷第8期，第1717–1733页，2024年。[9] A. Pavlov、I. Shames 和 C. Manzie，《内点微分动态规划》，*IEEE* 控制系统技术汇刊，第29卷第6期，第2720–2727页，2021年。[10] Y. Aoyama、G. Boutselis、A. Patel 和 E. A. Theodorou，《重新审视约束微分动态规划》，载于*2021*年 *IEEE* 国际机器人与自动化会议 *(ICRA)*，第9738–9744页，2021年。[11] T. A. Howell、K. Tracy、S. Le Cleac'h 和 Z. Manchester，《CALIPSO：一种用于带锥约束和互补性约束轨迹优化的可微求解器》，载于机器人研究，第504–521页，Springer Nature Switzerland，2023年。[12] J. Nocedal、A. W¨achter 和 R. A. Waltz，《非线性内点法的自适应障碍更新策略》，*SIAM* 优化期刊，第19卷第4期，第1674–1693页，2009年。[13] G. Frison 和 M. Diehl，《HPIPM：用于模型预测控制的高性能二次规划框架》，*IFAC* 论文在线，第53卷第2期，第6563–6569页，2020年。第21届 IFAC 世界大会。[14] S. Mehrotra，《关于原始-对偶内点法的实现》，*SIAM* 优化期刊，第2卷，601，1992年4，575第。页–。[15] R. Schwan、Y. Jiang、D. Kuhn 和 C. N. Jones，《PIQP：近端内点二次规划求解器》，载于*2023*年第*62*届 *IEEE* 决策与控制会议 *(CDC)*，第1088–1093页，2023年。

[16] S. Caron、A. Zaki、P. Otta、D. Arnstr¨om、J. Carpentier、F. Yang 和 P.-A. Leziart，《qpbenchmark：Python 中可用的二次规划求解器基准测试》。https://github.com/qpsolvers/qpbenchmark，2025年。[17] M. Hinterm¨uller、K. Ito 和 K. Kunisch，《作为半光滑牛顿法的原对偶有效集策略》，SIAM 优化期刊，第13卷，第865–888页，2002年1月。[18] Y. Nesterov 和 A. Nemirovskii，《凸规划中的内点多项式算法》。工业与应用数学学会，1994年。[19] B. Stellato、G. Banjac、P. Goulart、A. Bemporad 和 S. Boyd，《OSQP：用于二次规划的算子分裂求解器》，数学规划计算，第12卷第4期，第637–672页，2020年。[20] F. Farshidian、E. Jelavic、A. Satapathy、M. Giftthaler 和 J. Buchli，《足式机器人的实时运动规划：一种模型预测控制方法》，载于2017年 IEEE-RAS 第17届人形机器人国际会议（Humanoids），第577–584页，2017年。[21] M. J. Gander 和 S. Vandewalle，《对 Parareal 时间并行时间积分方法的分析》，SIAM科学计算期刊，第29卷第2期，第556–578页，2007年。[22] A. W¨achter 和 L. T. Biegler，《关于大规模非线性规划内点过滤线搜索算法的实现》，数学规划，第106卷，第25–57页，2006年3月。[23] I. Taouil、H. Zhao、A. Dai 和 M. Khadiv，《利用潜在扩散模型实现物理一致的人形机器人行走操作》，载于2025年 IEEE-RAS 第24届人形机器人国际会议（Humanoids），第1–8页，2025年。[24] W. Jin，《面向灵巧操作的无互补多接触建模与优化》，载于机器人科学与系统 XXI (RSS)，2025年。[25] M. Ciebielski、V. Dh´edin 和 M. Khadiv，《人形机器人行走操作的任务与运动规划》，载于2025年 IEEE-RAS 第 24 届人形机器人国际会议（ Humanoids ），第 1179–1186页，2025年。[26] L. Vanroye、A. Sathya、J. De Schutter 和 W. Decr´e，《fatrop：用于机器人轨迹优化与控制的快速约束最优控制问题求解器》，载于2023年 IEEE/RSJ 智能机器人与系统国际会议（IROS），第10036–10043页，2023年。[27] C. Mastalli、S. P. Chhatoi、T. Corb´eres、S. Tonneau 和 S. Vi-jayakumar，《通过零空间求解实现逆动力学模型预测控制》，IEEE 机器人学汇刊，第39卷第4期，第3222–3241页，2023年。[28] G. Guennebaud、B. Jacob 等，《Eigen v3》。http://eigen.tuxfamily.org，2010年。[29] J. Carpentier、R. Budhiraja 和 N. Mansard，《约束动力学方程的近端稀疏求解》，载于机器人科学与系统 2021，2021年。[30] A. Escande、N. Mansard 和 P.-B. Wieber，《分层二次规划：快速在线人形机器人运动生成》，国际机器人研究期刊，第33卷第7期，第1006–1028页，2014年。[31] G. Frison、T. Sartor、A. Zanelli 和 M. Diehl，《BLAS-feo 的 BLAS API》，ACM 数学软件汇刊（TOMS），第46卷，第1–36页，2019年。[32] J. A. E. Andersson、J. Gillis、G. Horn、J. B. Rawlings 和 M. Diehl，《CasADi——用于非线性优化与最优控制的软件框架》，数学规划计算，第11卷第1期，第1–36页，2019年。[33] J. Carpentier、G. Saurel、G. Buondonno、J. Mirabel、F. Lamiraux、O. Stasse 和 N. Mansard，《Pinocchio c++ 库——快速灵活的刚体动力学算法及其解析导数实现》，载于 IEEE 系统集成国际研讨会（SII），2019年。[34] V. Dh´edin、H. Zhao 和 M. Khadiv，《动态运动的接触序列与接触面同步规划》，载于2025年 IEEE-RAS 第24届人形机器人国际会议（Humanoids），第245–252页，2025年。[35] H. Zhu、A. Meduri 和 L. Righetti，《利用蒙特卡洛树搜索进行高效物体操作规划》，载于2023年 IEEE/RSJ 智能机器人与系统国际会议（IROS），第10628–10635页，2023年。