

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 7

 在Colab中打开

动态编程

在第二章中，我们花了一些时间来思考简单摆的相位图，最后提出了一个挑战：我们能否设计一个非线性控制器来重塑相位图，只需非常适量的驱动，使直立的Fixed点变得全局稳定？在无约束扭矩的情况下，反馈抵消的解决方案（如反转重力）可以很好地工作，但也可能需要不必要的大量的控制努力。为杂技机器人和车杆系统提出的基于能量的摆动控制方案要吸引人得多，但需要一些聪明才智，可能无法扩展到更复杂的系统。在这里，我们研究了另一种解决问题的方法，使用计算最优控制来直接合成一个反馈控制器。

7.1 将控制设计制定为一个优化的过程

在本章中，我们将介绍最优控制--一种使用优化的控制设计过程。这种方法是强大的，原因有很多。首先，也是最重要的，它非常通用--允许我们为完全或欠动、线性或非线性、确定性或随机性、以及连续或离散系统同样指定控制目标。其次，它允许对潜在的非常复杂的期望行为进行简明的描述，将控制的目标指定为一个标量目标（加上一个约束条件的列表）。最后，也是最重要的一点，最优控制非常适用于数字解决方案。[1]对于那些希望得到严格处理的人来说，是一个非常好的参考资料；[2]是那些想要更容易接近的人的优秀（免费）参考资料。

最佳控制的基本思想是将控制的目标表述为标量成本函数的长期优化。让我们通过考虑一个比简单钟摆更简单的系统来介绍基本概念。

例子（7.1 双积分器的最佳控制公式）。

考虑到双积分器系统

$$\ddot{q} = u, \quad |u| \leq 1.$$

如果你想要一个系统的机械模拟（我总是这样做），那么你可以把它看作是一块在无摩擦表面上沿 x 轴运动的单位质量的砖头，控制输入提供了一个水平力， u 。

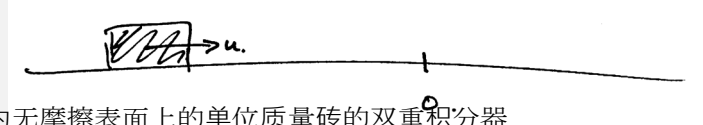


图-7.1-作为无摩擦表面上的单位质量砖的双重积分器

为了用最优控制来表述这个控制设计问题，我们必须确定一个标量目标，对每个候选控制策略 $\pi(\mathbf{x}, t)$ 从每个初始条件 (\mathbf{x}_0, t_0) 开始运行的长期性能进行评分，并确定一个必须满足的约束条件清单。对于驱动双积分器到原点的任务，人们可以想象出许多可以完成这一任务的最佳控制公式，例如：

- 最小时间： $\min_{\pi} t_f$ ，受制于 $\mathbf{x}(t_0) = \mathbf{x}_0$ ， $\mathbf{x}(t_f) = \mathbf{0}$ 。
- 二次方成本： $\min_{\pi} \int_0^{\infty} \mathbf{x}^T(t) \mathbf{Q} \mathbf{x}(t) dt$, $\mathbf{Q} \succ \mathbf{0}$ 。

其中第一种是优化时间的约束性优化公式，第二种是在每个实例中根据状态离原点的距离（在由矩阵 \mathbf{Q} 定义的度量空间中）产生惩罚，因此鼓励更直接地走向目标的轨迹，可能以需要更长的时间来到达目标为代价（事实上，这将导致以指数方式接近目标，而最小时间公式将在有限时间内到达目标）。请注意，这两个优化问题只有在系统有可能实际到达原点时才有良好的解，否则最小时间问题不能满足终端约束，而且随着时间的接近，二次成本中的积分不会收敛到一个有限值（幸运的是，双积分器系统是可控的，因此可以在有限时间内被驱动到目标）。

请注意，输入限制， $|u| \leq 1$ 也是使这个问题得到良好解决的必要条件；否则，两种优化都会导致最优策略使用无限大的控制输入来无限快地接近目标。除了输入限制，另一种限制控制力的常用方法是在输入（或“力”）上增加一个额外的二次成本，例如： $\int_0^{\infty} u^T(t) \mathbf{R} u(t) dt$, $\mathbf{R} \succ \mathbf{0}$ 。这可以是添加到上述任何一种表述中。我们将在本章末尾的例子中详细研究这些配方中的许多配方。

最佳控制在机器人领域有很长的历史。例如，在取放机械手的最小时间问题上有大量的工作，线性二次调节器（LQR）和带高斯噪声的线性二次调节器（LQG）已经成为任何执业控制工程师的基本工具。随着计算机和算法的日益强大，数值最优控制的普及在过去几年中以惊人的速度增长。

例子7.2（双积分器的最小时间问题）。

为了更直观，让我们对有输入约束的双积分器的最小时间问题的解决方案做一个非正式推导。

尽量减少 t_f
 受制于 $x(t_0)=x_0$ 。
 $x(t_f) = 0$,
 $\ddot{q}(t) = u(t)$ 。
 $|u| \leq 1$.

你期望一个最优控制器表现出什么行为？

你的直觉可能会告诉你，为了在有限的控制输入下在最短的时间内达到目标，砖头能做的最好的事情是向目标最大限度地加速，直到达到一个临界点，然后踩刹车，以便正好停在目标上。这被称为 "**砰砰**" 控制策略；这些策略对于输入有限制的系统来说通常是最优的，事实上对于双积分器来说是最优的，尽管在我们开发出更多的工具之前，我们不会证明这一点。

让我们来研究一下这个 "砰砰" 政策的细节。首先，我们可以推测出当刹车完全被踩下时，系统会精确地停在原点的状态。让我们从 $q(0)<0$ ， $\dot{q}(0)>0$ 的情况开始，"踩刹车"意味着 $u = -1$ 。对方程进行积分，我们有

$$\begin{aligned}\ddot{q}(t) &= u = -1 \\ \dot{q}(t) &= \dot{q}(0) - t \\ q(t) &= q(0) + \dot{q}(0)t - \frac{1}{2}t^2.\end{aligned}$$

代入 $t = \dot{q}(0) - \dot{q}$
 抛物线弧。

进入解决方案后，发现系统的轨道是

$$q = -\frac{1}{2}\dot{q}^2 + c_-$$

$$c_- = q(0) + \frac{1}{2}\dot{q}(0)^2$$

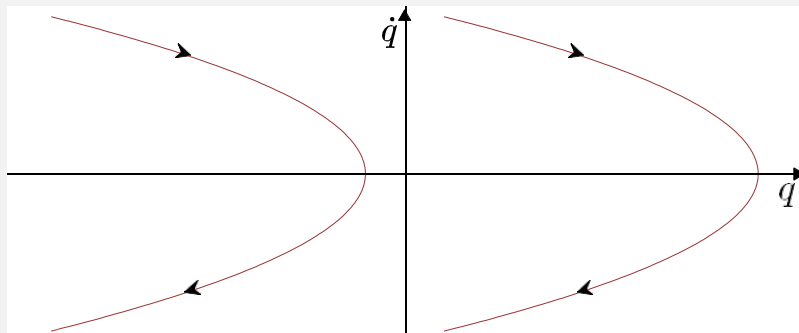


图7.2 - 系统的两个解决方案， $u = -1$

同样， $u=$ 的解是 $1q=\dot{q}^2+c_+$ ， $c_+=q(0)-\frac{1}{2}\dot{q}(0)^2$ 。

22

这些轨道中最重要的也许是通过原点的轨道（例如， $c_- = 0$ ）。按照我们最初的逻辑，如果系统在任何 q 下都比这个 \dot{q} 慢，那么最好的做法就是猛踩油门（ $u = -\text{sgn}(\dot{q})$ ）。如果系统的速度比我们所求得的 \dot{q} 快，那么最好的办法还是刹车；但不可避免的是，系统会过度偏离原点，出现要回来。我们可以用以下内容来总结这项政策。

$$u = \begin{cases} +1 & \text{如果 } (\dot{q} < 0 \text{ 且 } \dot{q} \leq -\sqrt{2q}) \text{ 或 } (\dot{q} \geq 0 \text{ 且 } \dot{q} < \sqrt{2q}) \\ 0 & \text{如果 } \dot{q} = 0 \end{cases}$$

1- 否则

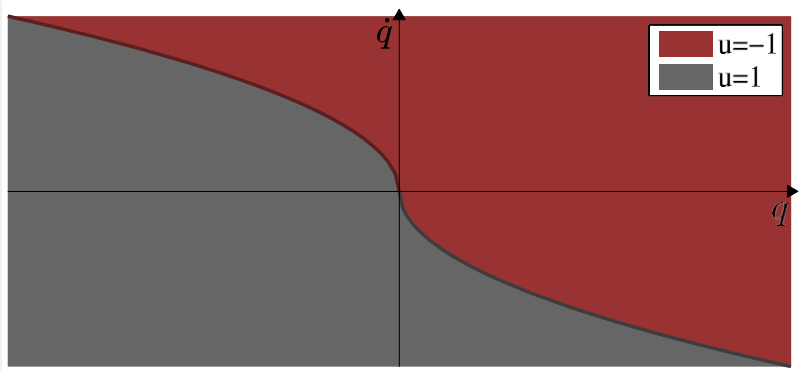


图-7.3-最小时间双积分器问题的候选最优（bang-bang）政策。

并说明了一些最优解的轨迹。

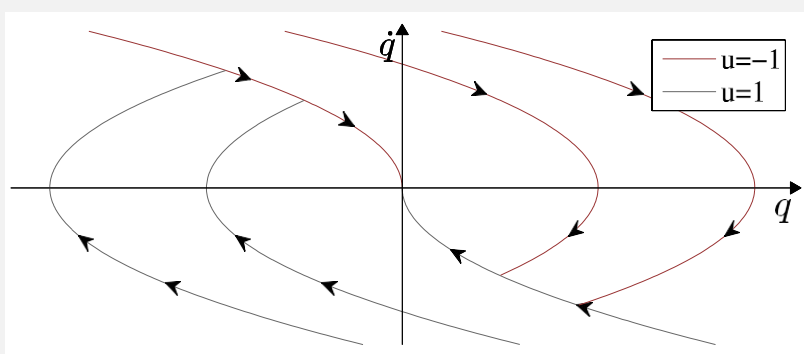


图-7.4-使用最优策略的系统的解决轨迹

为了完整起见，我们可以通过求解到达切换面所需的时间加上沿切换面移动到目标的时间来计算到达目标的最佳时间。通过一点代数，你会发现，到达目标的时间， $T(\mathbf{x})$ ，由以下公式给出

$$T(\mathbf{x}) = \begin{cases} 2\sqrt{\frac{1}{2}i^2 - q - q^*} & \text{对于 } u = +1 \text{ 制度。} \\ 0 & \text{对于 } u = 0, \\ i + 2\sqrt{\frac{1}{2}i^2 + q} & \text{对于 } u = -1, \end{cases}$$

这里绘制的
是。

图--使用 "砰砰" 政策到达原点的7.5时间

请注意，尽管政策是不连续的，但该函数是连续的（尽管不平滑）。

7.1.1 加法成本

当我们开始开发优化控制的理论和算法工具时，我们会发现有些公式比其他公式更容易处理。一个重要的例子是，使用 **加法成本** 制定目标函数可以带来巨大的简化，因为它们经常产生递归解。在加法成本公式中，轨迹的长期 "得分" 可以写成

$$\int_0^T \ell(x(t), u(t)) dt.$$

其中 $\ell()$ 是瞬时成本（也被称为 "运行成本"）， T 可以是一个有限实数或 ∞ 。我们将把具有有限 T 的问题规格称为 "有限-horizon" 问题，而 $T = \infty$ 则是 "infinite-horizon" 问题。地平线内问题的问题和解决方案往往更优雅，但需要注意的是

以确保最优控制器的积分收敛（通常是通过一个可实现的目标，让机器人累积零成本）。

上面双积分器的例子中建议的二次成本函数显然被写成了一个加法成本。乍一看，我们的最小时间问题的表述似乎不是这种形式，但实际上我们可以把它写成一个加法成本问题，使用一个有限的水平线和瞬时成本

$$\ell(x, u) = \begin{cases} 0 & \text{如果 } x = 0, \\ 1 & \text{否则的话。} \end{cases}$$

在接下来的几章里，我们将研究一些解决最优控制问题的方法。在本章的其余部分，我们将重点讨论加法成本问题和通过 **动态编程** 解决这些问题。

7.2 最优控制作为图形搜索

对于具有连续状态和连续动作的系统，动态编程是一套围绕加性成本最优控制问题的理论思想。对于具有有限、离散状态集和有限、离散动作集的系统，动态编程也代表了一套非常有效的数值算法，可以计算出最优反馈控制器。你们中的许多人以前都学过它，作为图形搜索的一个工具。

想象一下，你有一个带有状态（或节点）的有向图 $\{s_1, s_2, \dots\} \in S$ ，以及与标记为 $\{a_1, a_2, \dots\}$ 的边相关的“行动”。 $\{a\} \in A$ ，如下面这个微不足道的例子。

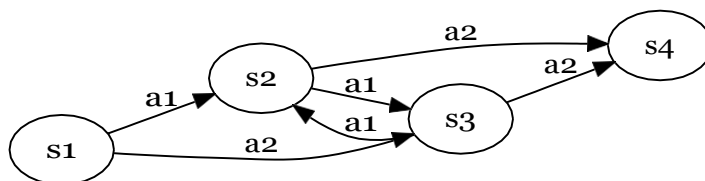


图 -7.6 一个简单的有向图。

我们还假设每条边都有一个相关的权重或成本，用 $\ell(s, a)$ 来表示处于状态 s 和采取行动 a 的成本。

$$s[n+1] = f[s[n], a[n]]。$$

例如，在上图中， $f(s_1, a_1) = s_2$ 。

在有向图上，有许多算法可以找到（或近似）从起点到目标的最优路径。在动态编程中，关键的见解是，我们可以通过递归求解从每个节点到目标的最优代价（运行最优控制器时将累积的代价）来找到最短路径。一个这样的算法从初始化所有 s_i 的估计值 $f^* = \text{开始}$ ，然后用一个迭代算法进行，设定

$$f^*(s_i) \leftarrow \min_{a \in A} [\ell(s_i, a) + f^*(f(s_i, a))]。$$
 (1)

在软件中， f^* 可以表示为一个维度等于离散状态数量的向量。这种算法被恰当地称为价值迭代，它能保证收敛到最优的成本----- $f^* \rightarrow J + *c$ [3]，而且在实践中会迅速收敛。通常情况下，更新是分批进行的--例如，一次更新所有 i 的估计值--但已知异步版本，即每次更新一个状态，也是收敛的，只要每个状态最终都是无限次地更新。假设该图有一个目标状态，有一个零成本的自我转换，那么这个成本-去向函数代表到目标的加权最短距离。

价值迭代是一个非常简单的算法，但它完成了一件非常惊人的事情：它通过迭代评估一步成本，从每个状态中有效地计算出最优策略的长期成本。如果我们知道最优成本，那么就很容易提取出最优策略， $a = \pi(*s)$ 。

$$\pi(*s_i) = \operatorname{argmin}_a [\ell(s_i, a) + J(*f(s_i, a))]。$$
 (2)

这是一个简单的算法，但玩一个例子可以帮助我们的直觉。

例子(7.3网格世界)

想象一下，一个机器人生活在一个网格（`Onite state`）世界中。想要到达目标地点。可能要与有障碍物的单元格交涉。行动是向上、向下、向左、向右移动，或者什么都不做。[4]

[点击这里查看该动画。](#)

图-7.7-网格世界最小时间问题的单步成本。目标状态的成本为零，障碍物的成本为，10,其他状态的成本为 1.



在Colab中打开

例子（双积分器的7.4动态编程）。

你可以在**DRAKE**中使用双积分器运行数值迭代（使用**arycentric**插值在节点之间插值）。



在Colab中打开

请花些时间通过自己编辑代码来尝试不同的成本函数。

让我们花点时间来体会一下这有多神奇。我们为双积分器找到最佳控制器的解决方案并不难，但它需要一些机械直觉和二阶梯方程的解决方案。所得的策略非同小可 -- 抛物线开关面的砰砰控制。价值迭代算法并没有直接使用这些东西--它是一个简单的图形搜索算法。但值得注意的是，它只需几秒钟的计算就能有效地产生相同的策略。

需要注意的是，由于离散化错误，计算出的政策和我们得出的最优政策之间存在一些差异。我们将要求你在问题中探讨这些问题。

然而，这个数字解决方案的真正价值在于，与我们对双积分器的分析解决方案不同，我们可以将这个相同的算法应用于任何数量的动态系统，几乎不需要修改。现在让我们把它应用于简单的钟摆，这在分析上是难以解决的。

例子（简单钟摆的7.5动态编程）。

你可以在**DRAKE**中使用以下方法为简单的钟摆运行数值迭代（使用**arycentric interpolation**在节点之间插值）。



在Colab中打开

同样，你可以通过自己编辑代码轻松尝试不同的成本函数。

7.3 连续动态编程

我认为图搜索算法作为第一步非常令人满意，但也很快对使用它所需的离散化的限制感到沮丧。在许多情况下，我们可以做得更好；想出在连续动态系统上更自然地工作的算法。我们将在本节中探讨这些扩展。

7.3.1 汉密尔顿-雅各比-贝尔曼方程

重要的是要明白，价值迭代方程，方程(1)和(2)，不仅仅是一种算法。它们也是最优性的条件：如果我们能产生一个满足这些方程的 J 和 π^* ，那么 π^* 一定是一个最优控制器。对于连续系统也有一套类似的条件。对于一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ 和一个infinite-horizon加性成本 $\int_0^\infty \ell(\mathbf{x}, \mathbf{u}) dt$ ，我们有。

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \right] \quad (3)$$

$$\pi^*(\mathbf{x}) = \operatorname{argmin}_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \right] \quad (4)$$

方程 3 被称为 **汉密尔顿-雅各比-贝尔曼 (HJB)** 方程。[5] 给出了一个这些方程作为离散时间极限的非正式推导是一个 P 的动态，并且还给出了著名的 "sufficient 定理"。在

[5] 是针对 infinite-horizon 情况的；我在这里把它修改为 infinite-horizon 情况的一种特殊形式。

技术哈密托方程。近似的 Jacobi 方程的

其 **时间导数** 的处理上

这取决于对状态的一阶偏导，我们在无时间导数中；公式是 3 汉密尔顿-雅各比方程的稳态解。

定理- (非正式陈述)。

7.1 HJB Sufficient Theorem

考虑一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ 和一个 infinite-horizon 加性成本 $\int_0^\infty \ell(\mathbf{x}, \mathbf{u}) dt$ 。 \mathbf{f} 和 ℓ 是连续函数， ℓ 是严格的正定函数，只有当 $\mathbf{x} = \mathbf{x}^*$ 时才会得到零。假设 $J(\mathbf{x})$ 是 HJB 方程的正 infinite 解，即 J 在 \mathbf{x} 中是连续二可的，并且是这样的

$$0 = \min_{\mathbf{u} \in U} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \right], \quad \text{对于所有的 } \mathbf{x}。$$

然后，在一些关于解决方案的存在性和约束性的技术条件下，我们可以得出 $J(\mathbf{x}) - J(\mathbf{x}^*)$ 是最佳去向成本，并且 π 是一个最佳政策。

这是一个更正式的版本，来自与 [Sasha Megretski](#) 的个人通信。

给定一个开放子集 $\Omega \subset \mathbb{R}^n$ ，有一个选定的元素 \mathbf{x}^* ，一个子集 $U \subset \mathbb{R}^m$ ，连续函数 $\mathbf{f}: \Omega \times U \rightarrow \mathbb{R}^n$ ， $g: \Omega \times U \rightarrow [0, \infty]$ ，连续二等分函数 $V: \Omega \rightarrow [0, \infty]$ ，以及函数 $\mu: \Omega \rightarrow U$ ，从而

- $g(\mathbf{x}, \mathbf{u})$ 是严格的正定核，在这个意义上，* 对于每一个矢量 $\mathbf{x}_k \in \Omega$ ， $\mathbf{u}_k \in U$ ($k=1, 2, \dots$) 的序列， $\lim_{k \rightarrow \infty} \mathbf{x}_k = \mathbf{x}$ ，这样 $\lim_{k \rightarrow \infty} g(\mathbf{x}_k, \mathbf{u}_k) = 0$ 。
- 函数 $g_V: \Omega \times U \rightarrow \mathbb{R}$ ，其定义为

$$g_V(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + \frac{dV(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u})。 \quad (5)$$

是非负的，并且只要 $\mathbf{u} = \mu(\mathbf{x})$ 就能达到零值。

c. 对于每一个 $x_0 \in \Omega$, 方程组

$$\dot{x}(t) = f(x(t), u(t)), x(0) = x_0 \text{ (即 } x(t) = x_0 + \int_0^t f(x(\tau), u(\tau)) d\tau \text{)} \quad (6)$$

$$u(t) = \mu(x(t)) \quad (7)$$

至少有一个片状连续解 $x: [0, \infty] \rightarrow \Omega$ 。

$u: [0, \infty] \rightarrow U$ 。

那么, 对于每一个 $x_0 \in \Omega$, 函数

$$J(x, u) = \int_0^\infty g(x(t), u(t)) dt.$$

在满足()的所有片断连续对 (x, u) 的集合上定义。 (6) 的所有片断连续对 (x) , 在每个片断连续的解中达到其最小值 $V(x_0) - V(x^*)$ 。 (6), (7).

证明。 首先, 观察一下, 对于所有片断连续的 (x, u) , 满足(6),

I. 由于(5),

$$\int_0^T g(x(t), u(t)) dt = V(x_0) - V(x(T)) + \int_0^T g^v(x(t), u(t)) dt \quad ;(8)$$

II. 只要 $J(x, u) < \infty$, 就存在一个正数 T 的 k 序列, $k = 1, 2, \dots, T_k \rightarrow +\infty$, 这样 $g(x(T_k), u(T_k)) \rightarrow 0$, 根据假设(a), 这意味着 $x(T_k) \rightarrow x^*$, 因为 $k \rightarrow \infty$ 。

特别是, 对于()的片断连续解 (x, u) , 满足方程(6)的片断连续解, 除此之外, 还满足方程(7), 条件(I)意味着

$$\int_0^T g(x(t), u(t)) dt = V(x_0) - V(x(T)) \leq V(x_0).$$

由于上限 $V(x_0)$ 不依赖于 T , 并且 $g(x(t), u(t)) \geq 0$ 是非负的, 我们得出结论, 即

$$\int_0^\infty g(x(t), u(t)) dt < \infty.$$

因此, 观察 (**) 适用, 并且, 对于相应的 T_k 。

$$\begin{aligned} J(x, u) &= \int_0^\infty g(x(t), u(t)) dt = \lim_{k \rightarrow \infty} \int_0^{T_k} g(x(t), u(t)) dt \\ &= \lim_{k \rightarrow \infty} V(x_0) - V(x(T_k)) = V(x_0) - V(x^*). \end{aligned}$$

此外, 由于 g 是非负的, 对于每一个 $x_0 \in \Omega$, 不等式 $V(x_0) \geq V(x^*)$ 必须成立。

为了完成证明, 对于一个任意的片状连续解, (6), 方程(8), 加上 g 的 v 非负性, 意味着

$$J(x, u) = \int_0^T g(x(t), u(t)) dt \geq V(x_0) - V(x(T)).$$

当 $J(x, u) < \infty$ 时, 将其应用于 $T = T_k$, 其中 T_k 在观察 (二) 中描述, 并在 $k \rightarrow \infty$ (且 $x(T_k) \rightarrow x^*$) 时取极限, 得到 $J(x, u) \geq V(x_0) - V(x^*)$ 。□

在不同的假设下, 也有可能证明 **suEiciency**。这里的特定假设是为了确保 $J(x(0)) < \infty$ 意味着 $x(t) \rightarrow x^*$ 。正如萨沙所说, "如果没有这样的东西, 就会出现各种反例"。

作为验证最优性的工具, **HJB** 方程实际上出乎意料地容易操作: 我们可以在不做任何积分的情况下验证一个无期限目标的最优性; 我们只需要检查一个关于最优成本-----去向函数 J 的衍生条件*。让我们看看这在双积分器的例子中的表现。

例子 (双积分器的 **7.6HJB**)

考虑使用二次成本将双积分器 (这次没有输入限制) 调控到原点的问题。

$$\ell(\mathbf{x}, \mathbf{u}) = q + \frac{1}{2} \dot{q}^2 + u^2.$$

我声称 (未经推导), 这个目标的最佳控制器是

$$\pi(\mathbf{x}) = -q - \sqrt{3\dot{q}}.$$

为了让你相信这确实是最优的, 我制作了以下的成本--去向函数。

$$J(\mathbf{x}) = \sqrt{3q} + \frac{1}{2} \dot{q}^2 + \sqrt{3\dot{q}^2}.$$

以

$$\frac{\partial J}{\partial q} = \frac{1}{2\sqrt{3q}} + \frac{1}{2\sqrt{3\dot{q}^2}} \quad \frac{\partial J}{\partial \dot{q}} = \dot{q} + \sqrt{3\dot{q}}.$$

我们可以写

$$\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) = q + \frac{1}{2} \dot{q}^2 + u^2 + (2\sqrt{3q} + 2\dot{q})\dot{q} + (2q + 2\sqrt{3\dot{q}^2})u.$$

这是一个关于 u 的凸二次函数, 所以我们可以通过找到关于 u 的梯度值为 0 的地方来找到关于 u 的最小值。

$$\frac{\partial}{\partial u} [\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u})] = 2u + q + 2\sqrt{3\dot{q}^2}.$$

设此等于并 0 求解 u , 可得。

$$u^* = -q - \sqrt{3\dot{q}}.$$

从而证实了我们的政策 π 实际上是最小化的。将 u^* 代入 **HJB** 可以看出, 右边确实简化为零了。我希望你被说服了!

请注意, 对双积分器的最小时间问题的 **HJB** 进行评估, 也会发现在梯度良好的地方, **HJB** 是满足的。这无疑是支持我们砰砰政策的越来越多的证据。

为最佳, 但由于 $\frac{\partial J}{\partial \mathbf{x}} \neq 0$ 并非处处都有, 它实际上并不满足如上所述的 **suEiciency** 定理的要求。在某种意义上,

的假设, 即 $\frac{\partial J}{\partial \mathbf{x}} \neq 0$ 在任何地方都是如此, 这使得它非常

弱。

7.3.2 求解最小化控制

要在算法中实际使用HJB，我们仍然面临一些障碍。当行动集是离散的，如在图搜索版本中，我们可以评估每一个可能的行动的一步成本加去向成本，然后简单地采取最佳。对于连续的行动空间，一般来说，我们不能依靠评估有限数量的可能u的策略来找到最小化器。

一切还没有结束。在上面的二次元成本双积分器的例子中，我们能够明确地求出以成本-去向为条件的最小化u。事实证明，这种策略实际上对我们感兴趣的许多问题都是有效的，即使系统（我们给定的）或成本函数（我们可以自由选择，但应该是可表达的）变得更加复杂。

回顾一下，我已经试图说服你，大多数感兴趣的系统都是控制性的，例如，我可以写道

$$f(\mathbf{x}, \mathbf{u}) = f_1(\mathbf{x}) + f_2(\mathbf{x})\mathbf{u}。$$

我们可以通过将自己限制在瞬时成本函数的形式来进行另一个戏剧性的简化

$$\ell(\mathbf{x}, \mathbf{u}) = \ell_1(\mathbf{x}) + \mathbf{u}^T \mathbf{R} \mathbf{u}, \quad \mathbf{R} = \mathbf{R}^T \succ 0。$$

在我看来，这不是很有限制性的--许多我自己选择写下的成本函数都可以用这种形式表达。鉴于这些假设，我们可以把HJB写成

$$0 = \min_{\mathbf{u}} [\ell_1(\mathbf{x}) + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J}{\partial \mathbf{x}} [f_1(\mathbf{x}) + f_2(\mathbf{x})\mathbf{u}]]。$$

由于这是一个关于u的正二次函数，如果系统对u没有任何约束，那么我们可以通过获取右手边的梯度，以闭合形式求出最小化的u。

$$\frac{\partial}{\partial \mathbf{u}} = 2\mathbf{u}^T \mathbf{R} + \frac{\partial J}{\partial \mathbf{x}} f_2(\mathbf{x}) = 0，$$

并将其设为零，得到

$$\mathbf{u}^* = -\frac{1}{2} \mathbf{R}^{-1} f_2^T(\mathbf{x}) \frac{\partial J}{\partial \mathbf{x}}。$$

如果对输入有线性约束，如扭矩限制，那么更普遍的情况是，这可以作为一个二次方程程序来解决（在任何特定的x）。

如果我们的系统不是控制性的，或者我们确实需要在u上指定一个不是简单二次的瞬时成本函数，会发生什么？如果目标是产生一个迭代算法，比如值迭代，那么一个常见的方法是对HJB的u做一个（正-二次）二次近似，并在算法的每次迭代中更新这个近似。这种广泛的方法通常被称为微分动态编程（c.f. [6]）。

7.3.3 J的数值解 *

在价值迭代算法中使用HJB的另一个主要障碍是，估计的最佳成本-去向函数 J^* 必须以某种方式用一组有限数字表示，但我们还不知道它必须采取的潜在形式。In fact, knowing the time-to-goal solution for minimum-time problem with

双重积分器，我们看到，即使是非常简单的动力学和目标，这个函数也可能需要是非光滑的。

对 J^* —一个在状态空间上定义的标量值函数—进行参数化的一个自然方法是在网格上定义其值。这种方法与用于解决其他偏微分方程（PDEs）的相对非常先进的数值方法有密切联系，例如出现在有限元建模或流体力学中的算法。然而，一个重要的区别是，我们的PDE生活在状态空间的维度上，而其他学科的许多网格表示是针对二维或三维空间的。另外，我们的PDE在状态空间中的位置可能有不连续的地方（或者至少是不连续的梯度），而这些位置并不是事先就知道的。

对这个问题的一个稍微普遍的看法是将网格（以及相关的插值函数）描述为只是函数逼近的一种表示形式。Using a neural network to represent the cost-to-go also falls under the domain of function approximation, perhaps representing the other extreme in terms of complexity; using deep networks in approximate dynamic programming is common in deep reinforcement learning, which we will discuss more later in the book.

带有函数近似的值迭代

如果我们用一个无限参数化的函数来近似 J^* ，参数为 α ，那么这就立即引起了许多重要问题。

- 如果真正的成本-去向函数不在规定的函数类中；例如，不存在一个对所有 \mathbf{x} 都满足 ϵ -Efficiency 条件的 α ，怎么办？（这种情况似乎很有可能发生）。
- 我们应该在迭代算法中应用什么更新？
- 我们能对其收敛性说些什么呢？

让我们首先考虑由价值迭代更新的最小二乘法给出的更新。

在价值迭代更新中使用最小二乘法解有时被称为 *Value Iteration*，其中 \mathbf{x}_k 是从连续空间中抽取的一些样本，对于离散时间系统，迭代的近似解为

$$J^*(\mathbf{x}_0) = \min_{\mathbf{u}[\cdot]} \sum_{n=0}^{\infty} \ell(\mathbf{x}[n], \mathbf{u}[n])。$$

$$\text{s. t. } \mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]), \mathbf{x}[0] = \mathbf{x}_0$$

成为

$$J_k = \min_{\mathbf{u}} [\ell(\mathbf{x}_k, \mathbf{u}) + J^*(f(\mathbf{x}_k, \mathbf{u}))] \quad (9)$$

$$\alpha \leftarrow \operatorname{argmin}_{\alpha} \sum_k (J^*(\mathbf{x}_k) - J^{\alpha}(\mathbf{x}_k))^2 \quad (10)$$

由于所需的值 J^d 只是对走动成本的初始猜测，我们将迭代地应用这个算法，直到（希望）我们达到某种数字收敛。

请注意，(10) 中的更新与做最小二乘法的优化不太一样。

$$\sum_k (J^*(\mathbf{x}_k) - \min_{\mathbf{u}} [\ell(\mathbf{x}_k, \mathbf{u}) + J^*(f(\mathbf{x}_k, \mathbf{u}))])^2。$$

因为在这个方程中， α 对 J^* 的两次出现都有影响。在(10)中，我们通过将 J 作为期望值来削减这种依赖关系；这个版本在实践中表现得更好。像许多事情一样，这是

一个旧的想法，被赋予了一个新的名字

在深度强化学习的文献中，人们认为右边的 f^* 是一个 "fixed" 目标网络 的输出。对于非线性函数近似器，(10) 中对 α 的更新经常被替换成梯度下降的一个步骤。

示例（7.7 神经值迭代）。

让我们试着用 [PyTorch](#) 中的神经网络重现我们的双积分值迭代例子。



一般来说，用一般的函数近似器进行价值迭代的收敛性和准确性保证是相当弱的。但对于 [线性函数逼近器](#) 的特殊情况，我们确实有一些结果。一个线性函数近似器的形式是：

$$f_{\alpha}^*(\mathbf{x}) = \sum_i \alpha_i \psi_i(\mathbf{x}) = \psi^T(\mathbf{x}) \alpha。$$

其中 $\psi(\mathbf{x})$ 是一个潜在的非线性特征的向量。特征的常见例子包括多项式、径向基函数或在网格上使用的大多数插值方案。线性函数逼近器的突出特点是能够精确求解 α ，以便在最小二乘法意义上最佳地表示所需函数。对于线性函数逼近器来说，这很简单。

$$\alpha \leftarrow \left(\psi^T(\mathbf{x}_K) \psi(\mathbf{x}_K) + J_1^d \right)^{-1} \psi^T(\mathbf{x}_K) J_K^d$$

其中的符号 $+$ 指的是 Moore-Penrose 伪逆。值得注意的是，对于线性函数近似器来说，这种更新仍然是已知的收敛于全局最优 α^* 。

网格上的值迭代

想象一下，我们用一个网状物在该状态空间上用 K 个网状物点 \mathbf{x}_k 来近似成本-去向函数，我们想进行价值迭代更新。

$$\forall k, f^*(\mathbf{x}_k) \leftarrow \min_{\mathbf{u}} [\ell(\mathbf{x}_k, \mathbf{u}) + f^*(f(\mathbf{x}_k, \mathbf{u}))]。 \quad (11)$$

但必须处理这样一个事实，即 $f(\mathbf{x}_k, \mathbf{u})$ 可能不会导致下一个状态直接出现在一个网格点。大多数网格的插值方案可以写成附近网格点数值的某种加权组合，例如

$$f^*(\mathbf{x}) = \sum_i \theta_i(\mathbf{x}) f^*(\mathbf{x}_i)。 \quad \sum_i \theta_i = 1$$

θ_i 为第 i 个网格点的相对权重。在 [DRAKE](#) 中，我们实现了重心插值 [2]。以 $\alpha_i = f^*(\mathbf{x}_i)$ ，即第 i 个网格点的代价估计值为例，我们可以看到，这正是一个重要的线性函数近似的 θ -tted 值迭代的特例。此外，假设 $\theta(\mathbf{x}_i) = 1$ ，（例如，[在一个网格点上](#) 对插值有贡献的唯一一点是在

网点本身），公式 (11) 中的更新正是最小二乘法的更新（并且它实现了零误差）。这就是你在上面已经实验过的数值迭代例子中使用的表示方法。

连续时间系统

对于具有连续时间动力学的系统的解决方案，我必须揭开一个迄今为止我一直隐藏的细节，以保持更简单的符号。让我们考虑一个具有有限-horizon的问题。

$$\begin{aligned} \min_{\mathbf{u}[\cdot]} \sum_{n=0}^N \ell(\mathbf{x}[n], \mathbf{u}[n]). \\ \text{s. t. } \mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]), \mathbf{x}[0] = \mathbf{x}_0 \end{aligned}$$

In fact, the way that we compute this is by solving the *time-varying cost-to-go function* backwards in time:

$$\begin{aligned} J^*(\mathbf{x}, N) &= \min_{\mathbf{u}} \ell(\mathbf{x}, \mathbf{u}) \\ J^*(\mathbf{x}, n-1) &= \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + J^*(f(\mathbf{x}, \mathbf{u}), n)]. \end{aligned}$$

价值迭代更新的收敛等同于在时间上倒退解决这个时间变化的成本-去向，直到它达到一个稳定状态的解决方案（有限-horizon解决方案）。这就解释了为什么价值迭代只在最优成本--去向有界时才收敛。

现在让我们考虑连续时间的版本。同样，我们有一个随时间变化的成本， $J^*(\mathbf{x}, t)$ 。现在

$$\frac{dJ^*}{dt} = \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial t},$$

和我们的最优性条件是

$$0 = \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial t}].$$

但由于 $\frac{\partial J^*}{\partial t}$ 不依赖于 \mathbf{u} ，我们可以把它从min中拉出来，写成（真）的HJB

$$-\frac{\partial J^*}{\partial t} = \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u})].$$

标准的数字配方[8]来解决这个问题，就是要近似于 $\frac{\partial J^*}{\partial t}$ 上然后在时间上对方程进行反向积分（如果目标是找到超视距解，则直到收敛）。如果，对于网格点 \mathbf{x}_i ，我们有 $\alpha(i, t) = \frac{\partial J^*}{\partial t}(\mathbf{x}_i, t)$ ，那么。

$$-\dot{\alpha}(i, t) = \min_{\mathbf{u}} [\ell(\mathbf{x}_i, \mathbf{u}) + \frac{\partial J^*(\mathbf{x}_i, t)}{\partial \mathbf{x}} f(\mathbf{x}_i, \mathbf{u})].$$

其中偏导数是在网格上用合适的奥尼特-迭代近似来估计的，通常在右边添加一些“粘性”项以提供额外的数值稳健性；见Lax-Friedrichs方案[8](5.3.1节)为例。

在应用控制（至少在机器人领域）中使用HJB数值解决方案的最明显和最一致的活动来自伯克利大学的Claire Tomlin小组。他们的工作利用了Ian Mitchell的Level Set Toolbox，该工具使用上述技术在笛卡尔网格上求解Hamilton-Jacobi PDEs，甚至包括双积分器的最小时间问题作为一个教学实例[9]。

7.4 延伸部分

到目前为止，我们所介绍的基本公式还有很多很好的扩展。我将尝试在这里列出几个最重要的。在这门课程中，我也让一些学生探索了非常有趣的扩展；例如，[10]看了

在（离散状态）价值函数上施加一个低秩结构，利用矩阵完备性的思想来获得良好的估计，尽管只更新了一小部分的状态。

7.4.1 随机控制的Onite MDPs

动态编程、最优控制的加性成本方法的最令人惊奇的特点之一是它相对容易地扩展到优化随机系统。

对于离散系统，我们可以通过在边上增加与行动相关的过渡概率来概括我们在图上的动力学。这个新的动态系统被称为马尔科夫决策过程（MDP），我们完全用过渡概率来写动态过程

$$\Pr(s[n+1] = s' | s[n] = s, a[n] = a)。$$

对于离散系统，这只是一个大的查找表。我们为任何系统的执行而产生的成本现在是一个随机变量，因此我们将控制的目标制定为优化预期成本，例如

$$J^*(s[0]) = \min_{a[\cdot]} E \left[\sum_{n=0}^{\infty} \ell(s[n], a[n]) \right] \quad (12)$$

请注意，还有许多其他潜在的目标，如最小化最坏情况下的误差，但预期成本是特殊的，因为它保留了动态编程递归。

$$J^*(s) = \min_{a, s'} E [\ell(s, a) + J^*(s')] = \min [\ell(s, a) + \sum \Pr(s'|s, a) J^*(s')]。$$

值得注意的是，如果我们使用这些优化条件来构建我们的价值迭代算法

$$\hat{J}(s) \leftarrow \min_{\text{如同}} [\ell(s, a) + \sum \Pr(s'|s, a) \hat{J}(s')]。$$

那么这个算法就具有与确定性系统的对应算法相同的强大收敛保证。而且，它的计算成本基本上没有增加。

确定性的连续状态值迭代的随机解释

这里有一个特别好的观察。让我们假设我们有离散的控制输入和离散时间的动力学，但有一个连续的状态空间。回顾上面描述的网状算法的V值迭代。事实上，所产生的更新与我们把系统当作离散状态的MDP完全相同， β_i 代表过渡到状态 x 的概率！这揭示了离散化对解决方案的影响--这里的离散化误差将导致一种与在相邻节点上扩散的概率相对应的diffusion。

这是我们以后为[随机和稳健控制](#)开发的更为通用的工具包的一个预览。

7.4.2 线性编程方法

对于离散MDPs，值迭代是一种神奇的算法，因为它很简单，但已知会收敛到全局最优， J^* 。然而，其他重要的算法也是已知的；其中一个最重要的算法是使用线性

编程。这种提法提供了另一种观点，但也可能更具有普遍性，甚至对问题的某些实例来说更有效率性。

回顾一下公式 (1) 中的优化条件。如果我们将成本-去向函数描述为一个矢量， $J_i = J(s_i)$ ，那么这些优化条件可以用矢量形式重写为

$$\mathbf{J} = \min_a [\ell(a) + \mathbf{T}(a)\mathbf{J}]. \quad (13)$$

其中 $\ell(a_i) = \ell(s_i, a)$ 是成本向量， $\mathbf{T}(a_{i,j}) = \Pr(s_j | s_i, a)$ 是过渡概率矩阵。让我们把 \mathbf{J} 作为决策变量的向量，并把公式替换为 (13) 的约束。

$$\forall a, \mathbf{J} \leq \ell(a) + \mathbf{T}(a)\mathbf{J}. \quad (14)$$

显然，对于有限 a ，这是一个有限的线性约束列表，对于任何满足这些约束的向量 \mathbf{J} ，我们有 $\mathbf{J} \leq \mathbf{J}^*$ (按元素排列)。现在写出线性程序。

$$\begin{aligned} & \text{最大限度 } \mathbf{c}^T \mathbf{J}, \\ & \text{受制于 } \forall a, \mathbf{J} \leq \ell(a) + \mathbf{T}(a)\mathbf{J}, \end{aligned}$$

其中 \mathbf{c} 是任何正矢量。这个问题的解决方案是 $\mathbf{J} = \mathbf{J}^*$ 。

也许更有趣的是，这种方法可以被推广到线性函数近似器。以我上面的符号的矢量形式，现在有一个 $\Psi_{ij} = \psi^T(\mathbf{x}_i)$ 的特征矩阵，我们可以写出 LP

$$\begin{aligned} & \mathbf{c}^T \Psi \alpha \text{ 最大化 } \alpha_0, \\ & \text{受制于 } \forall a, \Psi \alpha \leq \ell(a) + \mathbf{T}(a)\Psi \alpha. \end{aligned} \quad \begin{matrix} (15) \\ (16) \end{matrix}$$

这一次，解决方案不一定是最优的，因为 $\Psi \alpha^*$ 只近似于 \mathbf{J}^* ，以及 \mathbf{c} 元素的相对值（称为“状态相关性权重”）。可以确定不同特征下近似的相对紧密度[...]. 11].

7.5 练习题

练习 (7.1 贴现和价值迭代的收敛性)。

让我们考虑价值迭代的离散时间、状态和行动版本。

$$f^*(s_i) \leftarrow \min_{a \in A} [\ell(s_i, a) + f^*(f(s_i, a))]. \quad (17)$$

它找到了最优的政策，适用于无限期的成本函数

$\sum_{n=0}^{\infty} \ell(s[n], a[n])$ 。如果 J 是真正最优成本-----去，表明任何解决方案

$f^* = J + c$ ，其中 c 是任何常数标量，是这个值迭代更新的一个 fixed 点。Is the controller still optimal, even if the estimated cost-to-go function is off by a constant factor?

令人惊讶的是，增加一个折扣系数可以帮助解决这个问题。考虑一下 infinite-地贴现成本： $\sum_{n=0}^{\infty} \gamma^n \ell(s[n], a[n])$ ，其中 $0 < \gamma \leq 1$ 是贴现系数。

相应的价值迭代更新为

$$f^*(s_i) \leftarrow \min_{a \in A} [\ell(s_i, a) + \gamma f^*(f(s_i, a))]. \quad (18)$$

为简单起见，假设存在一个状态 s_i ， J 在最优政策下是一个零成本的 fixed 点；例如 $\ell(s_i, \pi^*(s_i)) = 0$ 且 $f(s_i, \pi^*(s_i)) = s_i$ 。 $f^* = J + c$

当 $\gamma < 1$ 时，仍然是价值迭代更新的一个Fixed点。展示你的工作。

练习 (7.2 选择一个成本函数)

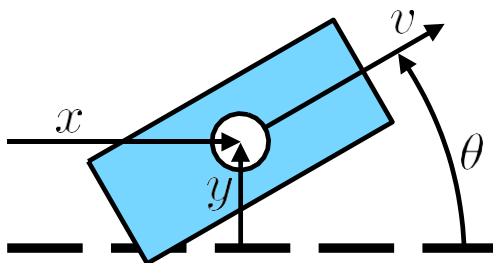


图-7.8-自主汽车以速度 v 在直路上行驶。

上图显示了一辆自主汽车在一条笔直的道路以恒定速度 $v > 0$ 运动。设 x 是汽车沿路的（纵向）位置， y 是它与中心线的（横向）距离， θ 是中心线与运动方向的角度。唯一的控制动作是转向速度 u ，它被约束在区间 $[u_{\min}, u_{\max}]$ 内（其中 $u_{\min} < 0$ 和 $u_{\max} > 0$ ）。我们

用简单的运动学模型描述汽车动态

$$\begin{aligned}\dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= u\end{aligned}$$

让 $\mathbf{x} = [x, y, \theta]^T$ 是状态向量。为了优化汽车的轨迹，我们考虑一个二次目标函数

$$J = \int_0^{\infty} [\mathbf{x}^T(t) \mathbf{Q} \mathbf{x}(t) + R u^2(t)] dt.$$

其中， \mathbf{Q} 是一个恒定的正负矩阵（因此是对称的）， R 是一个恒定的非负标量（注意，这里0允许 $R=0$ ）。

- 假设我们唯一的目标是使汽车和中心线之间的距离 y 尽可能小，尽可能快，而不用担心其他事情。你对 \mathbf{Q} 和 R 的选择会是什么？
- 如果你把上一点的权重 \mathbf{Q} 和 R 乘以一个任意的正系数 α ，汽车的行为会有什么变化？
- (a)点的成本函数可能容易导致过度的急转弯。你会增加 \mathbf{Q} 或 R 的哪一项来缓解这个问题？
- 乡村道路的两侧比中间更滑。这类目标函数（二次目标）是否丰富到足以包括对急转弯的惩罚，这种惩罚随着汽车与中心线的距离而增加？
- 有了这个动力学模型和这个目标函数，你是否会选择一个严格的正非线性的权重矩阵 \mathbf{Q} （与你希望汽车完成的任务无关）？为什么？

练习(7.3 欠妥最佳控制问题)

在这个练习中，我们将看到看似简单的成本函数是如何得到令人惊讶的结果的。考虑单积分系统 $\dot{x} = u$ ，初始状态 $x(0) = 0$ 。我们希望找到控制信号 $u(t)$ ，使看似简单的成本函数最小。

无害的成本函数

$$J = \int_0^T x^2(t) + (u^2(t) - 1)^2 dt.$$

与 $T \rightarrow \infty$ 。为此，我们考虑一个方波控制，参数为 $\tau > 0$ ：

$$u_\tau(t) = \begin{cases} 1 & \text{如果 } t \in [0, \tau] \cup [3\tau, 5\tau] \cup [7\tau, 9\tau] \cup \dots \\ -1 & \text{如果 } t \in [\tau, 3\tau] \cup [5\tau, 7\tau] \cup [9\tau, 11\tau] \cup \dots \end{cases}$$

- a. 哪些状态 x 和投入 u 的运行成本是多少？

$$\ell(x, u) = x^2 + (u^2 - 1)^2$$

等于零？

- b. 现在考虑两个控制信号 $u_{\tau_1}(t)$ 和 $u_{\tau_2}(t)$ ， $\tau_1 = \tau/2$ 。哪一个会产生较低的成本 J ？提示：先画出系统状态 $x(t)$ 在这两个控制信号下的演变过程。
- c. 当 τ 越来越小的时候，成本 J 会发生什么变化？最佳控制信号是什么样子的？你能用一个 ∞ 带宽控制器来实现它吗？

练习 (7.4 一个具有二次成本的线性系统)

考虑标量控制二阶方程

$$\dot{x} = x + u.$$

和非限时成本函数

$$J = \int_0^\infty [3x^2(t) + u^2(t)] dt.$$

正如我们将在[线性二次调节一章](#)中看到的那样，这种问题的最佳成本--去向的形式是 $J^* = Sx^2$ 。不难看出，这反过来意味着最优控制器的形式为 $u^* = -Kx$ 。

- a. 想象一下，你把表达式 $J^* = Sx^2$ 插入这个问题的HJB方程中，用 S 求解，得到 $S \leq 0$ 。这个结果会敲响警钟吗？解释一下你的答案。
- b. 使用HJB sufficiency定理，得出 S 和 K 的最佳值。
- c. 在使用数字控制器时，我们通常要对系统的动态进行采样，并用离散时间模型对其进行近似。让我们引入一个时间步长 $h > 0$ ，并将动力学离散为

$$\frac{x[n+1] - x[n]}{h} = x[n] + u[n]$$

和目标为

$$J = \sum_{n=0}^{\infty} (3x^2[n] + u^2[n])$$

One of the following expressions is the correct cost-to-go $J^*_h(x)$ for this discretized problem.你能在不求解离散时间HJB方程的情况下确定它吗？解释一下你的答案。

- i. $J_h^*(x) = Sxh^4$, $S_h = +3h + \sqrt{6h^2}$.
- ii. $J_h^*(x) = hSx^2$, $S_h = +12h + 2\sqrt{h^2 + h + 1}$.
- iii. $J_h^*(x) = hSx^2$, $S_h = +32h^2 + \sqrt{h^2 + h + 2}$.

间问题

练习7.5 (最小时的值迭代

在这个练习中，我们分析了值迭代算法的性能，考虑了它在[双积分器最小时间问题上的应用](#)。在[这个Python笔记本中](#)，你会发现这个分析所需要的一切。花点时间浏览一下这个笔记本，了解其中的代码，然后回答以下问题。

- a. 在[笔记本](#) "值迭代策略的性能" 一节的末尾，我们绘制了采用值迭代策略的闭环中双积分器的状态轨迹，以及由此产生的控制信号。
 - i. 状态空间轨迹是否遵循我们在[例子](#)中看到的理论上最优的二次元弧线？
 - ii. 我们从价值迭代中得到的控制策略是一个砰砰的策略吗？换句话说，控制信号是否只取集合 $\{1, -0, 1\}$ 中的值？
 - iii. 请用两句话解释一下，这样做的原因是什么？行为。
- b. 在[笔记本](#)的 "值迭代算法" 部分，增加 q 轴和 q' 轴上的结点数量，重新确定值迭代中使用的状态空间网格。这是否否定了你在(a)点中看到的问题？
- c. 在[笔记本](#)的最后一节，实现[例子](#)中理论上最优的控制策略，并使用图谱来验证闭环系统的行为是否符合预期。

练习

7.6(Fitted Value Iteration for the Double Integrator)

在这个练习中，我们将用神经网络实现Fitted值迭代，并使用它来生成一个双积分器的控制器。在[这个python笔记本中](#)，你将需要实现一些东西。花点时间浏览一下这个笔记本，了解其中的代码，然后回答以下问题。为了方便你，书面问题也会列在笔记本中。

- a. 通过笔记本中的编码部分进行工作。
- b. 尽管它已经很接近了，但在笔记本上，为什么我们的实现可能无法成功地将系统完全稳定在原点？
- c. 在笔记本中我们实现了 "Fitted value iteration"。在下面的问题中，你将被要求思考我们在第七章开始时看到的 "图形搜索" 版本的价值迭代和Fitted价值迭代之间的区别。请回答下面的问题并作简要解释。
 - i. 实际上，在连续状态空间中，图搜索值迭代能否直接执行？Fitted值迭代可以吗？
 - ii. 实际上，图形搜索价值迭代能否直接执行连续行动空间？Fitted值迭代可以吗？
- d. 对于我们的实现，我们假设状态之间的转换是确定性的。这意味着，给定一个特定的状态 x_t 和行动 u_t ，我们将

在数学上，总是在同一个唯一的 X_{t+1} 中结束。

$$x_{t+1} = f(x_t, u_t)$$

在现实世界的场景中，我们经常要处理我们的模型没有捕捉到的干扰或动态，这可能导致我们的状态转换是随机的。在数学上。

$$P(x_{t+1}|x_t, u_t) = f(x_t, u_t)$$

如果我们有随机的过渡，笔记本实现的哪些步骤需要改变，以及如何改变？下面列出了可能选择的步骤--你可以选择多个。在描述变化时，口头描述就可以了，你不需要包括任何代码。

- i. 在执行 O_t 之前，对点 x 的集合进行采样。注意，这不包括 x_{next} 。
- ii. 在执行 O_t 之前生成成本 G 。假设我们使用的是二次元成本。
- iii. 在训练循环中更新成本函数目标。`Jd, ind = torch.min(G + discount*Jnext, dim=1)`。
- iv. 在模拟系统时选择政策中的最佳行动。

参考文献

1. Dimitri P. Bertsekas, "动态编程和最优控制", Athena Scientific, 2000.
2. Richard S. Sutton和Andrew G. Barto, "强化学习. 简介", 麻省理工学院出版社。 2018.
3. Dimitri P. Bertsekas 和 John N. Tsitsiklis, "神经动态编程", Athena Scientific, 十月。 1996.
4. Richard S. Sutton和Andrew G. Barto, "强化学习. 简介", 麻省理工学院出版社。 1998.
5. Dimitri P. Bertsekas, "Dynamic Programming & Optimal Control", Athena Scientific, vol. I and II, May 1, 2005.
6. David H. Jacobson 和 David Q. Mayne, "Differential Dynamic Programming", American Elsevier Publishing Company, Inc., 1970.
7. Remi Munos和Andrew Moore, "Barycentric Interpolators for Continuous Space and Time Reinforcement Learning", *Advances in Neural Information Processing Systems*, vol. 1024-103011, 1998.
8. Stanley Osher和Ronald Fedkiw, "Level Set Methods and Dynamic Implicit Surfaces", Springer, 2003.
9. Ian M. Mitchell and Jeremy A. Templeton, "A Toolbox of Hamilton-Jacobi求解器", 《混合系统计算与控制》论文集, 3月。 2005.
10. Yuzhe Yang and Guo Zhang and Zhi Xu and Dina Katabi, "Harnessing Structures for Value-Based Planning and Reinforcement Learning", *International Conference on Learning Representations*, 2020.

11. Daniela Pucci de Farias, "近似动态编程的线性编程方法。理论与应用", 博士论文, 斯坦福大学, 6月。2002.

[上一章](#)

[目录](#)

[下一章](#) [可访](#)

[问性](#)

© Russ Tedrake, 2021

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 8

 在Colab中打开

线性二次方调节器

虽然解决连续系统的动态编程问题在一般情况下是非常困难的，但有一些非常重要的特殊情况，其解决方案是很容易的。其中大部分涉及线性动态和二次成本的变体。最简单的情况，称为线性二次调节器（LQR），被表述为将一个时间不变的线性系统稳定到原点。

线性二次调节器可能是迄今为止最优控制理论中最重要和最有影响的结果。在本章中，我们将推导出基本算法和各种有用的扩展。

8.1 基本推导

考虑一个状态空间形式的线性时间不变系统。

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}.$$

境内的成本函数为

$$J = \int_0^{\infty} [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}] dt, \quad \mathbf{Q} = \mathbf{Q}^T \geq 0, \mathbf{R} = \mathbf{R}^T > 0.$$

我们的目标是找到满足HJB的最佳成本-去向函数 $J^*(\mathbf{x})$ 。

$$\forall \mathbf{x}, \quad 0 = \min_{\mathbf{u}} [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}} (\mathbf{Ax} + \mathbf{Bu})].$$

这里有一个重要的步骤--众所周知，对于这个问题，最佳的成本-去向函数是二次的。这很容易验证。让我们选择这样的形式。

$$J^*(\mathbf{x}) = \mathbf{x}^T \mathbf{S} \mathbf{x}, \quad \mathbf{S} = \mathbf{S}^T \geq 0.$$

这个函数的梯度是

$$\frac{\partial J}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{S}_o$$

由于我们在结构上保证了 \min 里面的项是二次和凸的（因为 $\mathbf{R} \succ 0$ ），我们可以通过找到这些项的梯度消失的解决方案来明确地取到最小值。

$$\frac{\partial}{\partial \mathbf{u}} = 2\mathbf{u}^T \mathbf{R} + 2\mathbf{x}^T \mathbf{S} \mathbf{B} = 0.$$

这就得到了最佳政策

$$\mathbf{u}^* = \pi^*(\mathbf{x}) = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} \mathbf{x} = -\mathbf{K} \mathbf{x}.$$

将其插入到HJB中，并简化后可得到

$$0 = \mathbf{x}^T [\mathbf{Q} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + 2 \mathbf{S} \mathbf{A}] \mathbf{x}.$$

除了 $2\mathbf{S} \mathbf{A}$ ，这里所有的项都是对称的，但由于 $\mathbf{x}^T \mathbf{S} \mathbf{A} \mathbf{x} = \mathbf{x}^T \mathbf{A}^T \mathbf{S} \mathbf{x}$ ，我们可以写成

$$0 = \mathbf{x}^T [\mathbf{Q} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{S} \mathbf{A} + \mathbf{A}^T \mathbf{S}] \mathbf{x}.$$

因为这个条件必须对所有的 \mathbf{x} 都成立，所以考虑矩阵方程是足够的

$$0 = \mathbf{S} \mathbf{A} + \mathbf{A}^T \mathbf{S} - \mathbf{S} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{Q}.$$

这个极其重要的方程是**代数李卡蒂方程**的一个版本。但众所周知，当且仅当系统可控时，该方程有一个单一的正定解，并且有很好的数值方法来找到该解，即使是在高维问题上。通过调用 `(K,S) = LinearQuadraticRegulator(A,B,Q,R)`，可以从 **DRAKE** 获得最优策略和最优成本-去向函数。

如果说走访成本的二次方形式的出现似乎很神秘。

认为线性系统 $\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B} \mathbf{K}) \mathbf{x}$ 的解的形式是

$\mathbf{x}(t) = e^{(\mathbf{A} - \mathbf{B} \mathbf{K})t} \mathbf{x}(0)$ ，并尝试将其插回积分成本函数。你会发现，成本的形式是 $J = \mathbf{x}^T(0) \mathbf{S} \mathbf{x}(0)$ 。

值得更仔细地研究一下最优政策的形式。由于价值函数代表的是走的成本，所以尽可能快地往下走是明智之举。事实上， $-\mathbf{S} \mathbf{x}$ 是在价值的最陡峭下降方向上。

功能。然而，并非所有方向都能在状态空间中实现。 $-\mathbf{B}^T \mathbf{S} \mathbf{x}$ 恰恰代表了最陡峭的下降对控制空间的投影。

最后，矩阵 \mathbf{R}^{-1} 的预缩放使下降的方向有偏差，以考虑到相对的

我们在不同的控制输入上设置的权重。请注意，尽管这种解释是直截了当的，但我们正在下降的斜率（在价值函数中， \mathbf{S} ）是一个复杂的动态和成本的函数。

例8.1（双积分器的LQR）。

现在可以使用LQR来重现我们上一章的**HJB例子**：

`examples/double_integrator/lqr.py`

与手工推导的例子一样，我们的数字解决方案返回的是

$$\mathbf{K} = [1, \sqrt{3}], \quad \mathbf{S} = \begin{bmatrix} \sqrt{3} & 1 \\ 1 & \sqrt{3} \end{bmatrix}.$$

8.1.1 非线性系统的局部稳定化

尽管我们的主要兴趣是在非线性动力学方面，但LQR与我们极为相关，因为它可以提供最佳控制的局部近似。

解释该非线性系统的解。给定非线性系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ ，和a稳定的工作点， $(\mathbf{x}_0, \mathbf{u}_0)$ ， $\mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) = \mathbf{0}$ 。

坐标系统

$$\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0 \quad \bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}_0。$$

并观察到

$$\dot{\bar{\mathbf{x}}} = \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})。$$

我们可以用一阶泰勒展开来近似地计算出

$$\dot{\bar{\mathbf{x}}} \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{x}} (\mathbf{x} - \mathbf{x}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{u}} (\mathbf{u} - \mathbf{u}_0) = \mathbf{A} \bar{\mathbf{x}} + \mathbf{B} \bar{\mathbf{u}}。$$

同样，我们可以在误差坐标中定义一个二次成本函数，或者采取一个关于操作点的非线性成本函数的二阶近似（成本函数中的线性项和常数项可以通过参数化J的全二次形式轻松纳入推导，正如在下面的线性二次跟踪推导中所看到的）。

由此产生的控制器的形式为 $\bar{\mathbf{u}}^* = -\mathbf{K} \bar{\mathbf{x}}$ 或

$$\mathbf{u}^* = \mathbf{u}_0 - \mathbf{K}(\mathbf{x} - \mathbf{x}_0)。$$

为了方便起见，**DRAKE**允许你在大多数动力系统（包括由许多子系统组成的框图）上调用`controller = LinearQuadraticRegulator(system, context, Q, R)`；它将为你的执行线性化。

例子（杂技机器人、车杆和四旋翼机器人的8.2LQR）

LQR为我们已经描述过的一些模型系统的典型“平衡”问题提供了一个非常满意的解决方案。这里是带有这些例子的笔记本，再次。



我认为非常有说服力的是，同样的推导（和有效的相同代码）可以稳定如此多样的系统！我认为这是对的。

8.2 有限跨度公式

回顾一下，Finite-horizon问题的成本-去向是随时间变化的，因此HJB sufficiency条件需要一个额外的 $\frac{\partial J^*}{\partial t}$ 项。

$$\forall \mathbf{x}, \forall t \in [t_0, t_f]。 \quad 0 = \min_{\text{燃料}} [\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} + \frac{\partial J^*}{\partial t}]$$

8.2.1 有限跨度LQR

考虑由LTI状态空间方程支配的系统，其形式为

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}_0$$

和一个有限-horizon成本函数, $J = h(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t))dt$, 其中

$$\begin{aligned} h(\mathbf{x}) &= \mathbf{x}^T \mathbf{Q} \mathbf{x}, \quad \mathbf{Q}_f = \mathbf{Q}_f^T \geq 0 \\ \ell(\mathbf{x}, \mathbf{u}) &= \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}, \quad \mathbf{Q} = \mathbf{Q}^T \geq 0, \mathbf{R} = \mathbf{R}^T > 0 \end{aligned}$$

写下HJB, 我们有

$$0 = \min_{\mathbf{u}} [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}} (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) + \frac{\partial J^*}{\partial t}]_0$$

由于 \mathbf{u} 的正二次形式, 我们可以通过设置梯度为零来找到最小值。

$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}} &= 2\mathbf{u}^T \mathbf{R} + \frac{\partial J^*}{\partial \mathbf{x}} \mathbf{B} = 0 \\ \mathbf{u} &= \pi(\mathbf{x}, t) = -\frac{1}{2} \mathbf{R}^{-1} \mathbf{B}^T \frac{\partial J^*}{\partial \mathbf{x}} \end{aligned}$$

In order to proceed, we need to investigate a particular form for the cost-to-go function, $J^*(\mathbf{x}, t)$. 让我们试试这种形式的解决方案。

$$J^*(\mathbf{x}, t) = \mathbf{x}^T \mathbf{S}(t) \mathbf{x}, \quad \mathbf{S}(t) = \mathbf{S}^T(t) \geq 0.$$

在这种情况下, 我们有

$$\frac{\partial J^*}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{S}(t), \quad \frac{\partial J^*}{\partial t} = \mathbf{x}^T \dot{\mathbf{S}}(t) \mathbf{x}.$$

因此

$$\begin{aligned} \mathbf{u}^* &= \pi(\mathbf{x}, t) = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{S}(t) \mathbf{x} \\ 0 &= \mathbf{x}^T [\mathbf{Q} - \mathbf{S}(t)^{-1} \mathbf{B} \mathbf{R} \mathbf{B}^T \mathbf{S}(t) + \mathbf{S}(t) \mathbf{A} + \mathbf{A}^T \mathbf{S}(t) + \dot{\mathbf{S}}(t)] \mathbf{x}. \end{aligned}$$

因此, $\mathbf{S}(t)$ 必须满足条件 (被称为连续时间 **二项式李嘉诚方程**) 。

$$-\dot{\mathbf{S}}(t) = \mathbf{S}(t) \mathbf{A} + \mathbf{A}^T \mathbf{S}(t) - \mathbf{S}(t)^{-1} \mathbf{B} \mathbf{R} \mathbf{B}^T \mathbf{S}(t) + \mathbf{Q}.$$

和终端条件

$$\mathbf{S}(t_f) = \mathbf{Q}_f.$$

Since we were able to satisfy the HJB with the minimizing policy, we have met the sufficiency condition, and have found the optimal policy and optimal cost-to-go function.

请注意, 前言中描述的超视距LQR解正是该方程的稳态解, 用 $\dot{\mathbf{S}}(t)=0$ 表示。事实上, 对于可控系统, 该方程是稳定的 (时间上的倒退), 正如预期的那样, 超视距解在超视距时间限制下收敛于超视距解。

8.2.2 时变的LQR

上面的推导是成立的, 即使动力学是由以下公式给出的

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}.$$

同样地, 成本函数 \mathbf{Q} 和 \mathbf{R} 也可以是时变的。这很令人惊讶, 因为时变线性系统是一个相当普遍的系统类别。它基本上不需要对时间依赖性如何进入进行假设。

但如果 \mathbf{A} 或 \mathbf{B} 在时间上是不连续的，那么我们就必须使用适当的技术来准确地积分二阶梯方程。

8.2.3 非线性系统的局部轨迹稳定化

时变LQR最强大的应用之一是围绕非线性系统的额定轨迹进行线性化，并使用LQR提供一个轨迹控制器。这将与我们在[轨迹优化一章](#)中开发的算法很好地结合起来。

让我们假设我们有一个名义轨迹， $\mathbf{x}_0(t)$ ， $\mathbf{u}_0(t)$ 为 $t \in [t_1, t_2]$ 定义的。与时间不变的分析类似，我们首先要确定一个相对于轨迹的局部坐标系。

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0(t) \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0(t)。$$

现在我们有

$$\dot{\bar{\mathbf{x}}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}_0 = \mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)。$$

我们可以再次用一阶泰勒展开对其进行近似，即

$$\dot{\bar{\mathbf{x}}} \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{x}} (\mathbf{x} - \mathbf{x}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{u}} (\mathbf{u} - \mathbf{u}_0) - \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) = \mathbf{A}(t)\bar{\mathbf{x}} + \mathbf{B}(t)\bar{\mathbf{u}}。$$

这与使用LQR来稳定Fixed-point非常相似，但有一些重要的区别。首先，线性化是随时间变化的。第二，我们的线性化对沿可行轨迹的任何状态都有效（不仅仅是Fixed-point），因为坐标系是随着轨迹移动的。

同样，我们可以在误差坐标中定义一个二次成本函数，或者沿轨迹采取一个非线性成本函数的二阶近似（成本函数中的线性项和常数项可以通过参数化J的全二次形式轻松纳入推导中，正如在下面的线性二次跟踪推导中看到的那样）。

由此产生的控制器的形式为 $\bar{\mathbf{u}}^* = -\mathbf{K}(t)\bar{\mathbf{x}}$ 或

$$\mathbf{u}^* = \mathbf{u}_0(t) - \mathbf{K}(t)(\mathbf{x} - \mathbf{x}_0(t))。$$

DRAKE提供了一个[FiniteHorizonLinearQuadraticRegulator](#)方法；如果你把一个非线性系统传给它，它将使用自动二分法在适当的坐标中为你执行线性化。

请记住，稳定性是一个关于随着时间的推移而发生的事情的声明。为了谈论稳定轨迹的问题，必须对所有 $t \in [t_1, \infty]$ 的轨迹进行定义。这可以通过考虑一个无时间的轨迹来实现，这个轨迹是

在时间 $t_2 \geq t_1$ 终止于一个可稳定的Fixed点，并在 $t \geq t_2$ 时保持在Fixed点。 $S(t_2) = S_\infty$ ，并在时间上从 t_2 向后求解至 t_1

在剩余的轨迹中。*现在我们可以说，我们已经稳定了弹道！*

8.2.4 线性四次方最佳跟踪

为了完整起见，我们考虑线性二次调节器的一个略微更普遍的形式。标准的LQR推导试图将系统驱动到零。现在考虑一下这个问题。

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ h(\mathbf{x}) &= (\mathbf{x} - \mathbf{x}(t_f))^T \mathbf{Q} (\mathbf{x} - \mathbf{x}(t_f)) \quad \mathbf{Q} = \mathbf{Q}^T \geq 0 \\ \ell(\mathbf{x}, \mathbf{u}, t) &= (\mathbf{x} - \mathbf{x}(t))^T \mathbf{Q} (\mathbf{x} - \mathbf{x}(t)) + (\mathbf{u} - \mathbf{u}(t))^T \mathbf{R} (\mathbf{u} - \mathbf{u}(t)) \quad \mathbf{R} = \mathbf{R}^T \geq 0 \\ \mathbf{Q} &= \mathbf{Q}^T \mathbf{R} \geq 0, \mathbf{R} = \mathbf{R}^T \geq 0\end{aligned}$$

现在，猜一个解决方案

$$J^*(\mathbf{x}, t) = \mathbf{x}^T \mathbf{S}_{xx}(t) \mathbf{x} + 2\mathbf{x}^T \mathbf{s}_x(t) + s_0(t) \quad \mathbf{S}_{xx}(t) = \mathbf{S}^T(t) \geq 0.$$

在这种情况下，我们有

$$\frac{\partial J^*}{\partial \mathbf{x}} = 2\mathbf{x}^T \mathbf{S}_{xx}(t) + 2\mathbf{s}_x^T(t) \quad \frac{\partial J^*}{\partial t} = \mathbf{x}^T \dot{\mathbf{S}}_{xx}(t) \mathbf{x} + 2\mathbf{x}^T \dot{\mathbf{s}}_x(t) + \dot{s}_0(t).$$

使用HJB。

$$0 = \min_{\mathbf{u}} [(\mathbf{x} - \mathbf{x}(t))^T \mathbf{Q} (\mathbf{x} - \mathbf{x}(t)) + (\mathbf{u} - \mathbf{u}(t))^T \mathbf{R} (\mathbf{u} - \mathbf{u}(t)) + \frac{\partial J^*}{\partial t} (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}}] \quad \frac{\partial J^*}{\partial t}$$

我们有

$$\begin{aligned}\frac{\partial}{\partial \mathbf{u}} &= 2(\mathbf{u} - \mathbf{u}(t))^T \mathbf{R} + (2\mathbf{x}^T \mathbf{S}_{xx}(t) + 2\mathbf{s}_x^T(t)) \mathbf{B} = 0, \\ \mathbf{u}^*(t) &= \mathbf{u}_d(t) - \mathbf{R}^{-1} \mathbf{B}^T [\mathbf{S}_{xx}(t) \mathbf{x} + \mathbf{s}_x(t)].\end{aligned}$$

HJB可以通过向后整合得到满足

$$\begin{aligned}-\dot{\mathbf{S}}_{xx}(t) &= \mathbf{Q} - \mathbf{S}_{xx}(t) \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{S}_{xx}(t) + \mathbf{S}_{xx}(t) \mathbf{A} + \mathbf{A}^T \mathbf{S}_{xx}(t) \\ -\dot{\mathbf{s}}_x(t) &= -\mathbf{Q} \mathbf{x}_d(t) + [\mathbf{A}^T - \mathbf{S}_{xx}(t) \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T] \mathbf{s}_x(t) + \mathbf{S}_{xx}(t) \mathbf{B} \mathbf{u}_d(t) \\ -\dot{s}_0(t) &= \mathbf{x}_d(t)^T \mathbf{Q} \mathbf{x}_d(t) - \mathbf{s}_x^T(t) \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{s}_x(t) + 2\mathbf{s}_x(t)^T \mathbf{B} \mathbf{u}_d(t).\end{aligned}$$

从final条件

$$\begin{aligned}\mathbf{S}_{xx}(t_f) &= \mathbf{Q}_f \\ \mathbf{s}_x(t_f) &= -\mathbf{Q} \mathbf{x}_f(t_f) \\ s_0(t_f) &= \mathbf{x}_d^T(t_f) \mathbf{Q} \mathbf{x}_d(t_f).\end{aligned}$$

注意， \mathbf{S}_{xx} 解决方案与更简单的LQR推导相同，并且是对称的（正如我们所假设的）。还要注意的是， $s_0(t)$ 对控制没有影响（甚至是间接影响），所以通常可以忽略。

关于二次函数形式的一个快速观察，这可能对调试有帮助。我们知道 $J(\mathbf{x}, t)$ 必须是均匀的正数。这是真的，即 $\mathbf{S}_{xx} \geq 0$ 和 $s_0 \geq \mathbf{s}_x^T \mathbf{S}_{xx}^{-1} \mathbf{s}_x$ ，这来自于在 $\mathbf{x}(min)$ 处对函数的评估，其定义为

$$\left[\frac{\partial J^*(\mathbf{x}, t)}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}_{min}(t)} = 0.$$

8.2.5 线性最终边界值问题

Finite-horizon LQR公式可以通过设置一个finite \mathbf{Q}_f 来施加一个严格的final边界值条件。然而，从一个finite初始条件向后积分Riccati方程并不是很实用。为了解决这个问题。

让我们考虑求解 $\mathbf{P}(t) = \mathbf{S}(t)^{-1}$ 。利用矩阵关系 $\frac{d\mathbf{S}^{-1}}{dt} = -\mathbf{S}^{-1} \frac{d\mathbf{S}}{dt} \mathbf{S}^{-1}$ 。

$\frac{d\mathbf{S}}{dt}$

我们有。

$$-\dot{\mathbf{P}}(t) = -\mathbf{P}(t) \mathbf{Q} \mathbf{P}(t) + \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}(t) - \mathbf{P}(t) \mathbf{A}^T.$$

与final条件

$$\mathbf{P}(t_f) = 0.$$

这个Riccati方程可以在时间上向后整合，以得到一个解决方案。

非常有趣和强大的是，如果选择 $\mathbf{Q}=0$ ，因此在时间 T 之前对轨迹不施加任何位置成本，那么这个反李嘉图方程就变成了一个线性ODE，可以明确地解决。这些关系被用于可控性格拉米亚的推导，但在这里我们用它们来设计一个反馈控制器。

8.3 变化和扩展

8.3.1 离散时间李卡提方程

基本上上述所有的结果对于离散时间系统来说都有一个自然的关联。更重要的是，离散时间版本往往更容易在模型预测控制（MPC）环境中思考，我们将在下面和接下来的章节中讨论。

考虑到离散时间的动态。

$$\mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n].$$

而我们希望最小化

$$\min \sum_{n=0}^{N-1} \mathbf{x}^T[n] \mathbf{Q} \mathbf{x}[n] + \mathbf{u}^T[n] \mathbf{R} \mathbf{u}[n], \quad \mathbf{Q} = \mathbf{Q}^T, \mathbf{R} \succ 0, \mathbf{R}^T = 0.$$

走出去的成本由以下公式得出

$$J(\mathbf{x}, n-1) = \min_{\mathbf{u}} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + J(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, n).$$

如果我们再一次把

$$J(\mathbf{x}, n) = \mathbf{x}^T \mathbf{S}[n] \mathbf{x}, \quad \mathbf{S}[n] = \mathbf{S}^T[n], \quad \mathbf{S}[N] = 0,$$

那么我们有

$$\mathbf{u}^*[n] = -\mathbf{K}[n] \mathbf{x}[n] = -(\mathbf{R} + \mathbf{B}^T \mathbf{S}[n] \mathbf{B})^{-1} \mathbf{B}^T \mathbf{S}[n] \mathbf{A} \mathbf{x}[n].$$

屈服

$$\mathbf{S}[n-1] = \mathbf{Q} + \mathbf{A}^T \mathbf{S}[n] \mathbf{A} - (\mathbf{A}^T \mathbf{S}[n] \mathbf{B})(\mathbf{R} + \mathbf{B}^T \mathbf{S}[n] \mathbf{B})^{-1} \mathbf{B}^T \mathbf{S}[n] \mathbf{A}, \quad \mathbf{S}[N] = 0,$$

这就是著名的*Riccati difference equation*。地平线上的LQR解决方案是由这个方程的（正-去-点）Fixed-point给出的。

$$\mathbf{s} = \mathbf{q} + \mathbf{a}^T \mathbf{s} - (\mathbf{a}^T \mathbf{s} \mathbf{b})(\mathbf{r} + \mathbf{b}^T \mathbf{s} \mathbf{b})^{-1} \mathbf{b}^T \mathbf{s} \mathbf{a}.$$

和连续时间的情况一样，这个方程非常重要，我们有专门的数值方法来解决这个离散时间代数李卡蒂方程（DARE）。当你在一个只有离散状态和单一周期时间步长的系统上评估LinearQuadraticRegulator方法时，DRAKE会自动委托给这些数值方法。

示例（离散8.3时间与连续时间LQR）。

你可以在这个笔记本中探索离散时间和连续时间公式之间的关系。



8.3.2 具有输入和状态约束的LQR

线性最优控制的一个自然延伸是考虑对输入或状态轨迹的严格约束。最常见的是线性不等式约束，如 $\forall n, |u[n]| \leq 1$ 或 $\forall n, x[n] \geq -2$ （任何形式的线性约束

$Cx + Du \leq e$ 可以用同样的工具来解决）。关于解决方案，人们知道很多在离散时间的情况下，这个问题的计算难度很大。

比起无约束的情况。几乎总是这样，我们放弃以闭合形式求解最佳控制策略，而是从某个特定的初始条件 $x[0]$ 求解某个有限水平线上的最佳控制轨迹 $u[\cdot]$ 。幸运的是，这个问题是一个凸优化问题，我们通常可以快速可靠地解决这个问题，在每个时间点上解决这个问题，有效地将运动规划算法变成一个反馈控制器；这个想法就是著名的模型预测控制（MPC）。我们将在[轨迹优化章节](#)中提供细节。

由于“显式MPC”方面的工作，我们实际上了解了不等式约束的LQR问题的最优政策是什么样子的[1]--最佳策略现在是分片线性的（尽管仍然是连续的），每片都由一个多角形描述，而最佳成本-去向在相同的多角形上是分片二次的。不幸的是，碎片的数量随着约束条件的数量和问题范围呈指数级增长，使得除了非常小的问题之外，计算这些碎片是不切实际的。然而，有一些有希望的方法来近似显式MPC（参见[.2]）。

有一种重要的情况确实有闭合形式的解决方案，那就是具有线性**平等**约束的LQR（参见[3]，第三b节）。这在稳定具有运动学约束的机器人的情况下尤为重要，例如在四杆连杆中出现的封闭运动学链，甚至是对双脚着地的双足机器人的线性化。

8.3.3 LQR是一个凸式优化

人们也可以使用线性矩阵不等式（LMI）来设计LQR的收益。我将推迟推导，直到我们涵盖LQR的政策梯度观点，因为LMI的表述是基于基本政策评价标准的变量变化。如果你想往前看，你可以[在这里](#)找到这个公式。

解决代数Riccati方程仍然是计算LQR解决方案的首选方式。但知道也可以用凸式优化来计算它是很有帮助的。除了加深我们的理解外，这对于概括基本的LQR解决方案（例如用于**稳健的稳定**）或作为更大的优化的一部分共同解决LQR收益也很有用。

8.3.4 通过最小二乘法的有限跨度LQR

我们也可以用优化方法获得离散时间有限-horizon（包括时变或跟踪变体）LQR问题的解决方案--在这种情况下，它实际上简化为一个简单的最小二乘问题。本节的介绍可以看作是[Youla参数化](#)（有时称为“Q参数化”）在控制方面的一个简单实现。这里的表述的小变化在LQR的minimax变体（优化最坏情况下的性能）中发挥了重要作用，我们将在鲁棒控制章节中讨论（例如[.45]）。

首先，让我们认识到，默认的参数化不是凸的。鉴于

$$\min \sum_{n=0}^{N-1} x^T[n] Q x[n] + u^T[n] R u[n], \quad Q = Q^T \succeq 0, R = R^T \succ 0$$

受制于 $x[n+1] = Ax[n] + Bu[n]$ 。
 $x[0] = x_0$

如果我们希望搜索形式为

$$\mathbf{u}[n] = \mathbf{K}\mathbf{x}_n[n].$$

那么我们有

$$\begin{aligned}\mathbf{x}[1] &= \mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{K}\mathbf{x}_0, \\ \mathbf{x}[2] &= \mathbf{A}(\mathbf{A} + \mathbf{B}\mathbf{K}_0)\mathbf{x}_0 + \mathbf{B}\mathbf{K}_1(\mathbf{A} + \mathbf{B}\mathbf{K}_0)\mathbf{x}_0 \\ \mathbf{x}[n] &= \left(\prod_{i=0}^{n-1} (\mathbf{A} + \mathbf{B}\mathbf{K}_i) \right) \mathbf{x}_0\end{aligned}$$

正如你所看到的，成本函数中的 $\mathbf{x}[n]$ 项包括我们的决策变量相乘 -- 导致一个非凸的目标。诀窍是将决策变量重新参数化，并将反馈写成以下形式。

$$\mathbf{u}[n] = \tilde{\mathbf{K}}_n \mathbf{x}_0.$$

导致

$$\begin{aligned}\mathbf{x}[1] &= \mathbf{A}\mathbf{x}_0 + \tilde{\mathbf{B}}_0 \mathbf{x}_0, \\ \mathbf{x}[2] &= \mathbf{A}(\mathbf{A} + \tilde{\mathbf{B}}_0) \mathbf{x}_0 + \tilde{\mathbf{B}}_1 \mathbf{x}_0 \\ \mathbf{x}[n] &= \mathbf{A}^n \mathbf{x}_0 + \sum_{i=0}^{n-1} \mathbf{A}^{n-i-1} \tilde{\mathbf{B}}_i \mathbf{x}_0\end{aligned}$$

现在，所有的决策变量， $\tilde{\mathbf{K}}_i$ ，都线性地出现在 $\mathbf{x}[n]$ 的解中，因此（凸的）四次方地出现在目标中。

我们仍然有一个取决于 \mathbf{x} 的目标函数，但我们希望找到所有 \mathbf{x} 的最佳 $\tilde{\mathbf{K}}$ 。为了实现这一点，让我们评估这个最小二乘法问题的最佳条件，首先是取得关于 $\tilde{\mathbf{K}}$ 的目标梯度，即。

$$\frac{\partial}{\partial \tilde{\mathbf{K}}_i} \mathbf{x}^T (\mathbf{R} + \sum_{m=i+1}^{N-1} \mathbf{B}^T (\mathbf{A}^{m-i-1})^T \mathbf{Q} \mathbf{A}^{m-i-1} \mathbf{B}) \mathbf{x} = \sum_{m=i+1}^{N-1} (\mathbf{A}^m)^T \mathbf{Q} \mathbf{A}^{m-i-1} \mathbf{B}.$$

我们可以通过解决线性矩阵方程来满足所有 \mathbf{x} 的这个最优条件。

$$-\tilde{\mathbf{K}}_i^T (\mathbf{R} + \sum_{m=i+1}^{N-1} \mathbf{B}^T (\mathbf{A}^{m-i-1})^T \mathbf{Q} \mathbf{A}^{m-i-1} \mathbf{B}) + \sum_{m=i+1}^{N-1} (\mathbf{A}^m)^T \mathbf{Q} \mathbf{A}^{m-i-1} \mathbf{B} = 0.$$

我们总是可以求出 $\tilde{\mathbf{K}}_i$ ，因为它与一个（对称）正定核矩阵相乘（它是一个正定核矩阵和许多正半定核矩阵的总和），它总是可逆的。

如果你需要恢复原始的 \mathbf{K} 参数，你可以用以下方法递归地提取它们

$$\begin{aligned}\tilde{\mathbf{K}}_0 &= \mathbf{K}_0, \\ \tilde{\mathbf{K}}_n &= \mathbf{K}_n \prod_{i=0}^{n-1} (\mathbf{A} + \mathbf{B}\mathbf{K}_i), \quad 0 < n \leq N-1.\end{aligned}$$

但实际上往往没有这个必要。在某些应用中，只要知道LQR控制下的性能成本就足够了，或者用基于干扰的反馈明确地处理对干扰的响应（我已经答应在鲁棒控制一章中这样做）。毕竟，我们在这里写的问题表述，没有提到干扰，假设模型是完美的，而且

控件 $\tilde{\mathbf{K}}_n \mathbf{x}_0$ 与 $\mathbf{K}_n \mathbf{x}[n]$ 一样适合部署。

"系统级合成" (SLS) 是一个重要的、稍有不同的名称。

的方法，即直接对 **闭环响应** 进行优化[6]. 尽管SLS是一个非常通用的工具，但对于我们在这里考虑的特定配方，它可以简化为创建额外的决策变量 Φ_n ，例如

$$\mathbf{x}[n] = \Phi_n \mathbf{x}[0].$$

并将上面的优化写成

$$\begin{aligned} \min_{\tilde{\mathbf{K}}, \Phi_n} \sum_{n=0}^{N-1} \mathbf{x}^T [0] (\Phi_n^T \mathbf{Q} \Phi_n + \mathbf{K}_n^T \mathbf{R} \tilde{\mathbf{K}}_n) \mathbf{x}[0]. \\ \text{受制于} \quad \forall n, \quad \Phi_{n+1} = \mathbf{A} \Phi_n + \tilde{\mathbf{K}}_n. \end{aligned}$$

再次，如果我们只想得到简单情况下的解决方案，这里提出的算法并不像解Riccati方程那样有效，但如果我们把LQR综合与其他目标/约束结合起来，它们就变得非常强大。例如，如果我们想增加一些稀疏性约束（例如，强制一些 $\tilde{\mathbf{K}}_n$ 的元素为零），那么我们可以在以下条件下解决二次优化问题线性平等约束[7].

8.4 练习题

8.5 注意事项

8.5.1 有限跨度LQR推导（一般形式）。

为了完整起见，我在这里包括了连续时间有限- horizon LQR的推导，其中包含了所有的功能和口号。

考虑一个状态空间形式的时变艾因（近似的）连续时间动力系统。

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u} + \mathbf{c}(t).$$

和一个一般二次形式的运行成本函数。

$$\begin{aligned} \ell(t, \mathbf{x}, \mathbf{u}) &= \begin{bmatrix} \mathbf{x} \end{bmatrix}^T \mathbf{Q}(t) \begin{bmatrix} \mathbf{x} \end{bmatrix} + \begin{bmatrix} \mathbf{u} \end{bmatrix}^T \mathbf{R}(t) \begin{bmatrix} \mathbf{u} \end{bmatrix} + 2 \mathbf{x}^T \mathbf{N}(t) \mathbf{u}. \\ \forall t \in [t_0, t_f]. \quad \mathbf{Q}(t) &= \begin{bmatrix} Q_{xx}(t) & q_x(t) \\ q_x^T(t) & q_0(t) \end{bmatrix}, \quad \mathbf{Q}_{xx}(t) \succ 0. \quad \mathbf{R}(t) = \begin{bmatrix} R_{uu}(t) & r_u(t) \\ r_u^T(t) & r_0(t) \end{bmatrix}, \\ \mathbf{R}_{uu}(t) &\succ 0. \end{aligned}$$

请注意，我们的LQR "最佳跟踪" 推导是在这种形式，因为我们总是可以写出

$$(\mathbf{x} - \mathbf{x}(dt))^T \mathbf{Q} (\mathbf{x} - \mathbf{x}(dt)) + (\mathbf{u} - \mathbf{u}(dt))^T \mathbf{R} (\mathbf{u} - \mathbf{u}(dt)) + 2(\mathbf{x} - \mathbf{x}(dt))^T \mathbf{N} (\mathbf{u} - \mathbf{u}(dt)).$$

通过采取

$$\begin{aligned} \mathbf{Q}_{xx} &= \mathbf{Q}, & \dot{q}_x &= -\mathbf{Q} \mathbf{x}_d - \mathbf{N} \mathbf{u}, & \dot{q}_0 &= \mathbf{x}^T \mathbf{Q} \mathbf{x} + 2 \mathbf{x}^T \mathbf{N} \mathbf{u}_d, \\ \mathbf{R}_{uu} &= \mathbf{R}, & \dot{r}_u &= -\mathbf{R} \mathbf{u}_d - \mathbf{N}^T \mathbf{x}_d, & \dot{r}_0 &= \mathbf{u}^T \mathbf{R} \mathbf{u} + \mathbf{N}^T \mathbf{x}_d. \end{aligned}$$

当然，我们也可以用 \mathbf{Q} 添加二次成本。让我们寻找一个正的二次，时间变化的成本-----的函数形式。

$$\begin{aligned} J(t, \mathbf{x}) &= \begin{bmatrix} \mathbf{x} \end{bmatrix}^T \mathbf{S}(t) \begin{bmatrix} \mathbf{x} \end{bmatrix}, \quad \mathbf{S}(t) = \begin{bmatrix} s_{xx}(t) & s_{x0}(t) \\ s_{x0}^T(t) & s_0(t) \end{bmatrix}, \quad \mathbf{S}(t) \succ 0. \\ \frac{\partial J}{\partial \mathbf{x}} &= 2 \mathbf{x}^T \mathbf{S}_{xx} + 2 \mathbf{s}_{x0}, \quad \frac{\partial J}{\partial t} = \begin{bmatrix} 1 \end{bmatrix}^T \mathbf{S} \begin{bmatrix} 1 \end{bmatrix}. \end{aligned}$$

写出HJB。

$$\min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial I}{\partial \mathbf{x}} [\mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u} + \mathbf{c}(t)] + \frac{\partial I}{\partial t} \right] = 0,$$

我们可以找到最小化的 \mathbf{u} ，即

$$\frac{\partial}{\partial \mathbf{u}} = 2\mathbf{u}^T \mathbf{R} + 2\mathbf{r}^T + 2\mathbf{x}^T \mathbf{N} + (2\mathbf{x}^T + 2\mathbf{s}^T) \mathbf{B} = 0$$

$$\mathbf{u}^* = -\mathbf{R}^{-1} \left[\mathbf{N} + \mathbf{S} \mathbf{x} \mathbf{x}^T \mathbf{B} \right] \mathbf{x} = -\mathbf{K}(t) \mathbf{x} = -\mathbf{K}_x(t) \mathbf{x} - \mathbf{k}_0(t).$$

将其插入HJB，我们就得到了更新的Riccati微分方程。由于这必须对所有的 \mathbf{x} 都成立，我们可以收集二次、线性和offset项，并将它们分别设为零，得到。

$$\begin{aligned} -\dot{\mathbf{S}}_{xx} &= \mathbf{Q}_{xx} - (\mathbf{N} + \mathbf{x} \mathbf{x}^T \mathbf{S} \mathbf{B}) \mathbf{R}^{-1} (\mathbf{N} + \mathbf{x} \mathbf{x}^T \mathbf{S} \mathbf{B})^T + \mathbf{x} \mathbf{x}^T \mathbf{S} \mathbf{A} + \mathbf{A}^T \mathbf{S} \mathbf{x} \mathbf{x}^T \\ -\dot{\mathbf{s}}_x &= \mathbf{q}_x - (\mathbf{N} + \mathbf{x} \mathbf{x}^T \mathbf{S} \mathbf{B}) \mathbf{R}^{-1} (\mathbf{r}_u + \mathbf{B}^T \mathbf{s}_x) + \mathbf{A}^T \mathbf{s}_x + \mathbf{x} \mathbf{x}^T \mathbf{S} \mathbf{c}_0 \\ -\dot{s}_0 &= q_0 + \mathbf{r}_0^T (\mathbf{r}_u + \mathbf{B}^T \mathbf{s}_x) - \mathbf{r}_u^T \mathbf{R}^{-1} (\mathbf{r}_u + \mathbf{B}^T \mathbf{s}_x) + 2 \mathbf{s}_0^T \mathbf{c}_0 \end{aligned}$$

与最终条件 $\mathbf{S}(t_f) = \mathbf{Q}_f$ 。

在离散时间版本中，我们有...

呵！这些方程的数值解可以通过调用 [DRAKE](#) 中的 [FiniteHorizonLinearQuadraticRegulator](#) 方法获得。愿你永远不必自己输入它们并对它们进行单元测试。

参考文献

1. A.Alessio和A. Bemporad, "A survey on explicit model predictive control", *Int.非线性模型预测控制的评估和未来方向研讨会*。2009.
2. Tobia Marcucci和Robin Deits和Marco Gabiccini和Antonio Bicchi和Russ Tedrake, "近似混合模型预测控制用于复杂环境下的多触点推送恢复", *仿人机器人 (Humanoids)* , 2017年IEEE-RAS第17届国际会议, 2017。[[链接](#)]
3. Michael Posa and Scott Kuindersma and Russ Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems", *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp.1366-1373, May, [2016.[链接](#)]
4. J.Lofberg, "闭环最小化{MPC}的近似", *第42届{IEEE}。{国际}{决定}和{控制}会议 ({IEEE}。{猫}。{No}.03CH37475)* , 第2卷第1438--1442页2,, 12月。2003.
5. Sadra Sadraddini和Russ Tedrake, "有保证的约束满足的鲁棒输出反馈控制", *在第23届ACM国际混合系统会议论文集。计算和控制*, 第4页12,, [2020.[链接](#)]。
6. James Anderson and John C. Doyle and Steven Low and Nikolai Matni, "System {Level} {Synthesis}", *arXiv:1904.01634 [cs, math]*, apr, 2019.
7. Yuh-Shyang Wang and Nikolai Matni and John C. Doyle, "Localized {LQR}。{最优}{控制}", *arXiv:1409.6404 [cs, math]*, 9月。2014.

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 9

 在Colab中打开

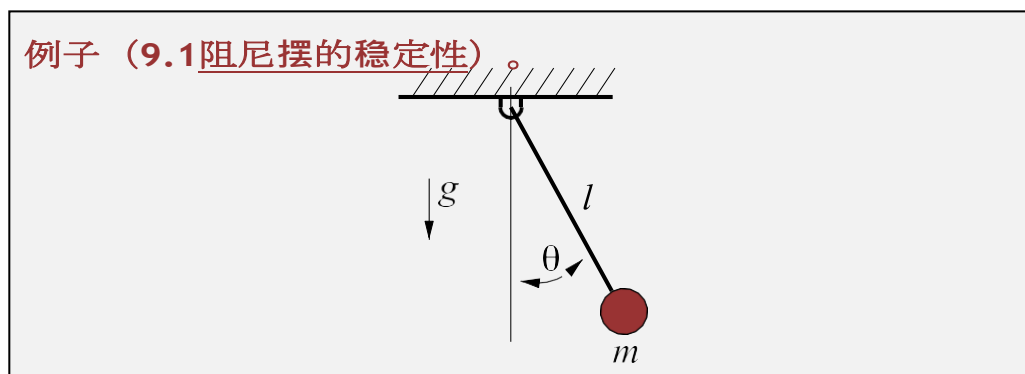
李亚普诺夫分析

最优控制提供了一个强大的框架，使用优化语言来制定控制问题。但是，解决非线性系统的最优控制问题是很难的在许多情况下，我们并不真正关心找到**最优**控制器，而是对任何能保证完成特定任务的控制器感到满意。在许多情况下，我们仍然使用优化的计算工具来制定这些问题，在这一章中，我们将学习一些工具，这些工具可以为系统提供有保障的控制方案，这些方案的复杂程度超出了我们可以找到最优反馈的范围。

有许多关于李亚普诺夫分析的优秀书籍；例如[1]是一本非常好的、可读性很强的参考书，而[2]可以提供一个严格的处理。在这一章中，我将总结（没有证明）一些来自李亚普诺夫分析的关键定理，但随后也将介绍一些数字算法.....其中许多是新的，以至于它们还没有出现在任何主流教科书中。

9.1 李亚普诺夫函数

让我们从我们最喜欢的简单例子开始。



回顾一下，阻尼单摆的运动方程如下

$$ml\ddot{\vartheta} + mgl \sin \vartheta = -b\dot{\vartheta}。$$

我把阻尼写在右边，以提醒我们这是一个外部扭矩，我们已经建立了模型。

这些方程代表了一个简单的二阶微分方程；在本章中

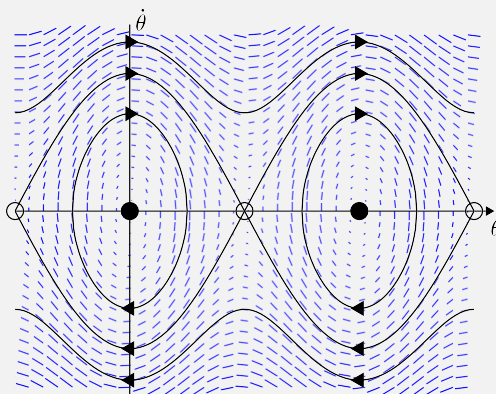
2我们花了一些时间讨论了这个二阶方程的已知解--实际上我们没有 $\vartheta(t)$ 作为初始条件的函数的闭式解。由于我们无法提供一个分析性的解决方案，在第二章中，我们求助于图形分析，并证实了这样的直觉：系统中存在 \emptyset xed点（在 $\vartheta=k\pi$ 的每个整数 k 处），在 $\vartheta=2\pi k$ 处的 \emptyset xed点是渐进稳定的，具有很大的吸引盆地。图形分析给了我们这种直觉，但我们能否真正证明这种稳定性呢？

财产？以一种可能对更复杂的系统也有效的方式？

前进的一个途径是看系统的总能量（动能+势能），我们可以把它写下来。

$$E(\vartheta, \dot{\vartheta}) = \frac{m}{2} l^2 \dot{\vartheta}^2 - mgl \cos \vartheta。$$

回顾一下，这个能量函数的轮廓是无阻尼摆的轨道。



证明下行 \emptyset xed点稳定性的一个自然途径是论证阻尼摆的能量减少（ $b>0$ ），因此系统最终会在最小能量 $E=-mgl$ 处静止，这发生在 $\vartheta=2\pi k$ 。让我们把这个论点变得稍微精确些。

评估能量的时间导数可以看出

$$\frac{d}{dt} E = -b\dot{\vartheta} \leq 0。$$

这可以证明能量永远不会增加，但实际上并不能证明当 $b>0$ 时能量会收敛到最小值，因为存在多种状态（不仅是最小值）， $\dot{E} = 0$ 。为了进行最后一步，我们必须观察到 $\dot{\vartheta} = 0$ 的状态集 \emptyset 不是一个不变的集合；如果系统处于，例如 $\vartheta = \pi$ ， $\dot{\vartheta} = 0$ 它不会停留在那里，因为 $\ddot{\vartheta} \neq 0$ 。而一旦离开该状态，能量将再次减少。事实上， \emptyset xed点是 $\dot{E} = 0$ 的状态集的唯一子集⁴，它确实形成了一个不变集。因此，我们可以得出结论，当 $t \rightarrow \infty$ 时，系统确实会在一个 \emptyset xed点静止（尽管它可以是任何能量小于或等于系统初始能量 $E(0)$ 的 \emptyset xed点）。

这是一个重要的例子。它证明了我们可以使用一个相对简单的函数--系统总能量--来描述一些关于钟摆的长期动态，尽管系统的实际轨迹（分析上）非常复杂。它还证明了使用一个非递增（而不是严格递减）的类似能量的函数来证明渐近稳定性的一个微妙之处。

李亚普诺夫函数将这个能量函数的概念推广到更普遍的系统，这些系统可能不是在某种机械能量意义上的稳定。如果我可以找到任何正函数，称之为 $V(\mathbf{x})$ ，随着时间的推移，系统的演化会越来越小，那么我就有可能用 V 来对系统的长期行为做出说明。 V 被称为**Lyapunov函数**。

回顾一下，我们为非线性系统的轴心点的稳定性定义了三个独立的概念：i.s.L.的稳定性、渐进稳定性和指数稳定性。我们可以用李亚普诺夫函数来依次证明这些概念。

定理-9.1-李亚普诺夫的直接方法

给定一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ， \mathbf{f} 是连续的，对于原点周围的某个区域 B （特别是包含原点的 \mathbb{R}^n 的一个开子集），如果我能产生一个标量的、连续二阶的函数 $V(\mathbf{x})$ ，这样

$$V(\mathbf{x}) > 0, \forall \mathbf{x} \in B \setminus \{0\}, V(0) = 0, \dot{V}(\mathbf{x}) \leq 0, \forall \mathbf{x} \in B \setminus \{0\}, \quad \dot{V}(0) = 0,$$

那么原点（ $\mathbf{x}=0$ ）在Lyapunov（i.s.L.）的意义上是稳定的。[注：符号 $B \setminus A$ 代表除去 A 元素的集合 B 。]

如果，另外，我们有

$$\dot{V}(\mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}) < 0, \quad \forall \mathbf{x} \in B \setminus \{0\}$$

则原点是（局部）渐近稳定的。而如果我们还有

$$\dot{V}(\mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}) \leq -\alpha V(\mathbf{x}), \quad \forall \mathbf{x} \in B \setminus \{0\},$$

对于某个 $\alpha > 0$ ，那么原点是（局部）指数稳定的。

请注意，对于后文，我们将使用符号 V_{\diamond} 来表示**正definite函数**，也就是说，对于所有 $\mathbf{x} \neq 0$ ， $V(0)=0$ ， $V(\mathbf{x}) > 0$ （对于正半definite，还有 $V \geq 0$ ，对于0负definite函数， $V \leq 0$ ）。

这里的直觉与我们在钟摆例子中的能量论证完全相同：因为 $\dot{V}(\mathbf{x})$ 总是零或负的，所以 $V(\mathbf{x})$ 的值只会随着时间的推移而变小（或保持不变）。在子集 B 内，对于每个 ϵ -球，我可以选择一个 δ ，使 $|\mathbf{x}(0)|^2 < \delta \Rightarrow |\mathbf{x}(t)|^2 < \epsilon$ ， $\forall t$ 通过选择 δ 足够小，使 $V(\mathbf{x})$ 的最大值的子级集在 δ 球完全包含在 ϵ 球中。由于 V 的值只能变小（或保持不变），这就给出了稳定性i. s. L。如果 \dot{V} 是严格意义上的负数。

的，那么它最终一定会到达原点（渐进稳定性）。指数条件是由以下事实暗示的： $\forall t > 0, V(\mathbf{x}(t)) \leq V(\mathbf{x}(0)) e^{-\alpha t}$ 。

注意，上面分析的系统， $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ，没有任何控制输入。因此，李亚普诺夫分析用于研究系统的被动动力学或闭环系统的动力学（系统+控制的反馈）。我们将在下文看到李亚普诺夫函数对输入-输出系统的概括。

9.1.1 全球稳定

一个平衡点是稳定的i.s.L.的概念本质上是一个局部的稳定概念（用原点周围的 ϵ 和 δ -球来定义），但渐进和指数稳定的概念可以应用于全球。李亚普诺夫定理也适用于这种情况，只需稍作修改即可。

定理9.2 - 全局稳定性的李亚普诺夫分析

给定一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ， \mathbf{f} 是连续的，如果我能够产生一个标量的、连续二等分的函数 $V(\mathbf{x})$ ，这样

$$\begin{aligned} V(\mathbf{x}) &> 0, \\ \dot{V}(\mathbf{x}) &= \frac{dV(\mathbf{x})}{dt} \leq 0 \\ V(\mathbf{x}) &\rightarrow \infty, \text{ 只要 } \|\mathbf{x}\| \rightarrow \infty. \end{aligned}$$

那么原点（ $\mathbf{x}=0$ ）是全局渐近稳定的（G.A.S.）。

如果再加上我们有

$$\dot{V}(\mathbf{x}) < -\alpha V(\mathbf{x}).$$

对于某个 $\alpha > 0$ ，那么原点是全局指数稳定的。

新的条件，关于 $\|\mathbf{x}\| \rightarrow \infty$ 的行为，被称为“**径向无界**”，是为了确保轨迹不能随着 V 的减少而发散到不确定状态；它只在全局稳定性分析中需要。

9.1.2 拉萨尔的不变量原则

也许你注意到上面的陈述和我们为摆的稳定性所做的论证之间的脱节。在摆的例子中，使用机械能的结果是Lyapunov函数的时间导数只有负的半截，但我们最终论证了平衡点是渐进稳定的。这需要做一些额外的工作，包括论证平衡点是系统在 $\dot{E} = 0$ 的情况下唯一可以停留的地方。

而其他每一个具有 $\dot{E} = 0$ 的状态都0只是瞬时的。我们可以将这一想法正式化，用于更一般的李亚普诺夫函数声明--它被称为拉萨尔定理。

定理-9.3-拉萨尔定理

给定一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ， \mathbf{f} 连续。如果我们能产生一个标量函数 $V(\mathbf{x})$ 的连续导数，对于它我们有

$$V(\mathbf{x}) > 0, \quad \dot{V}(\mathbf{x}) \leq 0$$

和 $V(\mathbf{x}) \rightarrow \infty$ ，因为 $\|\mathbf{x}\| \rightarrow \infty$ ，那么 \mathbf{x} 将收敛到最大的**不变集**。
其中 $\dot{V}(\mathbf{x}) = 0$ 。

明确地说，动态系统的**不变集** G 是指 $\mathbf{x}(0) \in G \Rightarrow \forall t > 0, \mathbf{x}(0, t) \in G$ 的集合。换句话说，一旦你进入这个集合，就永远不会离开。最大的不变集“不一定是相连的；事实上对于摆的例子，每个平衡点都是一个不变集，所以最大的不变集是系统所有平衡点的**联合**。还有一些拉萨尔定理的变种，在一个区域内起作用。

找到一个 $\dot{V} = 0$ 的李亚普诺夫函数比找到一个 $\dot{V} < 0$ 的李亚普诺夫函数更困难。LaSalle's 定理使我们有能力在这种情况下做出 渐进稳定性 的声明。在摆的例子中，每一个具有 $\dot{\vartheta} = 0$ 的状态都有 $0\dot{E} = 0$ ，但只有 $\vartheta = 0$ 点是在最大的不变集里。

例子（车杆系统的9.2摆动）。

回顾一下 使用部分反馈线性化来生成车-杆系统的摆动控制器的例子。我们首先通过写出它的能量，只考察了杆（摆）的动力学。

$$E(\mathbf{x}) = \frac{1}{2} \dot{\vartheta}^2 - \cos \vartheta。$$

渴望的能量， $E = 1$ ，以及二者之间的差异 $\hat{E}(\mathbf{x}) = E(\mathbf{x}) - E$ 。我们能够证明，我们提出的控制器产生了

$$\dot{\hat{E}} = -k \hat{\vartheta}^2 \cos^2 \vartheta。$$

其中 k 是一个我们可以选择的正增益。我们说这是很好的！现在我们有工具来理解，实际上我们有一个李亚普诺夫函数

$$V(\mathbf{x}) = \frac{1}{2} \hat{E}^2(\mathbf{x})。$$

根据LaSalle，我们只能论证

闭环系统将收敛到最大的不变量集，这里是指整个同曲线轨道： $\dot{\mathbf{x}} = \mathbf{0}$ 。

以稳定竖立。

图-9.1-李亚普诺夫函数。 $V(\mathbf{x}) = \frac{1}{2} \hat{E}^2(\mathbf{x})$

图--Lyapunov函数的9.2时间导数。 $\dot{V}(\mathbf{x})$ 。

从图中可以看出， $\dot{V}(\mathbf{x})$ 最终是一个相当不简单的函数！我们将在接下来的章节中开发计算工具来验证这种复杂系统的Lyapunov/LaSalle条件。我们将在接下来的章节中开发计算工具，以验证这种复杂性的系统的Lyapunov/LaSalle条件。

9.1.3 希尔顿-雅各比-贝尔曼方程 的关系

在这一点上，你可能会想，李亚普诺夫函数和我们在动态编程背景下讨论的成本-去向函数之间是否有任何关系。毕竟，成本-去向函数也在一个标量函数中捕获了大量关于系统长期动态的信息。如果我们重新审视HJB方程，我们就能看到这种联系

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right]$$

让我们想象一下，我们可以解出优化的 $\mathbf{u}^*(\mathbf{x})$ ，那么我们会发现

$$0 = \ell(\mathbf{x}, \mathbf{u}^*) + \frac{\partial J^*}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u}^*) \text{ 或简单地写为}$$

$$\dot{J}^*(\mathbf{x}) = -\ell(\mathbf{x}, \mathbf{u}^*) \quad \text{与} \quad \dot{V}(\mathbf{x}) < 0.$$

换句话说，在优化控制中，我们必须找到一个对每一个 \mathbf{x} 都符合这个梯度的成本-去向函数；这是非常困难的，涉及到解决一个潜在的高维偏微分方程。相比之下，李亚普诺夫分析所要求的要少得多--任何对所有状态都在走下坡路的函数（以任何速度）。这

对于理论工作，以及我们的数字算法来说，这可能要容易得多。还需要注意的是，如果我们确实设法找到了最佳去向成本 $J^*(\mathbf{x})$ ，那么只要 $\ell(\mathbf{x}, \mathbf{u}^*(\mathbf{x})) > 0$ ，它也可以作为Lyapunov函数。6

9.2 LYAPUNOV分析与凸式优化

到目前为止，我所提出的李亚普诺夫分析的主要局限性之一是，要为有趣的系统，特别是欠活动系统提出合适的李亚普诺夫函数候选，可能非常困难。（即使有人给我一个一般非线性系统的李亚普诺夫候选函数，李亚普诺夫条件也很难检查--例如，如果 \dot{V} 是某个向量 \mathbf{x} 上的任意复杂非线性函数，我如何检查 \dot{V} 对除原点外的所有 \mathbf{x} 都是严格的负数？

在这一节中，我们将看看一些验证李亚普诺夫条件的计算方法，甚至寻找李亚普诺夫函数本身的（系数）。

如果你在想象用数值算法来检查复杂的李亚普诺夫函数和复杂的动力学的李亚普诺夫条件，那么第一个想法是

可能我们可以评估 V 和 \dot{V} 在大量的样本点和检查 V 是否为正， \dot{V} 是否为负。[这确实有效](#)，并有可能与一些平滑性或规则性假设相结合，以概括出样本点以外的情况。但在许多情况下，我们将能够做得更好--提供优化算法，[对所有 \$\mathbf{x}\$ 严格检查这些条件](#)，而不需要密集抽样；这些也将给我们在制定搜索李亚普诺夫函数时提供额外的杠杆。

9.2.1 线性系统的李亚普诺夫分析

让我们花点时间来看看线性系统的情况如何。

定理-9.4-稳定线性系统的李亚普诺夫分析

想象一下，你有一个线性系统， $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ ，可以找到一个Lyapunov函数

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}, \quad \mathbf{P} = \mathbf{P}^T, \quad \mathbf{P} > 0,$$

这也满足了

$$\dot{V}(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{A} \mathbf{x} + \mathbf{x}^T \mathbf{A}^T \mathbf{P} \mathbf{x} < 0.$$

那么原点是全局渐近稳定的。

请注意，径向无界条件是由 $\mathbf{P} > 0$ 满足的，并且导数条件等同于矩阵条件

$$\mathbf{P} \mathbf{A} + \mathbf{A}^T \mathbf{P} < 0.$$

对于稳定的线性系统来说，二次李亚普诺夫函数的存在实际上是一个必要条件（也是一个充分条件）。此外，Lyapunov函数总是可以通过找到矩阵Lyapunov方程的正Definite解来找到

$$\mathbf{P} \mathbf{A} + \mathbf{A}^T \mathbf{P} = -\mathbf{Q}, \tag{1}$$

对于任何 $\mathbf{Q} = \mathbf{Q}^T > 0$ 。

这是一个非常强大的结果--对于非线性系统来说，要找到一个Lyapunov函数可能会很困难，但对于线性系统来说，这是直截了当的。事实上，这个结果经常被用来提出非线性系统的候选方案，例如，通过线性化方程和求解局部线性李亚普诺夫函数，该函数在以下情况下应该是有效的

在一个 ϵ -邻域的附近。

9.2.2 李亚普诺夫分析-- 半官方程序(SDP)

线性系统的李亚普诺夫分析与凸优化有极其重要的联系。特别是，我们也可以**用半脱机编程**（SDP）来制定上述线性系统的李亚普诺夫条件。半脱机编程是凸优化的一个子集--这是一类极其重要的问题，我们可以产生保证找到全局最优解的有效算法（在一定的数值容忍度内，不存在任何数值上的困难）。

如果你对凸优化了解不多，或者想快速复习一下，请花几分钟时间阅读附录中的优化初步知识。本节的主要要求是体会到有可能制定**efficient**优化问题，其中的约束条件包括指定一个或多个矩阵为正半定（PSD）。这些矩阵必须由决策变量的线性组合形成。对于一个微不足道的例子，优化

$$\min_{\mathbf{a}} \|\mathbf{a}\|_1, \text{ 受制于 } \mathbf{a}^T \mathbf{A} \mathbf{a} \leq 60,$$

返回一个 \mathbf{a} （ \mathbf{a} 在数字公差范围内）。

对于线性系统来说，这里面的价值是直接的。例如，我们可以通过使用参数 \mathbf{p} 填充一个对称矩阵 \mathbf{P} 来制定线性系统 $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ 的Lyapunov函数搜索，然后写出SDP。

$$\text{发现 } \mathbf{P} \text{ 受制于 } \mathbf{P} > 0, \quad \mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} < 0. \quad (2)$$

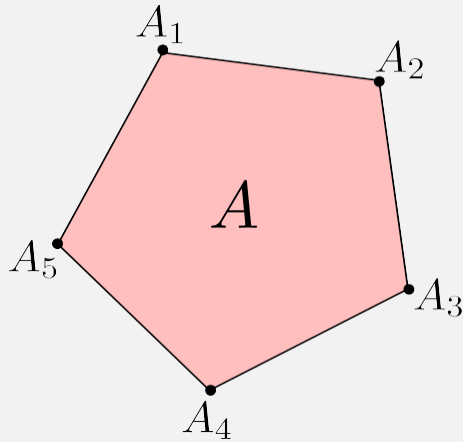
请注意，你可能永远不会使用这种特殊的表述方式，因为有专门的算法来解决简单的李亚普诺夫方程，这些算法更加有效，在数值上也更加稳定。但是SDP的表述确实提供了一些新的东西--我们现在可以很容易地表述对不确定线性系统的“**普通李亚普诺夫函数**”的搜索。

例子（线性系统的9.3常见李亚普诺夫分析）。

假设你有一个受方程 $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ 支配的系统，其中矩阵 \mathbf{A} 是未知的，但其不确定元素可以被约束。有许多方法可以写下这个不确定性集合；让我们选择通过将 \mathbf{A} 描述为一些已知矩阵的凸组合来写。

$$\mathbf{A} = \sum_i \theta_i \mathbf{A}_i, \quad \sum_i \theta_i = 1, \quad \forall i, \theta_i > 0.$$

这只是指定不确定性的一种方式；从几何学上讲，它是描述一个不确定参数的多边形（在 \mathbf{A} 的元素空间中，每个 \mathbf{A}_i 都是多边形中的一个顶点。



现在我们可以用以下方法来制定寻找一个共同的李亚普诺夫函数

$$\text{发现 } \mathbf{P} \text{ 受制于 } \mathbf{P} \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P} < -\epsilon \mathbf{I}, \quad \forall i, \quad \mathbf{P} > 0.$$

解算器将返回一个满足所有约束条件的矩阵 \mathbf{P} ，或者返回说 "问题不可行"。可以很容易地验证，如果 \mathbf{P} 满足所有顶点的李亚普诺夫条件，那么它就满足了集合中每个 \mathbf{A} 的条件。

$$\mathbf{P}(\sum_i \beta_i \mathbf{A}_i) + (\sum_i \beta_i \mathbf{A}_i)^T \mathbf{P} = \sum_i \beta_i (\mathbf{P} \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}) < 0,$$

注意，与已知线性系统的简单李亚普诺夫方程不同，这个条件的满足是一个充分条件，但不是一个必要条件。

--有可能不确定矩阵 \mathbf{A} 的集合是稳健的，但这种稳定性不能用普通的二次李亚普诺夫函数来证明。

你可以在 **DRAKE** 中自己尝试这个例子。



在Colab中打开

像往常一样，确保你打开代码并看一看。

这个结果有许多小的变种，可能会引起人们的兴趣。例如，一组非常相似的条件可以证明具有乘法噪声的线性系统的 "均方稳定性" (见例如[3], § 9.1.1).

这个例子非常重要，因为它建立了李亚普诺夫函数和 (凸) 优化之间的联系。但到目前为止，我们只证明了线性系统的这种联系，其中 PSD 矩阵为建立所有 \mathbf{x} 的 (二次) 函数的正性提供了一个神奇的秘诀。令人惊讶的是，事实证明是有的。

9.2.3 多项式系统的李亚普诺夫分析

[平方和优化](#) 提供了 SDP 在正多项式上进行优化的自然概括 (如果你不熟悉，请花点时间 [阅读附录](#))。这表明有可能将使用 SDP 搜索线性系统的李亚普诺夫函数的优化方法推广到搜索至少是多项式系统的李亚普诺夫函数： $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ ，其中 \mathbf{f} 是一个矢量值的多项式函数。如果我们将一个 V 度的 Lyapunov 参数化

候选者是一个具有未知系数的多项式，例如。

$$V(\alpha \mathbf{x}) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x x_2 + \alpha_4 x^2 + \dots, \quad \alpha \in \mathbb{R}^n$$

那么 V 的时间导数也是一个多项式，我可以制定优化方案。

$$\begin{aligned} \text{找到, 受制于 } V(\alpha \mathbf{x}) \text{ 是 SOS} \\ \dot{V}(\mathbf{x}) = -\frac{\partial V(\alpha \mathbf{x})}{\partial \mathbf{x}}(\mathbf{x}) \text{ 是 SOS。} \end{aligned}$$

因为这是一个凸式优化，如果存在一个解决方案，解算器将返回一个解决方案。

例子（9.4通过SOS验证Lyapunov候选者）。

这个例子是来自7.2[4].考虑非线性系统。

$$\begin{aligned} \dot{x}_0 &= -x_0 - 2x_1^2 \\ \dot{x}_1 &= -x_1 - x_0 x_1 - 2x_1^3 \end{aligned}$$

和 $V(x)$ Lyapunov函数 $V(x) = x_0^2 + 2x_1^2$ ，检验 $\dot{V}(x)$ 是否为负去核。

数值解决方案可以用几行代码编写，是一个凸优化。



例子（9.5车杆摆动（再次））。

在车杆摆动的例子中，我们采取了

$$V(\mathbf{x}) = \frac{1}{2} E^2(\mathbf{x}) = \left(\frac{1}{2} \dot{\theta}^2 - \cos \theta - 1 \right)^2.$$

这显然是一个平方之和。此外，我们表明，

$$\dot{V}(\mathbf{x}) = -\dot{\theta}^2 \cos \theta \leq 0.$$

这也是一个平方之和。因此，使用平方之和优化的建议实际上与非线性控制理论家多年来一直使用的配方（在纸笔上）没有什么区别。在这种情况下，我需要一个包括更多单项式的基向量。 $[1, \dot{\theta}, \cos \theta, \dots, \dot{\theta}^3, \dots]^T$ ，而且在优化中设置平等约束，当涉及到三角函数时，需要更复杂的术语匹配，但这个配方仍然有效。

例9.6（通过SOS搜索Lyapunov函数）。

验证一个候选的李亚普诺夫函数是很好的，但是当我们使用优化来验证李亚普诺夫函数的时候，真正的兴奋点就开始了。在下面的代码中，我们将 $V(x)$ 参数化为包含所有单项式的多项式，其度数为决策变量。

$$V(x) = c_0 + c_1 x_0 + c_2 x_1 + c_3 x_0 x_1 + c_4 x_0^2 + c_5 x_1^2 + \dots$$

我们将通过设置 $V(0) = 0$来设置比例（任意）以避免数字问题。 $V([1, 0]) = 1$ 。然后我们写道。

发现受制于 V 的是SOS。

\dot{V} 是sos。

 在Colab中打开

在数值收敛容限内，它发现了与我们上面选择的相同的系数（将不必要的项清零）。

重要的是要记住，有一些差距使这个解决方案的存在成为一个充分条件（用于证明 V 的每个子级集是 f 的不变集），而不是一个必要条件。首先，不能保证一个稳定的多项式系统可以用多项式Lyapunov函数来验证（任何程度的，事实上有已知的反例[...])。[5])，而在这里，我们只是在一个 \mathbb{R} 系数的多项式上进行搜索。其次，即使多项式李亚普诺夫函数确实存在，在SOS多项式和正多项式之间也存在差距。

尽管有这些注意事项，我发现这种表述在实践中出奇地有效。直观地说，我认为这是因为李亚普诺夫条件有相对多的灵活性--如果你能找到一个函数是系统的李亚普诺夫函数，那么也有许多"附近"的函数将满足同样的约束。

9.3 用于估计吸引区的李亚普诺夫函数

李亚普诺夫函数和不变集的概念之间还有一个非常重要的联系：*李亚普诺夫函数的任何子级集也是一个不变集*。这使我们有能力使用李亚普诺夫函数的子级集作为非线性系统吸引力区域的近似值。

定理9.5--李亚普诺夫不变量集和吸引区域定理

给定一个系统 $\dot{\mathbf{x}} = f(\mathbf{x})$ ， f 是连续的，如果我们能找到一个标量函数 $V(\mathbf{x})$ 和一个子水平集

$$G : \{\mathbf{x} | V(\mathbf{x}) \leq \rho\}.$$

据此

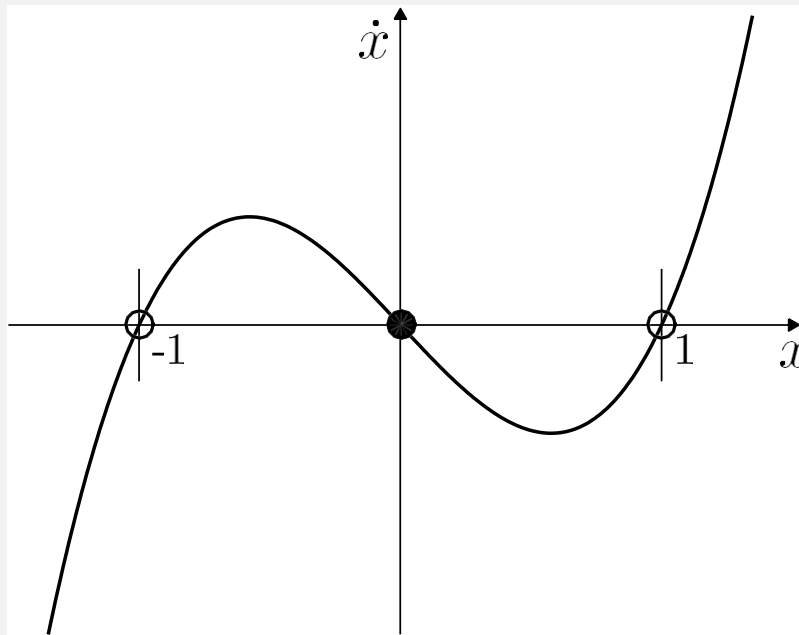
$$\forall \mathbf{x} \in G, \quad \dot{V}(\mathbf{x}) < 0.$$

则 G 是一个不变量集。根据LaSalle， \mathbf{x} 将收敛到 G 的最大不变子集，在这个子集上， $\dot{V} = 0$ 。

此外，如果 $\dot{V}(\mathbf{x}) < 0$ 在 G 中，那么原点是局部渐近稳定的，集合 G 在这个 $\mathbf{0}$ 点的吸引区域内。或者，如果 $\dot{V}(\mathbf{x}) < 0$ 在 G 中，并且 \mathbf{x}^* 是 G 中唯一的不变子集，其中 $\dot{V} = 0$ ，那么原点是渐近稳定的，并且集合 G 在这个 $\mathbf{0}$ 点的吸引区域内。

例子9.7（一维系统的吸引区域）。

考虑一阶一维系统 $\dot{x} = -x + x^3$ 。我们可以用图形分析工具快速理解这个系统。



在原点附近， \dot{x} 看起来像 $-x$ ，随着我们的移动，它看起来越来越像 x^3 。在原点有一个稳定的Fixed点，在 ± 1 处有不稳定的Fixed点。事实上，我们可以直观地推断，原点的稳定Fixed点的吸引区域是 $x \in (-1, 1)$ 。让我们看看我们是否能用李亚普诺夫论证来证明这一点。

首先，让我们将动力学围绕原点线性化，并使用线性系统的李亚普诺夫方程来确定一个候选的 $V(x)$ 。由于线性化是 $\dot{x} = -x$ ，如果我们取 $Q = 1$ ，则我们发现 $P = 1$ 是代数的正的非线性解。李亚普诺夫方程(1)继续进行

$$V(x) = \frac{1}{2}x^2$$

我们有

$$\dot{V} = x(-x + x^3) = -x^2 + x^4$$

因此，我们可以得出结论，子级集合 $V < 1$ 是不变的，而集合 $x \in (-1, 1)$ 是在非线性系统的吸引区域内。事实上，这个估计是严格的。

9.3.1 使用"普通李亚普诺夫函数"的稳健性分析

虽然我们将把大部分关于鲁棒性分析的讨论推迟到[后面的笔记中](#)，但我们在[上面的例子](#)中为线性系统简要介绍的[普通李亚普诺夫函数](#)的概念，可以很容易地扩展到非线性系统和吸引力区域分析。想象一下，你有一个动态系统的模型

但是，你对一些参数不确定。例如，你有一个四旋翼的模型，对质量和长度相当确定（这两个都很容易测量），但对惯性矩不确定。稳健性分析的一种方法是确定一个有界的不确定性，其形式可以是

$$\dot{\mathbf{x}} = f(\alpha \mathbf{x}), \quad \alpha_{min} \leq \alpha \leq \alpha_{max}.$$

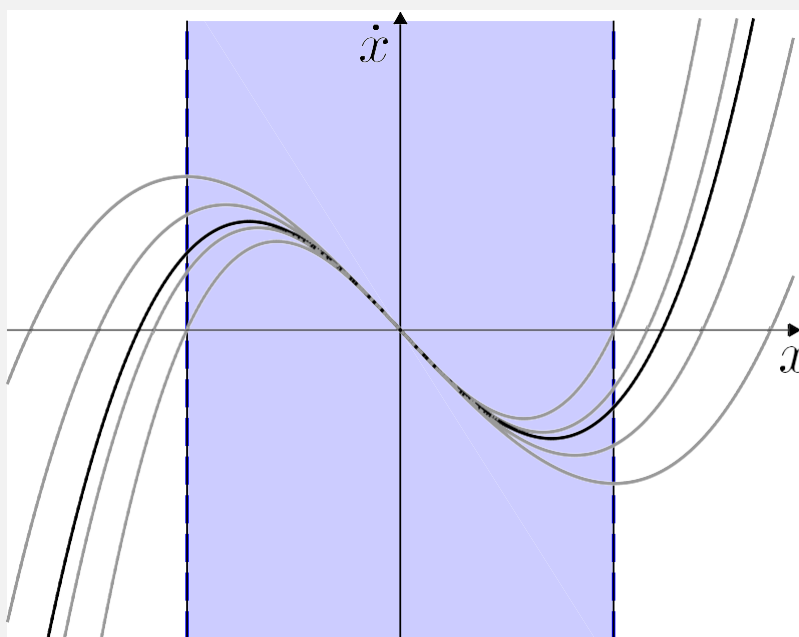
α 是你的模型中不确定参数的一个向量。更丰富的不确定性边界的规格也是可能的，但这将为我们的例子服务。

在标准的李亚普诺夫分析中，我们正在寻找一个对所有 \mathbf{x} 都走下坡路的函数，以对系统的长期动态做出陈述。在普通的李亚普诺夫分析中，如果我们能找到一个对所有可能的 α 值都走下坡路的单一李亚普诺夫函数，我们就能对一个不确定系统的长期动态做出许多类似的声明。

例子9.8(具有一维有增益不确定性的系统)

让我们考虑上面使用的同样的 $\dot{x} = -x^3$ ，但在动态中添加一个不确定的参数。具体来说，考虑系统。

绘制 α 的几个值的一维动态图，我们可以看到原点的 Oxed 点仍然是稳定的，但是吸引这个 Oxed 点的强大区域（下面的蓝色阴影）比 $\alpha=1$ 系统的吸引区域要小。



采取与上述相同的李亚普诺夫候选者， $V = \frac{1}{2}x^2$ ，我们有

$$\dot{V} = -x^2 + \alpha^4 x^2.$$

这个函数在原点为零，只要 $x^2 > \alpha^4 x^2$ ，对所有 α 都是负的，或者说

$$|x| < \frac{1}{\sqrt{\alpha_{\max}}} = \frac{\sqrt{2}}{3}$$

因此，我们可以得出结论， $|x| < \frac{\sqrt{2}}{3}$ 在稳健的吸引区域内。
的不确定系统。

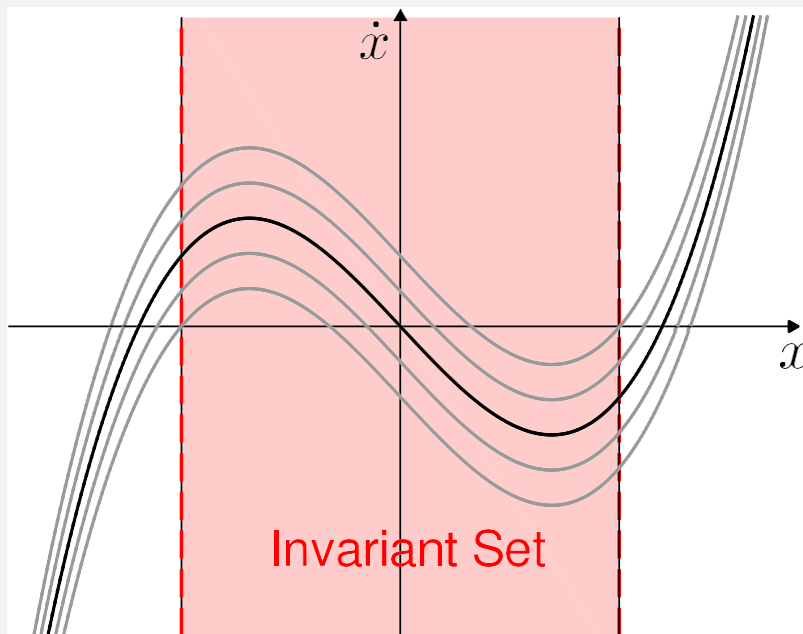
并非所有形式的不确定性都能像该例子中的增益不确定性那样简单处理。对于许多形式的不确定性，我们甚至可能不知道不确定系统的 **Fixed** 点的位置。在这种情况下，我们通常仍然可以使用普通的李亚普诺夫函数来给出一些关于系统的保证，例如 **鲁棒集不变性** 的保证。例如，如果你在一个四旋翼模型上有不确定的参数，你可能可以接受四旋翼稳定在 0.01 弧度的俯仰，但你想保证它绝对不会翻倒并撞向地面。

例子（9.9 一个具有加性不确定性的一维系统）

现在考虑一下这个系统。

$$\dot{x} = -x + x^3 + \alpha, \quad \frac{1}{4} \leq \alpha \leq \frac{1}{4}$$

绘制 α 的几个值的一维动力学图，这次我们可以看到 **Fixed** 点不一定在原点；**Fixed** 点的位置根据 α 的值而移动。但我们应该能够保证，如果不确定系统在原点附近开始，使用不变集论证，它将保持在原点附近。



采取与上述相同的李亚普诺夫候选者， $V = \frac{1}{2}x^2$ ，我们有

$$\dot{V} = -x^2 + x^4 + \alpha x$$

这个李亚普诺夫函数使我们能够轻易地验证，例如， $V \leq 1$ 是一个

稳健的不变集，因为只要 $V=1$ ，我们就有

$$\forall \alpha \in [\alpha_{min}, \alpha]_{max} \circ \quad \dot{V}(x, \alpha) < 0.$$

因此， V 永远不可能从小于三分之一的地方开始，并跨越到大于三分之一的地方。要看到这一点，请看

$$V = \frac{1}{3} \Rightarrow x = \pm \sqrt{\frac{2}{3}} \Rightarrow \dot{V} = -\frac{2}{9} \pm \alpha \sqrt{\frac{2}{3}} < 0, \forall \alpha \in [-\frac{1}{4}, \frac{1}{4}].$$

请注意，并非这个不变集的所有子级集都是不变的。例如 $V \leq \frac{1}{32}$ 不满足这个条件，通过目测，我们可以看到，它实际上不是稳健不变的。

9.3.2 多项式系统的吸引力区域估计

现在我们已经到了我认为可以成为许多严肃的机器人应用的工作马的工具。我们的大多数机器人实际上并不是全局稳定的（这并不是因为它们是机器人--如果你用力推我，我也会倒下），这意味着了解特定控制器可以保证工作的区域可能是至关重要的。

平方和优化实际上给了我们一个神谕，我们可以问：这个多项式对所有 \mathbf{x} 都是正的吗？为了将其用于区域分析，我们必须弄清楚如何修改我们对神谕的问题，以便当我们问一个函数在某个区域（ R^n 的一个子集）上是否为正时，神谕会说"是"或"不是"。这个技巧被称为S程序。它与约束优化中的拉格朗日乘数密切相关，并与代数几何中的"正数"有深刻联系。

S-程序

考虑一个标量多项式 $p(\mathbf{x})$ 和一个半代数集 $g(\mathbf{x}) < 0$ ，其中 g 是多项式的一个向量。如果我能找到一个多项式"乘数"， $\lambda(\mathbf{x})$ ，使得

$$p(\mathbf{x}) + \lambda^T(\mathbf{x})g(\mathbf{x}) \text{ 是SOS,} \quad \text{而} \lambda(\mathbf{x}) \text{ 是SOS。}$$

那么这就足以证明

$$p(\mathbf{x}) \geq 0 \forall \mathbf{x} \in \{\mathbf{x} | g(\mathbf{x}) \leq 0\}.$$

为了说服自己，观察一下，当 $g(\mathbf{x}) \leq 0$ 时，只有更难为正，但当 $g(\mathbf{x}) > 0$ 时，即使 $p(\mathbf{x})$ 为负，组合函数也有可能为SOS。我们有时会觉得使用简写的方法很方便。

$$g(\mathbf{x}) \leq 0 \Rightarrow p(\mathbf{x}) \geq 0$$

表示由S程序证明的暗示（例如，"只要 $g(\mathbf{x}) \leq 0$ ，就有 $p(\mathbf{x}) \geq 0$ "）。

我们还可以处理平等约束，只需稍作修改--我们不再要求乘数为正数。如果我能找到一个多项式"乘数"， $\lambda(\mathbf{x})$ ，使得

$$p(\mathbf{x}) + \lambda^T(\mathbf{x})g(\mathbf{x}) \text{ 是SOS}$$

那么这就足以证明

$$p(\mathbf{x}) \geq 0 \forall \mathbf{x} \in \{\mathbf{x} | g(\mathbf{x}) = 0\}.$$

这里的直觉是， $\lambda(x)$ 可以添加任意的正项来帮助我成为SOS，但这些项恰恰在 $g(x)=0$ 。

吸引区的基本表述

S程序为我们提供了只在状态空间的一个区域内评估正性所需的工具，这正是我们为引力区域分析认证李亚普诺夫条件所需的。让我们从一个正多项式的李亚普诺夫候选人 $V(\mathbf{x}) \succ 0$ 开始，然后我们可以写出李亚普诺夫条件。

$$\dot{V}(\mathbf{x}) \preceq 0 \forall \mathbf{x} \in \{\mathbf{x} | V(\mathbf{x}) \leq \rho\}.$$

使用平方和和S程序。

$$-\dot{V}(\mathbf{x}) + \lambda(\mathbf{x})(V(\mathbf{x}) - \rho) \text{ 是 SOS, 而 } \lambda(\mathbf{x}) \text{ 是 SOS.}$$

其中 $\lambda(\mathbf{x})$ 是一个具有自由系数的乘法多项式，在优化中要解决这些系数。

我认为在一个例子中最容易看到细节。

例子9.10（一维立体系统 的

牵引区域）。

让我们回到上面的例子。

$$\dot{x} = -x + x^3$$

并尝试用SOS优化来证明原点的Fixed点的吸引区域是 $x \in (-1, 1)$ ，使用Lyapunov候选数 $V=x^2$ 。


首先，对乘数多项式进行定义。

然后确定优化

发现
受制于

$\lambda(x) = c_0 + cx_1 + cx_2^2。$
 $\dot{V}(x) + \lambda(x)(V(x) - 1)$ 是 SOS
 $\lambda(x)$ 是 SOS

你可以在DRAKE中自己尝试这个例子。

 在Colab中打开

在这个例子中，我们只验证了预先设定的Lyapunov候选人的一个子级集是负的（证明了我们已经理解的ROA）。更有用的是，如果你能搜索出能满足这些条件的最大的 ρ 。不幸的是，在这个Fixed公式中，直接优化 ρ 会使优化问题变得不凸，因为我们会像 $\rho c_0, \rho cx_1, \dots$ 这样的条款在决策变量中是双线性的；我们需要平方和约束只在决策变量中是线性的。

幸运的是，由于问题是凸的， ρ 是Fixed的（因此可以可靠地解决），而 ρ 只是一个标量，我们可以对 ρ 进行简单的直线搜索，找到凸优化返回的最大值，从而得到可行的解决方案。这将是我们对吸引区的估计。

这个基本表述有许多变化；我将在下面描述其中的几个。还有一些重要的想法，如重新调整比例和度数匹配，这些想法都是对的。

可以对问题的数值产生巨大的影响，并有可能使它们对求解者更有利。但你不需要为了有效地使用这些工具而全部掌握它们。

例如（德雷克的吸引力9.11区域代码）。

在DRAKE中，我们已经将建立和解决吸引区域的平方和优化的大部分工作打包到一个方法`RegionOfAttraction(system, context, options)`中。这使得它变得很简单，比如说。

```
x = Variable("x")
sys = SymbolicVectorSystem(state=[x], dynamics=[-x+3x**2])
context = sys.CreateDefaultContext()
V = RegionOfAttraction(sys, context)
```



在Colab中打开

请记住，尽管我们已经努力使这些函数的调用变得方便，但它们并不是一个黑盒子。我强烈建议打开`RegionOfAttraction`方法，了解它的工作原理。有很多不同的选项/公式，以及大量的数值配方来改善优化问题的数值。

平等约束的表述

这里有一个重要的变化，用于确定候选吸引区域的水平集，它将S程序中的不等式变成了等式。这个公式在 $\lambda(\mathbf{x})$ 和 ρ 中是共同凸的，所以我们可以可以在一个单一的凸优化中优化它们。它非正式地出现在[4]中非正式地出现过，并在[6]。

在假设 $\dot{V}(\mathbf{x})$ 的Hessian在原点是负-definite（这很容易检查）的情况下，我们可以写出

$$\max_{\rho, \lambda(\mathbf{x})} \text{受} (\mathbf{x}^T \mathbf{V}^d(\mathbf{x}) - \rho) + \lambda(\mathbf{x}) \dot{V}(\mathbf{x}) \text{ 是 SOS。}$$

d 是一个给定的正整数。正如你所看到的， ρ 不再与 $\lambda(\mathbf{x})$ 的系数相乘。但是为什么这证明了一个吸引区呢？

你可以把平方之和约束理解为证明以下的含义

无论何时 $\dot{V}(\mathbf{x}) = 0$ ，我们知道，要么 $V(\mathbf{x}) \geq \rho$ ，要么 $\mathbf{x} = 0$ 。乘以一些

$\mathbf{x}^T \mathbf{x}$ 的倍数只是处理 " $\mathbf{x}=0$ "情况的一种巧妙方法，这是必要的。

因为我们预计 这一隐含关系足以证明 $\dot{V}(\mathbf{x}) \leq 0$

只要 $V(\mathbf{x}) \leq \rho$ ，因为 V 和 \dot{V} 是光滑多项式；我们在[一个练习中](#)详细研究这个说法。

用S程序代替不等式也有一个潜在的好处，即消除了对 $\lambda(\mathbf{x})$ 的SOS约束。但这种提法的最大优点也许是可以利用这个代数品种的商环来大大简化问题，特别是最近一些利用抽样品种进行精确验证的结果[6]。

搜索 $V(\mathbf{x})$

到目前为止，机械已经使用优化来找到最大的吸引力区域，该区域可以在给定一个候选的Lyapunov函数的情况下被证明。这不一定是

坏的假设。对于大多数稳定的 \mathcal{O}^* 点，我们可以用线性分析来证明局部稳定性，这个线性分析给了我们一个候选的二次Lyapunov函数，可以在一个区域内用于非线性分析。实际上，当我们解决一个LQR问题时，来自LQR的代价是一个很好的非线性系统的候选Lyapunov函数。如果我们只是分析一个现有的系统，那么我们可以通过解决李亚普诺夫方程 (Eq 1)。

但是，如果我们相信系统是区域稳定的，尽管有一个不确定的线性化？或者我们可以通过使用度数大于2的Lyapunov候选者来证明更大的状态空间量。在吸引力区域的情况下，我们是否也可以用平方和优化来找到这一点？

为了达到这个目的，我们现在将使 $V(x)$ 成为一个多项式（某种 \mathcal{O}^* 度），其系数作为决策变量。首先，我们需要添加约束条件以确保

$$V(0) = 0, \quad V(x) - \epsilon x^T x \text{ 是 SOS。}$$

其中 ϵ 是某个小的正常数。这个 ϵ 项只是确保 V 是严格的正非线性的。现在让我们考虑一下我们的基本表述。

$$-\dot{V}(x) + \lambda(x)(V(x) - 1) \text{ 是 SOS, 而 } \lambda(x) \text{ 是 SOS。}$$

注意，我已经替换了 $\rho=1$ ；现在我们正在搜索 V ，水平集的规模可以随意设置为 1。事实上，这样做更好--如果我们不设置 $V(0) = 0, V(x) \leq 1$ 子水平集，那么优化问题就会被约束不足，并可能给求解器带来问题。

不幸的是，现在决策变量中的导数约束是不凸的（双线性），因为我们要寻找 λ 和 V 的系数，而且它们是相乘的。我们的平等约束公式也不能解决这个问题（因为 V 的系数也出现在 \dot{V} 中）。在这里，我们不得不求助于某种较弱的优化形式。在实践中，我们使用双线性交替法取得了很好的实际效果：从初始 V 开始（例如从LQR开始），搜索 λ ；然后 $\mathcal{O}^* \lambda$ ，搜索 V ，重复直到收敛（例如，见[7, 8]）。一个典型的目标选择是使状态空间中吸引区的一些凸代理量最大化；通常我们使用描述椭圆的二次方的行列式。

如果没有数字问题，这个算法保证有递归的可行性，并在短短的几次交替中趋于收敛，但不能保证它找到最优解。

示例（9.12搜索李亚普诺夫函数）。

(详情即将公布...)



在Colab中打开

ROA的凸面外部近似值

到目前为止，我们所有的吸引力区域近似都是“内部近似”--保证所确定的区域包含在系统的真正吸引力区域内。如果我们要提供一些稳定性的保证，这就是我们想要的。随着我们增加多项式和乘数的度数，我们期望估计的吸引区域将收敛于真实区域。

事实证明，如果我们转而考虑外部近似，从另一侧收敛于真实的ROA，我们可以写出能够直接搜索Lyapunov函数的公式，作为一个凸优化。正如我们将在下面看到的，这种方法也允许人们为控制器设计写出凸式公式。这些方法在Henrion和Korda的工作中已经得到了很好的探讨

(例如：[9])，使用矩的方法，从[10]，也被称为“占领措施”。他们的处理方法强调了有限元线性程序和LMI松弛的层次；这些只是我们在这里写的SOS优化的对偶形式。我认为SOS的形式更清晰，所以在这里将坚持使用它。一些占领措施的论文包括无定维线性程序的对偶公式，这看起来非常相似；我和我的合作者通常在我们关于这个主题的论文中指出平方之和的版本（例如[11]）。

为了找到一个外部近似值，我们用一组非常相关的条件来寻找类似李亚普诺夫的“障碍证明” $B(x)$ ，而不是求解李亚普诺夫函数来保证向原点收敛。像李亚普诺夫函数一样，我们希望 $\dot{B}(x) \leq 0$ ；这次我们要求这一点在任何地方都是真的（或者至少在某个感兴趣的ROA足够大的集合中）。如果我们能找到这样一个函数，那么任何 $B(x) < 0$ 的状态肯定都是在ROA之外。

\emptyset ed点的吸引区域--因为 B 不能增加，所以它永远无法到达 $B > 0$ 的原点。超水平集 $\{x | B(x) \geq 0\}$ 是真正吸引区域的外部近似值。

$$-\dot{B}(x) \text{ 是 SOS, } B(0) > 0。$$

为了找到最小的这种外部近似（给定一个 \emptyset ed degree的多项式），我们选择一个试图“压低” $B(x)$ 值的目标。我们通常通过引入另一个多项式函数 $W(x)$ 来实现这一目标，其要求是

$$W(x) \geq 0, \text{ 且 } W(x) \geq B(x) + 1,$$

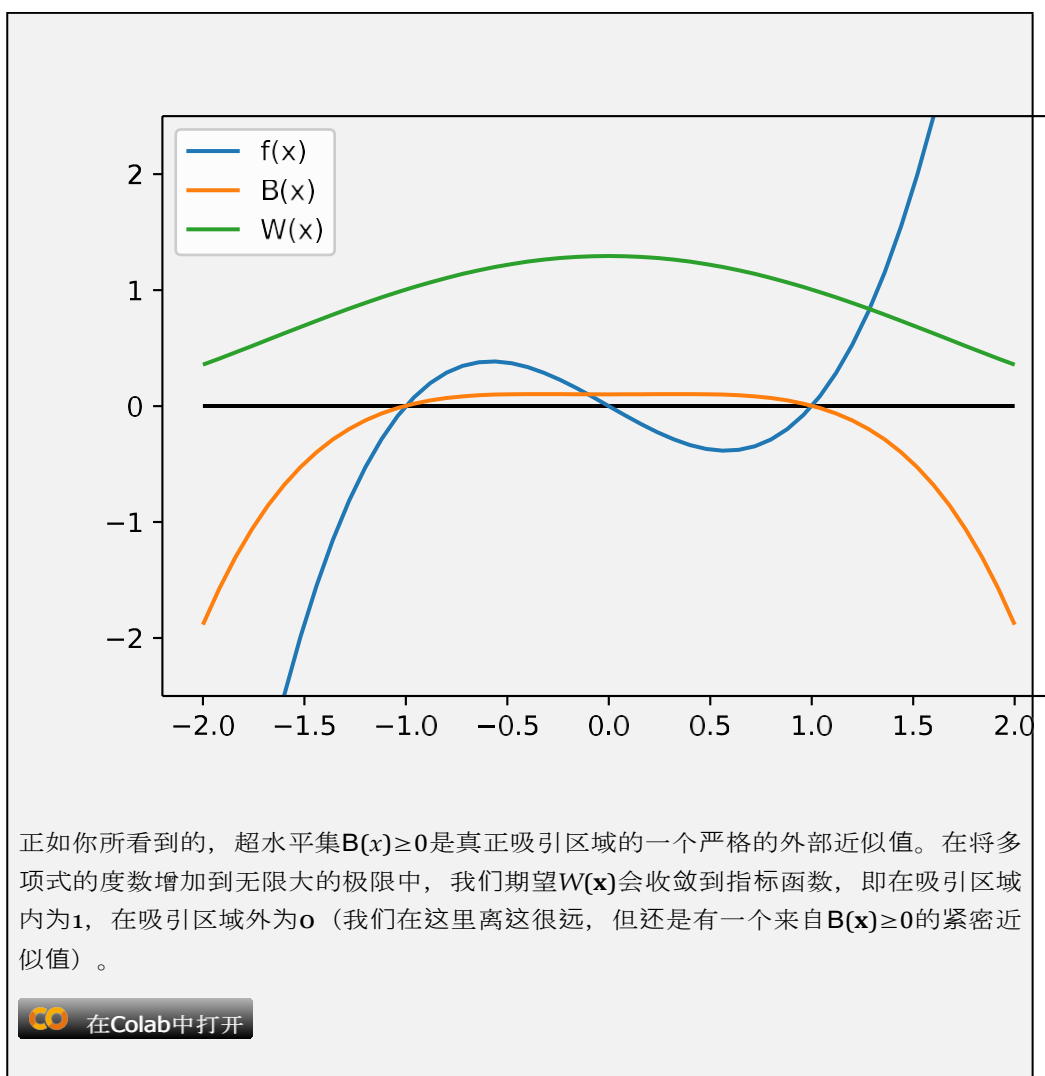
实现为SOS约束。然后我们最小化积分 $\int_x W(x) dx$ ，它可以很容易地在一个紧凑的集合上计算，如 R^n 中的球[12]，并且在系数上是线性的。也存在更复杂的替代方案[13]。

例子9.13（立方体多项式的外近似）。

让我们再来看看我最喜欢的简单动力系统， $\dot{x} = -x + x^3$ ，它在 $\{x = |x| < 1\}$ 的零点处有一个 \emptyset ed点吸引区域。这一次，很好地估计了使用外部近似法计算ROA。我们可以通过以下程序来实现这一目标。

$$\begin{array}{ll} \min_{B(x), W(x)} & \int_{-2}^2 W(x) dx \\ \text{受制于} & \begin{array}{ll} \dot{B}(x) & \text{是 SOS} \\ W(x) & , \text{ 是} \\ W(x) - B(x) - 1.0 & \text{SOS,} \\ B(0) \geq 0. & \text{是 SOS} \end{array} \end{array}$$

为了使这个问题在数字上更有说服力，你会在代码中看到，我要求 $\dot{B}(x)$ 是严格的负非线性，因为 $B(0) \geq 0.1$ 而且我选择在 $B(x)$ 和 $W(x)$ 中只包含偶数级单项式。绘制解决方案可以看出。



9.4 有限时间可及性

到目前为止，我们已经用李亚普诺夫分析法来分析稳定性，从根本上说，这是对系统在时间上走向不确定的行为的描述。但李亚普诺夫分析也可以用来分析非线性系统的有限时间行为。在接下来的几章中，我们将看到这种有限时间分析的一些应用。它甚至可以应用于不稳定的系统，或者稳定在极限周期而不是固定点的系统。它还可以应用于只在一个有限时间间隔内定义的系统，比如我们在执行一个有边界的计划轨迹时就可能是这样。

9.4.1 时变动力学和Lyapunov函数

在关注有限时间之前，让我们首先认识到，基本的（有限时间）Lyapunov分析也可以应用于时变系统： $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ 。我们可以分析这个系统的稳定性（局部、区域或全局），变化不大。如果我们

$$\forall t, \forall \mathbf{x} \neq 0, \quad \begin{aligned} & V(\mathbf{x}) > 0, \\ & \dot{V}(t, \mathbf{x}) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{x}} \mathbf{f}(t, \mathbf{x}) < 0, \quad \dot{V}(t, 0) = 0, \end{aligned}$$

那么我们之前的所有声明仍然成立。在我们的SOS公式中， t 只是多了一个不确定因素。

同样，即使对于一个时间不变的系统，也有可能确定一个时间变化的李亚普诺夫函数 $V(t, \mathbf{x})$ ，并使用几乎相同的条件建立局部、区域或全局稳定性。

$$\forall t, \forall \mathbf{x} \neq 0. \quad V(t, \mathbf{x}) > 0, \quad V(t, 0) = 0, \\ \forall t, \quad \forall \mathbf{x} \neq 0. \quad \dot{V}(t, \mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}) + \frac{\partial V}{\partial t} < 0, \quad \dot{V}(t, 0) = 0.$$

这两个想法各自独立，但它们经常一起出现，因为时变动力学也许是研究时变李亚普诺夫函数的最佳动机。毕竟，我们确实知道，一个稳定的时变系统必须有一个证明这种稳定性的时变李亚普诺夫函数（来自“逆向李亚普诺夫函数”定理）。但我们并不知道如何表示这个函数；作为一个例子，记得我们知道有一些稳定的多项式系统不能用多项式李亚普诺夫函数来验证。对于全局稳定性分析，时变李亚普诺夫分析并没有增加建模能力：因为条件必须对所有 t 都满足，我们可以直接把 t 设为常数，并使用时变函数。但在有些情况下，时间变化分析可以为区域或局部稳定情况提供不同的分析。当我们研究极限循环时，我们会看到一个很好的例子。

9.4.2 有限时间可及性

有限时间可达性分析是控制设计和验证的一个重要概念，在这里我们试图了解一个系统在仅有的有限时间间隔 $[t_1, t_2]$ 内的行为。它几乎总是基于区域的分析，并试图做出以下形式的声明。

$$\mathbf{x}(t_1) \in X_1 \Rightarrow \mathbf{x}(t_2) \in X_2.$$

其中 $X_1, X_2 \subset \mathbb{R}^n$ 是状态空间的区域。更一般地说，我们可能想了解时变的可达集 $\forall t \in [t_1, t_2], X(t)$ 。

可达性分析可以在时间上向前进行：我们选择 X_1 并尝试找到**最小的** X_2 ，对于该区域我们可以使声明成立。 $X(t)$ 被称为**前向可达集(FRS)**，它对于证明运动计划等非常有用。例如，你可能想证明你的无人机在接下来的5秒钟内不会撞上一棵树。在这种情况下， X_1 可以被认为是代表飞行器当前状态的一个点，或者是代表当前状态的外部近似的一个区域，如果状态不确定的话。在这种情况下，我们会把 X 称为前向可达集。在这个用例中，我们通常会在我们的李亚普诺夫分析中选择任何近似值来证明对可达区域的估计也是一个外部近似值。 $X_2 \subseteq \hat{X}_2$ 。

可达性分析也可以在时间上向后进行：我们选择 X_2 并试图使声明可以被证明成立的区域 **X 最大化**。现在 $X(t)$ 被称为**后向可达集(BRS)**，为了稳健起见，我们通常试图证明我们的估计是一个内部近似值， $\hat{X}_1 \subseteq X_1$ 。

X_2 取为固定点， $t_2=0$ 和 $t_1=-\infty$ 。但是，无时间的BRS也有重要作用，例如，当我们

将多个控制器组合在一起时，在

以实现一个更复杂的任务，我们将很快研究这个问题。

9.4.3 通过李亚普诺夫函数的可达性

李亚普诺夫函数可用于证明无时间可及性，即使是连续时间系统。基本的方法是在一个（可能是时间变化的）不变集上证明李亚普诺夫条件。再次，我们通常将其表示为

一个包含原点的李亚普诺夫函数的（时变的）水平集， $V(\mathbf{x}) \leq \rho(t)$ 。其中 $\rho(t)$ 现在是时间的正标量函数。由于我们已经把时间作为一个决策变量，我们也可以很容易地适应时变动力学和李亚普诺夫函数，所以。

$$\forall t \in [t_1, t_2], \quad \forall \mathbf{x}, \quad V(t, \mathbf{x}) > 0, \quad V(t, 0) = 0$$

$$\forall t \in [t_1, t_2], \quad \forall \mathbf{x} \in \{\mathbf{x} | V(t, \mathbf{x}) = \rho(t)\}, \quad \dot{V}(t, \mathbf{x}) < \dot{\rho}(t)。$$

请注意，**Onite-time**可达性是为了证明集合的不变性，而不是稳定性， \dot{V} 条件只需要在水平集合的边界上得到证明。如果 V 在边界上是递减的，那么轨迹就不可能离开。我们当然可以要求更多--我们可能想证明系统正在向 $V(t, \mathbf{x}) = 0$ 收敛，甚至可能以某种速度收敛--但只需要不变性来证明可达性。

同样，对于多项式系统和动力学来说，平方和优化是一个强大的工具，可以用数字证明这些特性，并优化估计区域的体积。在把 t 作为一个不确定因素后，我们可以再次使用**S**-程序来证明条件 $\forall t \in [t_1, t_2]$ 。

就像吸引力区域的情况一样，我们有很多表述方式。我们可以认证一个现有的李亚普诺夫候选体 $V(t, \mathbf{x})$ ，然后只需尝试最大化/最小化 $\rho(t)$ 。或者我们也可以搜索 $V(t, \mathbf{x})$ 的参数。同样，我们可以使用时间变化的李亚普诺夫方程，或时间变化的**LQR**李嘉诚方程的解来初始化这一搜索。

在实践中，我们经常只在**Onite**的样本集 $t_i \in [t_1, t_2]$ 上证明 \mathbf{x} 的Lyapunov条件。实际上，我并不反对在一维内取样；在扩展到更高维度时没有任何问题，而且可以对取样误差的约束做出实际的严格声明。在这些系统中，将 t 加入到所有的方程中，可以极大地提高**SOS**认证所需的多项式的程度。所有这些都很好地写在[14]中写得很好，其稳健的变体被开发在[158]。

9.5 刚体动力学是（有理）多项式的

在这一章中，我们已经谈了很多关于多项式系统的数值方法。但即使是我们的简单摆，在动力学上也有一个 $\sin \vartheta$ 。我是不是在浪费你的时间？我们必须求助于非多项式方程的多项式近似吗？事实证明，我们的多项式工具可以对几乎所有+的机器人的操纵方程进行精确分析。我们只需要再做一些工作来揭示这个结构。

这方面最明显的例外是如果你的机器人有螺旋形的螺钉连接（见[16]

让我们先观察一下，刚体运动学是多项式的（除了螺旋形关节）。这一点很重要--"刚体"假设的本质是身体上各点之间的欧几里得距离得以保留；如果 \mathbf{p}_1 和 \mathbf{p}_2 是身体上的两个点，那么运动学就会强制要求 $|\mathbf{p}_1 - \mathbf{p}_2|^2$ 是常数--这些是多项式约束。当然，我们通常用 $\sin \vartheta$ 和 $\cos \vartheta$ 来写运动学的最小坐标。但由于刚体假设，这些术语只以最简单的形式出现，我们可以简单地制造新变量 $s_i = \sin \vartheta_i, c_i = \cos \vartheta_i$ ，并加上 $s_i^2 + c_i^2 = 1$ 的约束。对于一个更彻底的讨论见，例如，[16]和[17]。由于多体系统的势能只是身体上所有点的重量乘以垂直位置的累积，因此势能是多项式的。

如果我们的机器人的构型（位置）可以用多项式描述，那么速度也可以：前向运动学 $\mathbf{p}_i = f(\mathbf{q})$ 意味着 $\dot{\mathbf{p}}_i = \dot{\mathbf{q}} \mathbf{f}'$ ，这是多项式的 $s, c, \dot{\vartheta}$ 。由于我们机器人的动能是由所有质量的动能的积累， $T = \sum_i \frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i$ ，动能能量也是多项式的（即使我们用惯性矩阵和角速度来写它）。

最后，[运动方程](#)可以通过对拉格朗日（动能减去势能）的导数得到。这些导数仍然是多项式的！

例9.14（通过SOS的简单摆的全局稳定性）。

我们在这一章的开头用我们对能量的直觉来讨论简单摆的稳定性。现在我们将用凸优化来取代这种直觉（因为它也适用于我们的直觉失效的更多二元系统）。

让我们把坐标从 $[\vartheta, \dot{\vartheta}]^T$ 改为 $\mathbf{x}=[s, c, \dot{\vartheta}]^T$ ，其中 $s \equiv \sin \vartheta$ ， $c \equiv \cos \vartheta$ 。然后我们可以把摆的动力学写成

$$\dot{\mathbf{x}} = \begin{bmatrix} c\dot{\vartheta} \\ -s\dot{\vartheta} \\ -\frac{1}{\ell}(b\dot{\vartheta} + mgl s) \end{bmatrix}.$$

现在让我们把Lyapunov候选者 $V(s, c, \dot{\vartheta})$ 参数化，作为包含所有2度以下单项式的未知系数的多项式。

$$V = \alpha_0 + \alpha_1 s + \alpha_2 c + \dots \alpha_9 s^2 + \alpha_{10} s c + \alpha_{11} \dot{\vartheta}.$$

现在我们将制定可行性问题。

$$\begin{array}{l} \text{发现} \\ \alpha \end{array} \quad \text{受制于 } V \text{ 是 SOS。} \quad -\dot{V} \text{ 是 SOS。}$$

事实上，这要求太高了 -- 实际上， \dot{V} 只需要在 $s^2 + c^2 = 1$ 时为负数。¹ 我们可以用S程序来完成这个任务，而写成

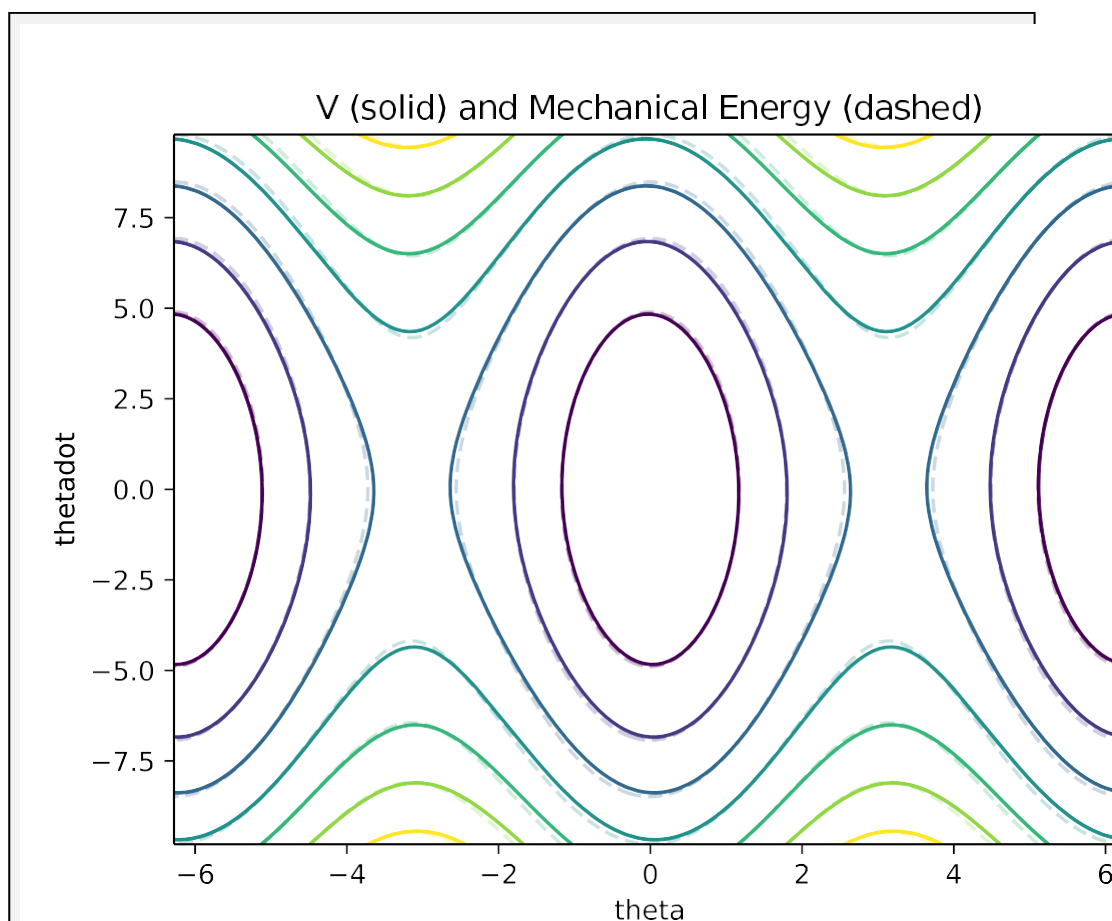
$$\begin{array}{l} \text{发现} \\ \alpha, \lambda \end{array} \quad \text{受制于 } V \text{ 是 SOS。} \quad -\dot{V} - \lambda(\mathbf{x})(s^2 + c^2 - 1) \text{ 是 SOS。}$$

(回顾一下， $\lambda(\mathbf{x})$ 是另一个具有自由系数的多项式，当 $s^2 + c^2 \neq 1$ 时，优化可以用来使条款任意地更正。)最后，对于风格点，在DRAKE的代码例子中，我们要求指数稳定



[在Colab中打开](#)

像往常一样，请确保你打开代码看一看。其结果是一个看起来很熟悉的李亚普诺夫函数（在此以等高线图的形式可视化）。



啊哈！不仅优化找到了满足我们李亚普诺夫条件的李亚普诺夫函数的系数，而且结果看起来很像机械能。事实上，这个结果比能量好一点.....有一些额外的小项，证明了指数的稳定性，而不需要引用拉萨尔定理。

单自由度摆锤确实让我们忽略了一个重要的细节：虽然操纵器方程 $M(q)\ddot{q} + C(q, \dot{q})\dot{q} = \dots$ 是多项式的，为了解决 \ddot{q} ，我们实际上必须将两边都乘以 M^{-1} 。不幸的是，这 **不是** 一个多项式运算，所以事实上多体系统的最终动力学是 **有理** 多项式。不仅如此，不建议用符号评估 M^{-1} —方程很快就变得非常复杂。但我们实际上可以用隐含形式的动力学写出李亚普诺夫条件，例如写出 $V(q, \dot{q}, \ddot{q})$ 并要求它

以满足各地的李亚普诺夫条件，即 $M(q)\ddot{q} + \dots = \dots + Bu$ 是满足的，使用 S 程序。

例子（9.15 以隐式形式验证动力学原理）

通常情况下，我们以 $\dot{x} = f(x, u)$ 的形式来写二阶梯方程。但让我们暂时考虑一下动力学形式的情况

$$g(x, u, \dot{x}) = 0.$$

这种形式严格说来更为通用，因为我总是可以写出 $g(x, u, \dot{x}) = f(x, u) - \dot{x}$ ，但重要的是在这里我也可以把 g 的底行写成

$M(q)\ddot{q} + C(q, \dot{q})\dot{q} - \tau_s = Bu$ 。这种形式也可以代表 二阶代数的

DAEs比ODEs更普遍； \mathbf{G} 甚至可以包括代数约束，如 $s^2 + c^2 = 1$ 。最重要的是，对于操纵者来说， \mathbf{g} 可以是多极的，即使 \mathbf{f} 是有理的。**DRAKE**提供了访问

隐

式的连续动力学，与之相对

[`CalcImplicitTimeDerivativesResidual`](#)方法。

有趣的是，我们可以直接在系统上检查李亚普诺夫条件 $\dot{V}(\mathbf{x}) \leq 0$ ，其隐含形式为 $\mathbf{g}(\mathbf{x}, \mathbf{z}) \leq 0$ ， $\forall \mathbf{x}, \mathbf{z} \in \mathcal{X}$ 。如果我们能用SOS证明 $Q(\mathbf{x}, \mathbf{z}) \leq 0$ ， $\forall \mathbf{x}, \mathbf{z} \in \mathcal{X}$ ，那么我们就已经验证了 $\dot{V}(\mathbf{x}) \leq 0$ ，尽管付出了增加不确定数 \mathbf{z} 和额外的S程序的非微不足道的代价。

有几件事**确实**打破了这种干净的多项式世界观。例如，旋转弹簧，如果建模为 $\tau = k(\vartheta_0 - \vartheta)$ 将意味着 ϑ 与 $\sin \vartheta$ 和 $\cos \vartheta$ 一起出现，而 ϑ 和 $\sin \vartheta$ 之间的关系很遗憾**不是多项式**。LQR的线性反馈实际上看起来像线性弹簧，尽管将反馈写成 $u = -\mathbf{K} \sin \vartheta$ 是一个可行的选择。

在实践中，你也可以用多项式对任何平滑非线性函数进行泰勒近似。这在实践中是一种有效的策略，因为你可以限制多项式的度数，而且在许多情况下，有可能对近似的误差提供保守的界限。

一个值得了解的**0nal**技术是坐标的改变，通常被称为**立体投影**，它提供了一个坐标系，在这个坐标系中，我们可以用多项式取代正弦和余弦。

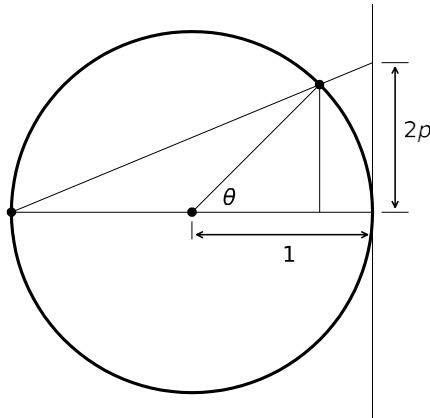


图-9.7-立体图投影。

通过投射到直线上，并使用类似的三角形，我们发现 $p = \frac{\sin \vartheta}{1 + \cos \vartheta}$

求解 $\sin \vartheta$ 和 $\cos \vartheta$ ，可以发现

$$\sin \vartheta = \frac{2p}{1+p^2}, \quad \cos \vartheta = \frac{1-p^2}{1+p^2} \quad \text{和} \quad \frac{\partial p}{\partial \vartheta} = \frac{1-p^2}{2}$$

其中 $\frac{\partial p}{\partial \vartheta}$ 可用于链式规则，以推导出动力学的 \dot{p} 。虽然

方程是有理的，它们共享分母 $1+p^2$ ，可以用质量矩阵的形式有效地处理。与 $s = \sin \vartheta$ 和 $c = \cos \vartheta$ 的简单替换相比，这是一个最小的表示法（标量对标量，不需要 $s^2 + c^2 = 1$ ）；不幸的是，它在 $\vartheta = \pi$ 处有一个奇点，所以可能不能用于全局分析。

9.6 控制设计

在本章中，我们已经开发了一些非常强大的工具，用于推理闭环系统的稳定性（已经有了控制器的规格）。我希望你一直在问自己--我可以用这些工具来设计更好的控制器吗？当然，答案是“可以”。在本节中，我将讨论控制方法，这些方法是李亚普诺夫函数的凸优化方法的最直接结果。另一个非常自然的想法是在反馈运动规划算法的内容中使用这些工具，这也是[即将到来的一章](#)的主题。

9.6.1 线性系统的状态反馈

让我们重新审视公式中线性系统的李亚普诺夫条件。[2](#)但现在加入一个线性状态反馈 $\mathbf{u} = \mathbf{K}\mathbf{x}$ ，导致闭环动力学 $\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{BK})\mathbf{x}$ 。我们可以证明，所有稳定的 \mathbf{K} 的集合可以由以下两个矩阵不等式来描述[\[.3\]](#):

$$\mathbf{p} = \mathbf{p}^T \succ 0, \quad \mathbf{p}(\mathbf{a} + \mathbf{bk}) + (\mathbf{a} + \mathbf{bk})^T \mathbf{p} \prec 0.$$

不幸的是，这些不等式在决策变量 \mathbf{P} 和 \mathbf{K} 中是 **双线性的**，因此是非凸性的。事实证明，通过简单的坐标变化，我们可以把这些变成 **线性** 矩阵不等式 (LMI)， $\mathbf{Q} = \mathbf{P}^{-1}$ ， $\mathbf{Y} = \mathbf{KP}^{-1}$ 。导致

$$\mathbf{q} = \mathbf{q}^T \succ 0, \quad \mathbf{aq} + \mathbf{qa}^T + \mathbf{by} + \mathbf{y}^T \mathbf{b} \prec 0.$$

此外，给定矩阵 \mathbf{A} 和 \mathbf{B} ，存在一个矩阵 \mathbf{K} ，使得 $(\mathbf{A} + \mathbf{BK})$ 是稳定的，当且仅当存在满足这个（严格）线性矩阵不等式的矩阵 \mathbf{Q} 和 \mathbf{Y} 。

9.6.2 通过交替进行的控制设计

对于使用凸优化的控制设计，我们将在很大程度上依赖于动力学是控制的假设，即 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ 。这次让我把它写成：

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{i=0}^{m-1} \mathbf{u}_i \mathbf{f}_{ii}(\mathbf{x}).$$

正如我们已经讨论过的，对于机械系统来说，这个假设是完全合理的。

对于线性最优控制，我们找到了形式为 $\mathbf{u} = -\mathbf{K}\mathbf{x}$ 的控制器。将其自然推广到多项式分析中，就是寻找形式为 $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ 的控制器，其中 $\boldsymbol{\pi}(\mathbf{x})$ 是一项多项式。（对于像上面的钟摆这样的机械系统，我们可以使 $\boldsymbol{\pi}$ 成为 s 和 c 的多项式。）

如果我们将这种控制应用于李亚普诺夫条件（用于全局分析），我们很快就会看到问题所在。我们有

$$V(\mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}} \left[\mathbf{f}(\mathbf{x}) + \sum_{i=0}^{m-1} \mathbf{f}_i(\mathbf{x}) \pi_i(\mathbf{x}) \right].$$

因此，如果我们试图寻找 V 和 $\boldsymbol{\pi}$ 的参数，同时，决策变量成倍增加，问题将是非凸的。

一个非常自然的策略是使用交替法。这个想法很简单，我们将 $\partial_{\mathbf{x}} \boldsymbol{\pi}$ 和优化 V ，然后 $\partial_{\mathbf{x}} V$ 和优化 $\boldsymbol{\pi}$ 并重复，直到收敛。这种方法起源于著名的鲁棒控制的“DK迭代”（例如

[[18](#)]).它利用了每一步的结构化凸优化的优势。

例如，我们可以考虑用LQR对一个非线性系统进行局部稳定，然后寻找一个线性控制法（甚至是高阶多项式控制法）来实现更大的吸引盆地。但要注意的是，一旦我们从全局稳定转向吸引区优化，我们现在需要在三组变量之间交替进行。 $V(\mathbf{x})$ 、 $\pi(\mathbf{x})$ 、 $\lambda(\mathbf{x})$ ，其中 $\lambda(\mathbf{x})$ 是S-程序的乘数多项式。我们在[19]中正是采取了这种方法。在那篇论文中，我们表明交替可以增加对Acrobot的吸引力的认证区域。

交替方法在内循环中利用了凸优化的优势，但它仍然只是解决非凸联合优化的局部方法。它受制于局部最小值。主要的优点是，如果没有数字问题，我们期待递归的可行性（一旦我们有了控制器和李亚普诺夫函数，实现了稳定性，即使有一个小的吸引区域，我们也不会失去它）和目标上的单调改进。也可以尝试用其他非凸的优化程序（如随机梯度下降，或顺序二次编程）来更直接地优化这些目标（例如[6]），但严格的可行性更难保证。通常情况下，李亚普诺夫条件只是在样本处或沿着样本轨迹进行评估；我们仍然可以使用单一的SOS验证步骤来证明性能，控制器在部署前就已经确定。

9.6.3 控制-利亚普诺夫函数

连接李亚普诺夫分析和控制设计的另一个核心思想是“控制-李亚普诺夫函数”。给定一个系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ ，控制-李亚普诺夫函数 V 是一个正DeŒnite函数，对于该函数

$$\forall \mathbf{x} \neq 0, \exists \mathbf{u} \quad \dot{V}(\mathbf{x}, \mathbf{u}) = \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u}) \mathbf{f}(\mathbf{x}, \mathbf{u}) < 0 \text{ 和 } \exists \mathbf{u} \quad \dot{V}(0, \mathbf{u}) = 0.$$

换句话说，对于所有的 \mathbf{x} ，存在着一个允许 V 下降的控制。再一次，我们用额外的条件来装饰这个基本属性（例如，径向无界，或满足控制不变集），以构建全球或区域稳定性声明。需要理解的是，我们可以设计控制-李亚普诺夫函数，而不需要明确地将控制器参数化；通常情况下，控制动作甚至在执行时才会通过找到一个特定的 \mathbf{u} 下坡来确定。

我们的求和工具包非常适合于解决不确定数上的 \forall 量子的问题。处理 \exists 量子的问题要困难得多；我们没有一个类似S程序的解决方案。有趣的是，有一种我们在上面讨论过的方法可以有效地将 \exists 转化为 \forall --这就是吸引力区域分析的外部近似方法。

在外部逼近中，我们产生一个障碍证明，以确定控制器不能去的状态集（对于任何 \mathbf{u} ）。我们的障碍证明现在的形式是

$$\forall \mathbf{x}, \forall \mathbf{u}, \quad \dot{B}(\mathbf{x}, \mathbf{u}) \leq 0.$$

当然， $B(\mathbf{x}) \geq 0$ 现在是 \mathcal{O} xed点的真正“后向可达集”（BRS）的外近似值[920]。同样，[11]有一些很好的例子，可以直接用平方和的形式写出来。

现在，这里变得有点令人沮丧了。当然， $B(\mathbf{x})$ 的子水平集是控制不变的（通过适当的 \mathbf{u} 的选择）。但我们并不（不能）期望整个估计区域（ α -超水平集）能被变成不变的。估计区域是后向可达集的外部近似值。[21]给出了一个从BRS中提取多项式控制律的配方；这个控制律的内部近似可以通过原始的SOS吸引区域工具得到。不幸的是这是次优的/保守的。虽然我们会

要想在内部近似中直接证明控制-利亚普诺夫条件， \exists 量子化仍然是一个障碍。

9.6.4 带有SOS的近似动态编程

我们已经建立了HJB条件和Lyapunov条件之间最重要的联系。

$$\dot{J}^*(\mathbf{x}) = -\ell(\mathbf{x}, \mathbf{u}^*) \quad \text{与} \quad \dot{V}(\mathbf{x}) < 0.$$

HJB涉及到解决一个复杂的PDE；通过将其改为不等式，我们放松了问题，使其适合于凸优化。

走动成本的上限和下限

要求 $\dot{V}(\mathbf{x}) < 0$ 对于证明稳定性是足够的。但我们也可以用这种思路来提供严格的证明，作为代价的上界或下界。给定一个控制动力系统 $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ ，和一个fixed控制器 $\pi(\mathbf{x})$ ，我们可以找到一个函数 $V(\mathbf{x})$ 。

- $\forall \mathbf{x}, \dot{V}^\pi(\mathbf{x}) \leq -\ell(\mathbf{x}, \pi(\mathbf{x}))$ ，以提供一个上界，或
- $\forall \mathbf{x}, \dot{V}^\pi(\mathbf{x}) \geq -\ell(\mathbf{x}, \pi(\mathbf{x}))$ ，以提供一个下限。

要看到这一点，沿任何解的轨迹 $\mathbf{x}(t)$ ， $\mathbf{u}(t)$ 取两边的积分。对于上界，我们得到

$$\int_0^\infty \dot{V}^\pi(\mathbf{x}) dt = V^\pi(\mathbf{x}(\infty)) - V^\pi(\mathbf{x}(0)) \leq \int_0^\infty -\ell(\mathbf{x}(t), \pi(\mathbf{x}(t))) dt.$$

假设 $V^\pi(\mathbf{x}(\infty))=0$ ，我们有

$$V^\pi(\mathbf{x}(0)) \geq \int_0^\infty \ell(\mathbf{x}(t), \pi(\mathbf{x}(t))) dt.$$

上限是我们希望在认证程序中使用的--它提供了一个保证，即从 \mathbf{x} 开始的系统实现的总成本小于 $V(\mathbf{x})$ 。但事实证明，下限对于控制设计来说要好得多。这是因为我们可以写成

$$\forall \mathbf{x}, \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial V f}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u})] \geq 0 \quad \equiv \quad \forall \mathbf{x}, \forall \mathbf{u}, \ell(\mathbf{x}, \mathbf{u}) + \frac{\partial V f}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u}) \geq 0.$$

因此，无需预先指定一个控制器，如果我们能找到一个函数 $V(\mathbf{x})$ ，使 $\forall \mathbf{x}, \forall \mathbf{u}, \dot{V}(\mathbf{x}, \mathbf{u}) \geq -\ell(\mathbf{x}, \mathbf{u})$ ，那么我们就有一个最佳的去向成本的下限了。

你应该花一分钟来说服自己，不幸的是，同样的技巧对上界不起作用。同样，我们将需要 \exists 作为 \mathbf{u} 的量子化，而不是 \forall 。

二次方之和公式

结合我们已经看到的一些想法，这导致了一个自然的最优控制的平方和表述。

$$\begin{aligned} & \text{最大} \int_{\mathbf{x}} V(\mathbf{x}) d\mathbf{x}. \\ & \text{受制于 } \ell(\mathbf{x}, \mathbf{u}) + \frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \text{ 是SOS.} \\ & V(0) = 0. \end{aligned}$$

SOS约束执行下限，而目标通过在某个紧凑区域内最大化积分来"推高"下限。然后，我们可以再次尝试通过使用这个下限作为控制-李亚普诺夫函数，或者通过提取（和认证）多项式控制器来提取控制法。

也许你注意到这是[线性编程方法在动态编程中](#)的自然延伸。对于具有连续状态的系统，LP方法只在样本点验证不等式条件；而在这里，我们对所有的 \mathbf{x} 、 \mathbf{u} 进行验证。这是一个重要的概括：并不是因为它可以更好地认证（下限不是一个特别有价值的认证），而是因为它可以扩展到密集采样不可行的尺寸。这在基于网格的值迭代和非常可扩展的LQR之间提供了某种光谱。

然而，这种方法的重大挑战不是维数，而是实现有意义的近似所需的多项式的程度。请记住，即使是最小时间的双积分器问题，其最佳成本-----也是非光滑的。就像我们将在轨迹优化的背景下看到的伪[频谱方法](#)一样，选择正确的多项式基础可以为这种方法的数值稳健性带来巨大的差异。

例子 (9.16) SOS ADP for the cubic polynomial)

9.7 其他计算方法

例子 (9.17) SOS ADP用于摆锤摆动问题)。

9.7.1 "可满足性模数理论" (SMT)

[dReal](#)在DRAKE中可用。

9.7.2 混合整数编程 (MIP) 公式

9.7.3 延续方法

9.8 神经拉普诺夫函数

9.9 收缩度量

9.10 练习题

练习

9.1 (GlobalStability 的ValidLyapunovFunction

对于系统

$$\begin{aligned}\dot{x}_1 &= -\frac{6x_1}{(1+x_1^2)^2} + 2x_2 \\ \dot{x}_2 &= -\frac{2(x_1+x_2)}{(1+x_1^2)^2}\end{aligned}$$

给你一个正的非线性函数 $V(\mathbf{x}) = \frac{1}{2}x_2^2 + x_1^2$ ([这里](#)绘制的) 和

告诉大家，对于这个系统， $\dot{V}(\mathbf{x})$ 在整个空间上是负去核的。 $V(\mathbf{x})$ 是否是一个有效的李亚普诺夫函数，以证明原点的全局渐进稳定性，为

上述方程所描述的系统？请说明你的答案。

练习（9.2不变量集和吸引力区域）

给你一个动态系统 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, \mathbf{f} 是连续的, 它有一个位于原点的平衡点。让 B_r 是一个以原点为中心的半径为 $r > 0$ 的(不确定)球。 $B_r = \{\mathbf{x} : \|\mathbf{x}\| \leq r\}$ 。假设你找到了一个连续不可逆的标量函数 $V(\mathbf{x})$, 这样。 $V(0) = 0, V(\mathbf{x}) > 0$ 对于 B 中的所有 $\mathbf{x} \neq 0$, 并且 $\dot{V}(\mathbf{x}) < 0$ 对于 B 中的所有 $\mathbf{x} \neq 0$ 。请判断以下陈述是真还是假。请简要说明你的答案。

- B_r 是给定系统的一个不变集, 即: 如果初始状态 $\mathbf{x}(0)$ 位于 B 中, 那么 $\mathbf{x}(t)$ 在所有 $t \geq 0$ 时都属于 B_r 。
- B_r 是平衡点 $\mathbf{x} = 0$ 的ROA的一个子集, 即: 如果 $\mathbf{x}(0)$ 位于 B 中, 那么 $\lim_{t \rightarrow \infty} \mathbf{x}(t) = 0$ 。

练习（9.3Lyapunov函数是唯一的吗？）

如果 $V_1(\mathbf{x})$ 和 $V_2(\mathbf{x})$ 是有效的李亚普诺夫函数, 证明原点的全局渐进稳定性, 那么 $V_1(\mathbf{x})$ 是否一定等于 $V_2(\mathbf{x})$?

练习（9.4证明全局渐进稳定）。

考虑由以下内容组成的系统

$$\begin{aligned}\dot{x}_1 &= x_2 - x_1^3 \\ \dot{x}_2 &= -x_1 - x_2^3\end{aligned}$$

证明李亚普诺夫函数 $V(\mathbf{x}) = x_1^2 + x_2^2$ 证明了全局渐进稳定性
这个系统的原点。

练习（9.5欧几里得空间的梯度流）。

我们可以用李亚普诺夫分析法来分析梯度下降等优化算法的行为。考虑一个目标函数 $\ell: \mathbb{R}^n \rightarrow \mathbb{R}$, 我们想最小化 $\ell(\mathbf{x})$ 。梯度流是梯度下降的连续时间类似物, 被定义为 $\dot{\mathbf{x}} = -\nabla \ell$ 。

- 证明目标函数 $\ell(\mathbf{x}) - \ell(\mathbf{x}^*)$ 是梯度流的Lyapunov函数, 其中 \mathbf{x}^* 是唯一的最小化器。
- 我们可以用李亚普诺夫来论证, 一个优化问题将收敛到一个全局最优, 即使它是非凸的。假设的Lyapunov函数 ℓ , 有负的definite ℓ 。证明目标函数 ℓ 在原点有一个唯一的最小化器。
- 考虑以下图中的目标函数。这是否是一个有效的显示全局渐进稳定性的Lyapunov函数?

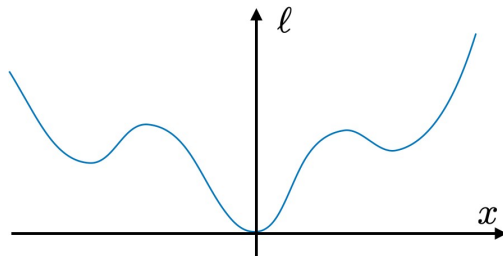


图 -9.8 渐进稳定性的李亚普诺夫函数候选。
d. 考虑目标函数 $\ell(x) = x^4 - 8x^3 + 18x^2 + x^2$, $x \in \mathbb{R}^2$ 。

1112

找到最大的 ρ , 使 $\{x \mid \ell(x) < \rho\}$ 是一个有效的吸引区域为原点。

练习 (9.6 轮式机器人的控制-利亚普诺夫函数)

在这个练习中, 我们研究了用 控制-利亚普诺夫函数 来驱动轮式机器人的想法, 实现了在 [22].

与之前的例子类似, 我们使用机器人的运动学模型。我们用 z_1 和 z_2 表示其笛卡尔位置, 用 z_3 表示其方向。控制是线性 u_1 和角 u_2 的速度。运动方程为

$$\begin{aligned}\dot{z}_1 &= u_1 \cos z_3, \\ \dot{z}_2 &= u_1 \sin z_3, \\ \dot{z}_3 &= u_2.\end{aligned}$$

我们的目标是设计一个反馈法则 $\pi(\mathbf{z})$, 以驱动机器人从任何初始条件到达原点 $\mathbf{z} = \mathbf{0}$ 。正如在 [22] 中指出, 如果我们用极坐标分析, 这个问题就会变得非常容易。如下图所示, 我们让 x_1 是机器人的径向坐标, x_2 是角坐标, 然后我们确定 $x_3 = x_2 - z_3$ 。分析 Figure, 基本运动学考虑导致

$$\begin{aligned}\dot{x}_1 &= u_1 \cos x_3, \\ \dot{x}_2 &= -\frac{u_1 \sin x_3}{x_1}, \\ \dot{x}_3 &= -\frac{u_1 \sin x_3}{x_1} - u_2.\end{aligned}$$

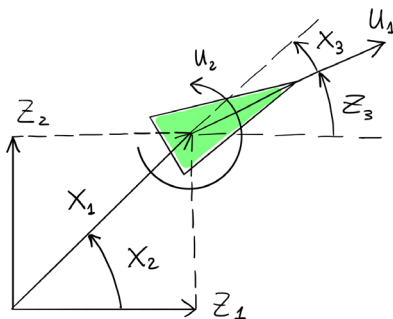


图 -9.9 带有直角坐标系 \mathbf{z} 和极地坐标系 \mathbf{x} 的轮式机器人。

- a. 对于候选的李亚普诺夫函数 $V(\mathbf{x}) = V_1(x_1) + V_2(x_2, x_3)$, 有 $V_1(x_1) = x_1^2$ 和 $V_2(x_2, x_3) = (-x_2^2 + x_3^2)$, 计算时间导数

$$\dot{V}_1(\mathbf{x}, u_1) \text{ 和 } \dot{V}_2(\mathbf{x}, \mathbf{u}).$$

b. 表明选择

$$\begin{aligned} u_1 &= \pi(\mathbf{x}) = -x_1 \cos x_3. \\ u_2 &= \pi(\mathbf{x}) = x_3 - \frac{(x_2 + x_3) \cos x_3 \sin x_3}{x_3}. \end{aligned}$$

在这种情况下，我们让 $\pi_2(\mathbf{x})$ 承担其极限值 $x_2 + 2x_3$ ，以确保反馈规律的连续性）。

- c. 解释为什么Lyapunov的直接方法不能让我们建立闭环系统的渐进稳定性。
- d. 将控制律 $\mathbf{u} = \pi(\mathbf{x})$ 代入运动方程，并推导出闭环动力学 $\dot{\mathbf{x}} = f(\mathbf{x}, \pi(\mathbf{x}))$ 。用LaSalle's定理说明闭环系统的（全局）渐近稳定性。
- e. 在[这个Python笔记本](#)中，我们设置了一个模拟环境，让你尝试我们刚刚得出的控制器。在专用单元格中输入(b)点的控制法，并使用笔记本中的绘图来检查你的工作。

练习9.7 (Lyapunov分析中SOS多项式的局限性)

- a. 是否存在不能用平方之和表示的正定核函数？
- b. 如果我们的动态系统的一个平衡点不承认SOS Lyapunov函数，我们可以对其稳定性得出什么结论？

练习（时间倒置的范德堡振荡器的9.8ROA估计）。

在这个练习中，你将使用SOS优化来逼近时间反转的范德波尔振荡器（[经典范德波尔振荡器](#)的一个变体，在时间上向后演化）的ROA。在[这个python笔记本](#)中，你被要求测试以下SOS公式。

- a. [上面例子](#)中的那个，增加了一个线搜索，使ROA的面积最大化。
- b. 一个单枪匹马的SOS程序，可以直接最大化ROA的面积，而不需要任何线路搜索。
- c. 前者的改进版，在优化问题中施加较少的SOS约束。

参考文献

1. Jean-Jacques E. Slotine and Weiping Li, "Applied Nonlinear Control", Prentice Hall, October, 1990.
2. Hassan K. Khalil, "Nonlinear Systems", Prentice Hall, December, 2001.
3. S. Boyd and L. El Ghaoui and E. Feron and V. Balakrishnan, "Linear Matrix Inequalities in System and Control Theory", SIAM, 1994.
4. Pablo A. Parrilo, "Robustness and Optimization中的Structured Semidefinite Programs and Semialgebraic Geometry Methods", 博士论文, 加州理工学院, 五月18, 2000.

5. Amir Ali Ahmadi and Miroslav Krstic and Pablo A. Parrilo, "A Globally Asymptotically Stable Polynomial Vector Field with no Polynomial Lyapunov Function", *Proceedings of Conference on Decision and Control*, 2011.
6. Shen Shen 和 Russ Tedrake, "Sampling Quotient-Ring Sum-of-Squares Programs for Scalable Verification of Nonlinear Systems", 第59届IEEE决策与控制会议 (CDC) 论文集, [2020.[链接](#)]。
7. Russ Tedrake and Ian R. Manchester and Mark M. Tobenkin and John W. Roberts, "{LQR-Trees}:通过平方之和验证的反馈运动规划", 《国际机器人研究杂志》, 第1038-1052页29,, 7月, [2010.[链接](#)]。
8. Anirudha Majumdar, "Funnel Libraries for Real-Time Robust Feedback Motion Planning", 博士论文, 麻省理工学院, 6月, [2016.[链接](#)]。
9. Didier Henrion和Milan Korda, "多项式控制系统吸引区域的凸计算", *IEEE Transactions on Automatic Control*, 第59,297-312页2。 2014.
10. Jean Bernard Lasserre, "Moments, Positive Polynomials and Their Applications", World Scientific, Vol. 1, 2010.
11. Michael Posa and Twan Koolen and Russ Tedrake, "Balancing and Step Recovery Capturability via Sums-of-Squares Optimization", *Robotics: 科学与工程*, [2017.[链接](#)]。
12. Gerald B Folland, "How to integrate a polynomial over a sphere", *The American Mathematical Monthly*, vol. no108,. pp5,.446-448, 2001.
13. D.Henrion and J.B. Lasserre and C. Savorgnan, " Approximate volume and integration for basic semialgebraic sets", *SIAM Review*, vol. 51, no.4, 第722-743页。 2009.
14. Mark M. Tobenkin and Ian R. Manchester and Russ Tedrake, "Invariant Funnels around Trajectories using Sum-of-Squares Programming", *Proceedings of 18th IFAC World Congress, extended version available online: arXiv:1010.3013 [math.DS]*, [2011.[链接](#)]
15. Anirudha Majumdar, "Robust Online Motion Planning with Reachable Sets", , 5月, [2013.[链接](#)]
16. Charles W. Wampler 和 Andrew J. Sommese, "Numerical algebraic geometry and algebraic kinematics", *Acta Numerica*, vol. 20, pp.469-567, 2011.
17. A.A.J.Sommese和C.W.Wampler, "工程和科学中出现的多项式系统的数值解法", 世界科学出版社。 2005.
18. R.Lind and G.J. Balas and A. Packard, "Evaluating {D-K} iteration for control design", *American Control Conference, 1994* , vol. pp3,-2792 vol2797.3, June29-1 July, 1994.
19. Anirudha Majumdar and Amir Ali Ahmadi and Russ Tedrake, "Control Design along Trajectories with Sums of Squares Programming", *Proceedings of 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp.4054-4061, [2013.[链接](#)]
20. Milan Korda和Didier Henrion以及Colin N Jones, "非线性动态系统的控制器设计和吸引力区域估计", *国际自动控制联合会第19届世界大会*。

(IFAC)。 2014.

21. Anirudha Majumdar and Ram Vasudevan and Mark M. Tobenkin and Russ Tedrake, "Convex Optimization of Nonlinear Feedback Controllers via Occupation Measures", *Proceedings of Robotics:Science and Systems (RSS)*, [2013.[链接](#)]
22. Michele Aicardi and Giuseppe Casalino and Antonio Bicchi and Aldo Balestrino, "Closed loop steering of unicycle like vehicles via Lyapunov techniques", *IEEE Robotics\& Automation Magazine*, Vol. 2, no. 1, pp, 1995.

[上一章](#)

[目录](#)

[下一章](#) [可访](#)

[问性](#)

© Russ Tedrake, 2021

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 10

 在Colab中打开

轨迹优化

我认为最优控制是一个强大的框架，可以用简单的目标函数来指定复杂的行为，让系统的动力学和约束条件来塑造结果的反馈控制器（反之亦然！）。但是到目前为止，我们所提供的计算工具在一些重要方面受到了限制。动态编程的数值方法涉及到在状态空间上放置一个网格，对于状态维度超过4或5的系统来说，这种方法的规模并不大。围绕名义操作点（或轨迹）的线性化使我们能够解决甚至非常高维系统的局部最优控制策略（例如使用LQR），但所产生的控制器的有效性仅限于线性化是非线性动力学的良好近似的状态空间区域。上一章的李亚普诺夫分析的计算工具可以提供，除其他外，一种有效的方法来计算这些区域的估计。但是，我们还没有提供任何真正的近似最优控制的计算工具，这些工具对高维系统的作用超出了围绕目标的线性化。这正是本章的目标。

为了扩展到高维系统，我们将制定一个更简单的优化问题的版本。在本章中，我们不是试图求解整个状态空间的最优反馈控制器，而是试图找到一个仅从单一初始条件出发的最优控制方案。我们可以用一个**轨迹** $\mathbf{x}(t)$, $\mathbf{u}(t)$ 来表示这个解决方案，而不是用一个反馈控制函数来表示，这个轨迹通常是在一个有限区间内确定的。

10.1 问题的制定

给定一个初始条件 \mathbf{x}_0 和在一个有限区间 $t \in [t_0, t_f]$ 上确定的 r 输入轨迹 $\mathbf{u}(t)$ ，我们可以使用标准的加性成本最优控制目标计算执行该轨迹的长期（有限-horizon）成本。

$$J(\mathbf{u}(\cdot) | \mathbf{x}_0) = \ell_f(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t)) dt.$$

我们将把轨迹优化问题写为

$$\begin{aligned} \underset{\mathbf{u}(\cdot)}{\text{min}} \quad & \ell_f(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t)) dt \\ \text{受制于} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \forall t \in [t_0, t_f] \\ & \mathbf{x}(t_0) = \mathbf{x}_0. \end{aligned}$$

一些轨迹优化问题还可能包括额外的约束条件，如避免碰撞（例如，约束条件是机器人的几何形状和障碍物之间的有符号距离保持正值）或输入限制（例如， $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$ ），可以对所有时间或轨迹的一些子集进行定义。

正如所写，上面的优化是对连续轨迹的优化。为了将其表述为一个数值优化，我们必须用一组有限数字对其进行参数化。也许并不奇怪，有许多不同的方法来写下这个参数化，在速度、稳健性和结果的准确性方面有各种不同的特性。我们将在下面概述几个最流行的方法；我推荐[...]。12] 来了解更多的细节。

值得对比的是，这个参数化问题与我们在连续动态编程算法中面临的问题。对于轨迹优化，我们只需要一个维度（时间）的有限维参数化，而在基于网格的值迭代算法中，我们必须在状态空间的维度上工作。我们基于网格的离散化随着状态维度的变化而变化，并导致了难以处理的数值错误。随着时间的推移，人们对二阶梯方程的离散化有了更多的了解，包括对误差控制的积分工作。而且，轨迹参数化所需的参数数量与状态维度呈线性关系，而不是基于网格的数值迭代中的指数关系。

10.2 线性系统的凸公式

让我们首先考虑线性系统的情况。事实上，如果我们从离散时间开始，我们甚至可以推迟解决如何最好地离散连续时间问题的问题。我们可以用几种不同的方式将这个优化问题 "转录" 为一个具体的数学程序。

10.2.1 直接转录

例如，让我们先把 $\mathbf{u}[\cdot]$ 和 $\mathbf{x}[\cdot]$ 都写成决策变量。然后我们就可以写出。

$$\begin{aligned} \underset{\mathbf{x}[\cdot], \mathbf{u}[\cdot]}{\text{min}} \quad & \ell_f(\mathbf{x}[N]) + \sum_{n=0}^{N-1} \ell(\mathbf{x}[n], \mathbf{u}[n]) \\ \text{受制于} \quad & \mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n], \quad \forall n \in [0, N-1] \\ & \mathbf{x}[0] = \mathbf{x}_0 \\ & + \text{额外的限制。} \end{aligned}$$

我们把这种建模选择--增加 $\mathbf{x}[\cdot]$ 作为决策变量并把离散动力学建模为显式约束--称为 "**直接转录**"。重要的是，对于线性系统，动力学约束是这些决策变量的线性约束。因此，如果我们能将我们的额外约束限制为线性不等式约束，并且我们的目标函数在 \mathbf{x} 和 \mathbf{u} 中是线性/二次的，那么所产生的轨迹优化是一个凸优化（具体来说是一个线性程序或二次程序，取决于目标）。因此，我们可以可靠地解决这些问题，在相当大的范围内达到全局最优；这些天，在高速度反馈控制器内在线解决这些优化问题是很常见的。

例10.1（双积分器的轨迹优化）。

我们已经研究了一些使用价值迭代的双积分器的最优控制问题。对于其中的一个问题--对 \mathbf{u} 没有约束的二次元目标

-- 我们现在知道，我们本可以用LQR "精确"地解决这个问题。但我们还没有为最小时间问题和受限的LQR问题给出满意的数字解决方案。

在轨迹表述中，对于离散时间双积分器，我们可以准确地解决这些问题，而对于连续时间双积分器，则有更好的精度。花点时间来体会一下吧！砰砰政策和成本-去向函数是相当不简单的状态函数；我们可以用凸优化来评估它们，这一点很令人满意当然，局限性在于我们一次只能解决一个初始条件的问题。



如果你以前没有研究过凸优化，你可能会对这个框架的建模能力感到惊讶。例如，考虑到以下形式的目标

$$\ell(\mathbf{x}, \mathbf{u}) = |\mathbf{x}| + |\mathbf{u}|。$$

这可以被表述为一个线性程序。要做到这一点，需要增加额外的决策变量 $\mathbf{s}_x[\cdot]$ 和 $\mathbf{s}_u[\cdot]$ --这些变量通常被称为 **松弛变量**--然后写成

$$\min_{\mathbf{x}, \mathbf{u}, \mathbf{s}_x, \mathbf{s}_u} \sum_n^{N-1} \mathbf{s}_x[n] + \mathbf{s}_u[n], \quad \text{s.t. } \mathbf{s}_x[n] \geq x[n], \quad \mathbf{s}_x[n] \geq -x[n], \quad \dots$$

凸优化领域充斥着这样的小技巧。了解和识别它们是（优化）行业的技能。但也有许多相关的约束条件无法用任何技巧重塑为凸约束（在原始坐标中）。一个重要的例子是避开障碍物。想象一下，一辆车必须决定它应该向左还是向右绕过一个障碍物。这代表了 \mathbf{x} 中的一个基本的非凸约束；我们将在下面讨论使用非凸优化的轨迹优化的意义。

10.2.2 直接射击

精明的读者可能已经注意到，将 $\mathbf{x}[\cdot]$ 作为决策变量加入并非严格必要。如果我们知道 $\mathbf{x}[0]$ 并且知道 $\mathbf{u}[\cdot]$ ，那么我们就应该能够用正向模拟来解决 $\mathbf{x}[n]$ 。对于我们的离散时间线性系统，这一点特别好。

$$\begin{aligned} \mathbf{x}[1] &= \mathbf{A}\mathbf{x}[0] + \mathbf{B}\mathbf{u}[0]。 \\ \mathbf{x}[2] &= \mathbf{A}(\mathbf{A}\mathbf{x}[0] + \mathbf{B}\mathbf{u}[0]) + \mathbf{B}\mathbf{u}[1]。 \\ &\vdots \\ \mathbf{x}[n] &= \mathbf{A}^n\mathbf{x}[0] + \sum_{k=0}^{n-1} \mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}[k]。 \end{aligned}$$

更重要的是，该解决方案在 $\mathbf{u}[\cdot]$ 中仍然是线性的。这真是不可思议.....我们可以摆脱一堆决策变量，把一个有约束的优化问题变成一个无约束的优化问题(假设我们没有任何其他约束)。这种方法--使用 $\mathbf{u}[\cdot]$ 而**不是** $\mathbf{x}[\cdot]$ 作为决策变量，并使用正向模拟来获得 $\mathbf{x}[n]$ --被称为**直接拍摄**转录。对于线性

在 \mathbf{x} 和 \mathbf{u} 中具有线性/二次目标的系统，它仍然是一个凸优化，并且比直接转录的决策变量和约束条件更少。

10.2.3 计算方面的考虑

那么，直接拍摄是否一律比直接抄写的方法好？我认为不是这样的。有几个潜在的原因，人们可能更喜欢直接转录。

- 数值调节。拍摄涉及到对潜在的大 n 进行计算⁴ⁿ，这可能导致约束条件中的大范围的coefficient值。这个问题（有时被称为“摇尾乞怜”）是轨迹优化中的一个基本问题：控制输入 $\mathbf{u}[0]$ 确实比控制输入 $\mathbf{u}[N-1]$ 更有机会对总成本产生巨大影响。但直接转录法通过将这一效应分散到大量均衡的约束条件中来解决这一数字问题。
- 增加了状态约束。将 $\mathbf{x}[n]$ 作为明确的决策变量使得它非常容易/自然地增加额外的状态约束；而且求解器有效地重复了每个约束的 $A^{n\theta}$ 的计算。在拍摄中，人们必须为每个新的约束条件展开这些条款。
- 平行化。对于较大的问题，评估约束条件可能是一个巨大的成本。在直接转录中，人们可以并行地评估动态/约束条件（因为每次迭代都是从已经给定的 $\mathbf{x}[n]$ 开始的），而射击从根本上说是一种串行操作。

对于线性凸问题，求解器已经足够成熟，这些差异通常不会太大。对于非线性优化问题，这些差异可能是实质性的。如果你看一下主流机器人学中的轨迹优化论文，你会发现直接转录和直接射击方法都被使用。（你有可能仅仅通过他们使用的转录方法就能猜出是哪个研究室写的论文！）

还值得注意的是，由直接转录产生的问题有一个重要的、可利用的“带状”稀疏模式--大多数约束条件只涉及少量的变量。这实际上与我们在Riccati方程中利用的模式相同。由于这些方法在实际应用中的重要性，许多专门的求解器已经被编写出来，以明确地利用这种稀疏性（例如[3]）。

10.2.4 连续时间

如果我们希望解决该问题的连续时间版本，那么我们可以将时间离散化并使用上述公式。最重要的决定是离散化/数字积分方案。对于线性系统，如果我们假设控制输入在每个时间步长中保持不变（又称零阶保持），那么我们可以完美地整合动力学。

$$\mathbf{x}[n+1] = \mathbf{x}[n] \int_m^{m+h} + [\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}]dt = \mathbf{A}^h\mathbf{e}\mathbf{x}[n] + \mathbf{A}^{-1}(\mathbf{e}^{\mathbf{A}h} - \mathbf{I})\mathbf{B}\mathbf{u}[n]。$$

是简单的情况（当 \mathbf{A} 是可逆的）。但在一般情况下，我们可以使用 $\mathbf{u}(t)$ 的任何 ∞ 参数化表示和任何数字积分方案来获得 $\mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n])$ 。

10.3 非凸面轨迹优化

我强烈建议你研究凸的轨迹优化案例；它可以引导你在精神上的清晰度和目的感。但在实践中轨迹优化经常被用来解决非凸问题。我们的公式会因为一些原因而变得非凸。例如，如果动力学是非线性的，那么动态约束就会变成非凸的。你也可能希望有一个非凸的目标或非凸的附加约束（如避免碰撞）。通常情况下，我们

使用非线性规划的工具来制定这些问题。

10.3.1 直接抄写和直接拍摄

当动力学是非线性的时候，我们上面写的直接转录和直接射击的公式仍然有效，只是产生的问题是非凸的。例如，离散时间系统的直接转录变得越发普遍。

$$\begin{aligned} \min_{\mathbf{x}[\cdot], \mathbf{u}[\cdot]} \quad & \ell(\mathbf{x}[N]) + \sum_{n_0}^{N-1} \ell(\mathbf{x}[n], \mathbf{u}[n]) \\ \text{受制于} \quad & \mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n]), \quad \forall n \in [0, N-1] \\ & \mathbf{x}[0] = \mathbf{x}_0 \\ & + \text{额外的限制。} \end{aligned}$$

直接射击也仍然有效，因为在算法的每次迭代中，我们可以通过正向模拟来计算给定的 $\mathbf{x}[0]$ 和 $\mathbf{u}[\cdot]$ 。但是，当我们考虑连续时间系统时，事情就变得有点有趣了。

对于非线性动力学，我们有很多选择，可以选择如何近似离散的动力学

$$\mathbf{x}[n+1] = \mathbf{x}[n] + \int_{t[n]}^{t[n+1]} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) dt, \quad \mathbf{x}(t[n]) = \mathbf{x}[n].$$

例如，在DRAKE中，我们有一整套数值积分器，可以实现不同水平的模拟速度和/或精度，这两者都高度依赖于 $\mathbf{f}(\mathbf{x}, \mathbf{u})$ 的细节。

在微分方程的数值积分中，一个非常重要的想法是使用变步积分作为控制积分误差的手段。Runge-Kutta-Fehlberg，也被称为 "RK45"，是最著名的变步积分器之一。我们通常会避免在约束条件下使用变步法（会导致梯度不连续），但也有可能在以下情况下完成类似的工作

通过允许采样时间， $t[\cdot]$ ，本身是决定性的，来实现轨迹优化。

变量。这允许优化器拉伸或缩小时间间隔以解决问题，如果你不预先知道轨迹的总持续时间应该是什么，这就特别有用。为这些时间变量添加一些约束条件是必要的，以避免琐碎的解决方案（如塌陷为零持续时间的轨迹）。我们甚至可以添加约束条件来约束积分误差。

10.3.2 直通拼接

对于我们的直接转录来说，有一套数值积分程序可用是非常令人满意的。但是数值积分器被设计成在时间上向前求解，这代表了我们在直接转录公式中实际上没有的设计约束。如果我们的目标是以少量的函数评价/决策变量/约束条件获得二阶梯方程的精确解，那么一些新的公式就有可能利用约束优化公式。这些方法包括所谓的拼合方法。

在直接配位中（c.f., [4]），输入轨迹和状态轨迹都明确表示为片断多项式函数。特别是，这种算法的甜蜜点是将 $\mathbf{u}(t)$ 作为一阶多项式，将 $\mathbf{x}(t)$ 作为一个三次多项式。

事实证明，在这个甜蜜的地方，我们在优化中唯一需要的决策变量是花键的所谓 "断点" 处的样本值 $\mathbf{u}(t)$ 和 $\mathbf{x}(t)$ 。你可能会认为，你需要立方花键的系数

参数，但你不知道。对于 $\mathbf{u}(t)$ 的一阶插值，给定 $\mathbf{u}(t_k)$ 和 $\mathbf{u}(t_{k+1})$ ，我们可以在区间 $t \in [t_k, t_{k+1}]$ 上求解每个值 $\mathbf{u}(t)$ 。但我们也有立体花键所需的一切：给定 $\mathbf{x}(t_k)$ 和 $\mathbf{u}(t_k)$ ，我们可以计算 $\dot{\mathbf{x}}(t_k) = f(\mathbf{x}(t_k), \mathbf{u}(t_k))$ ；四个值 $\mathbf{x}(t_k)$, $\mathbf{x}(t_{k+1})$, $\dot{\mathbf{x}}(t_k)$, $\dot{\mathbf{x}}(t_{k+1})$ 完全定义了立体花键在区间内的所有参数 $t \in [t_k, t_{k+1}]$ 。这非常方便，因为我们可以很容易地添加额外的对样本点的 \mathbf{u} 和 \mathbf{x} 的约束（如果要每个约束转化为对花键系数的约束，则相对更难）。

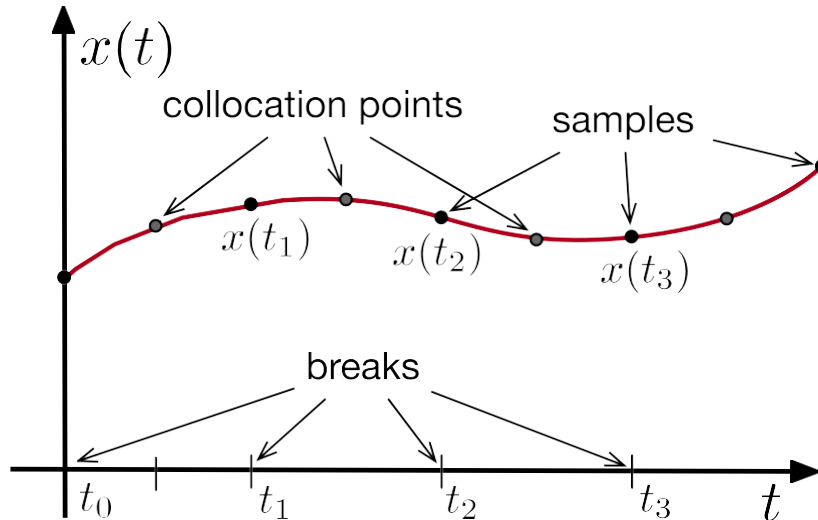


图-10.1-直接配位法中使用的立方花键参数。

事实证明，我们在每个时间段还需要一个约束，以强制执行动力学，并完全指定轨迹。在直接配位中，我们在所谓的配位点添加一个导数约束。特别是，如果我们选择拼合点为样条线中点，那么我们有

$$t_{c,k} = \frac{1}{2}(t_k + t_{k+1}), \quad h_k = t_{k+1} - t_k.$$

$$\mathbf{u}(t) = \left(\frac{1}{2} \mathbf{u}(t_k) + \mathbf{u}(t_{k+1}) \right) + \frac{h_k}{8} (\dot{\mathbf{x}}(t_k) - \dot{\mathbf{x}}(t_{k+1})),$$

$$\mathbf{x}(t) = \left(\frac{1}{2} \mathbf{x}(t_k) + \mathbf{x}(t_{k+1}) \right) + \frac{h_k}{8} (\dot{\mathbf{x}}(t_k) - \dot{\mathbf{x}}(t_{k+1})),$$

$$\dot{\mathbf{x}}(t) = -\frac{3}{2h_k} (\mathbf{x}(t_k) - \mathbf{x}(t_{k+1})) + \frac{1}{4h_k} (\dot{\mathbf{x}}(t_k) + \dot{\mathbf{x}}(t_{k+1})).$$

这些方程直接来自于将三维样条曲线 $\mathbf{x}(t)$ 到端点/导数然后在中点插值的方程。它们恰恰给我们提供了我们所需要的东西，以便将动力学约束添加到我们的合作点上进行评估。

$$\min_{\mathbf{x}[\cdot], \mathbf{u}[\cdot]} \ell(\mathbf{x}[N]) + \sum_{n=0}^{N-1} h_n \ell(\mathbf{x}[n], \mathbf{u}[n])$$

$$\text{受制于} \quad \dot{\mathbf{x}}(t_{c,n}) = f(\mathbf{x}(t_{c,n}), \mathbf{u}(t_{c,n})), \quad \forall n \in [0, N-1].$$

$$\mathbf{x}[0] = \mathbf{x}_0$$

+ 额外的限制。

我希望这个符号是清楚的--我使用 $\mathbf{x}[k] = \mathbf{x}(t_k)$ 作为决策变量，而 t 处的搭配约束 c,k 取决于决策变量： $\mathbf{x}[k]$ 、 $\mathbf{x}[k+1]$ 、 $\mathbf{u}[k]$ 、 $\mathbf{u}[k+1]$ 。实际的运动方程在断裂点 t_k 和拼合点 $t_{c,k}$ 都得到了评估。

再一次，直接配位通过满足优化的约束条件有效地整合了运动方程--这次产生的动力学整合精确到三阶，实际上是对运动方程的两次评估。

每段植物动态（因为我们用 $\dot{\mathbf{x}}(t_k)$ 表示两个区间）。[4]声称，在没有证明的情况下，随着断裂点的接近，轨迹将收敛到二元方程的真解。再一次，在成本函数中增加额外的条款或额外的输入/状态约束是非常自然的，计算目标和约束的梯度也非常容易。我个人认为，在求解技术中明确考虑轨迹的参数化编码是非常好的。

例子（摆锤、杂技演员和车杆的10.2直接对接）

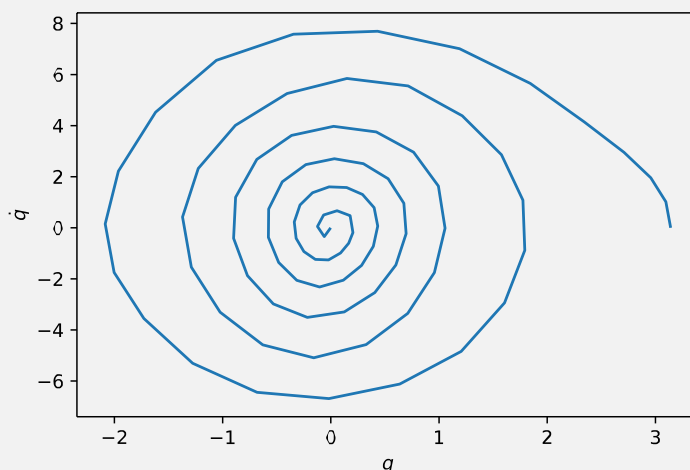


图-10.2-使用直接拼合优化的简单摆（有严格的扭矩限制）的摆动上升轨迹。

直接搭配也很容易解决钟摆、Acrobot和车杆系统的摆动问题。自己试试吧。

[在Colab中打开](#)



像往常一样，请确保你看一下代码！

10.3.3 伪谱系方法

[]的直接同位法是我们的第一个例子，它将最优控制问题的解决方案明确表述为参数化轨迹，并在一系列同位点上对导数添加约束。4]是我们第一个明确地将最优控制问题的解决方案表示为参数化的轨迹，并在一系列的配位点上为导数添加约束的例子。在上述算法中，选择的表示方法是片状多项式，例如立方棘线，而花键系数是决策变量。一个密切相关的方法，通常被称为“伪谱”最优控制，使用相同的拼合思想，但使用全局多项式基函数的线性组合表示轨迹。这些方法通常使用高得多的多项式，但可以利用巧妙的参数化来编写稀疏的拼合目标并选择拼合点[.5, 6].有趣的是，这些方法的表示方法的连续二等分性质导致了比我们看到的其他直接轨迹优化方法相对更多的定理和分析[6]-- 但尽管这些文章中使用了一些语言，请记住它们仍然是试图解决一个非凸优化问题的局部优化方法。虽然上面的直接拼合方法可能会通过向分片多项式添加更多的段（并让每段代表一个较小的时间间隔）来期望收敛到真正的最优解，但在这里，我们期望收敛发生在我们增加多项式的程度时。

伪谱方法有时也被称为 "正交配位", 因为 N 个基础多项式, $j_i(t)$, 被选择为在第 N 个配位点 t_i , 我们有

$$j_i(t_j) = \begin{cases} 1 & i=j \\ 0 & \text{否则的话。} \end{cases}$$

这可以通过选择

$$\phi_i(t) = \prod_{j=0, j \neq i}^N \frac{t - t_j}{t_i - t_j}.$$

请注意, 由于数值上的原因和分析上的原因, 传统上将时间从区间 $[t_0, t_f]$ 调整为 $[-1, 1]$ 。定位点的选择是基于高斯正交的小变化, 被称为 "高斯-洛巴托", 其中包括 $t=-1$ 和 $t=1$ 的定位点。

有趣的是, 一些论文还通过对时间间隔 $t \in [0, \infty]$ 的非线性重新缩放到半开区间 $\tau \in [-1, 1]$, 通过 $\tau = \frac{t-t_0}{t_f-t_0}$ 来实现超时空伪谱优化。[5, 6](#)。在这种情况下, 我们选择的配位时间包括 $\tau = -1$, 但不包括 $\tau = 1$, 使用所谓的 "Gauss-Radau" 点[\[5\]](#)。

隐性形式的动态约束

如果我们的主要兴趣是优化具有重大多体动力学的系统, 那么还有另一个看似微妙但可能很重要的机会, 可以在这些转录中的少数情况下加以利用。在某些情况下, 我们实际上可以直接以隐含的形式写出动力学约束。我们已经在李亚普诺夫分析的背景下[介绍过这个想法](#)。在许多情况下, 以隐式形式获得运动方程会更漂亮或更有效, $\mathbf{g}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{x}}) = 0$, 和

以避免求解 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ 的显式。这可以变成

当我们考虑那些显式没有唯一解的系统时, 这一点就更加重要了--当我们研究通过接触进行轨迹优化时, 我们会看到这样的例子, 因为摩擦力的库仑模型实际上导致了一个二元包容而不是二元方程。

配位方法, 直接对配位点的动态约束以其连续形式进行操作, 可以直接使用隐式形式。有可能使用隐式积分器为直接转录编写一个时间步进 (离散时间近似) --再次提供隐式形式的约束。隐式形式在射击方法中更难利用。

10.3.4 解决方案技术

以上介绍的不同转录方式代表了将 (潜在的连续时间) 最优控制问题映射到一组有限的决策变量、目标和约束条件中的不同方式。但是, 即使做出了这种选择, 也有许多方法来解决这个优化问题。任何[非线性规划](#)的一般方法都可以在这里应用; 在我们目前所包括的python例子中, 问题被直接交给顺序二次规划 (SQP) 求解器SNOPT, 或者交给内点求解器IPOPT。

由于问题的顺序性, 在这些轨迹优化问题中也有相当多的可利用的问题特定结构。因此, 有一些想法对于最优控制的轨迹优化表述来说是相当特殊的, 定制的求解器往往 (有时甚至是显著) 胜过通用的求解器。

这种轨迹优化结构最容易讨论, 也最容易在无约束的公式中实现, 所以我们将从这里开始。事实上, 近几年来, 我们

在机器人领域，使用（通常是特殊用途的）无约束轨迹优化求解器进行轨迹优化的流行趋势，其中约束问题通过惩罚方法被转化为无约束问题。我想说的是，基于增强拉格朗日的惩罚方法在最近的轨迹优化中特别流行。[78](#)].

有效地计算梯度

向求解器提供目标和约束条件的梯度并不是严格要求的--如果不提供梯度，大多数求解器会从有限差分获得梯度--但我强烈认为，如果提供精确的梯度，求解器会更快、更稳健。为直接转录方法提供梯度是非常直接的--我们只是为每个约束条件单独提供梯度。但是在直射法中，我们已经从程序中删除了 \mathbf{x} 个决策变量，但是仍然以 \mathbf{x} 为单位来写目标和约束条件，因此独立计算每个目标/约束条件的梯度将变得非常不容易。我们需要利用连锁规则。

为了简明扼要（和略微笼统），让我们把 $\mathbf{x}[n+1]=f_d(\mathbf{x}[n], \mathbf{u}[n])$ 作为连续动力学的离散时间近似；例如，上面使用的前向欧拉积分方案会得到 $f_d(\mathbf{x}[n], \mathbf{u}[n])=\mathbf{x}[n]+f(\mathbf{x}[n], \mathbf{u}[n])dt$ 。那么我们有

$$\frac{\partial J}{\partial \mathbf{u}_k} = \frac{\partial \ell(\mathbf{x}[N])}{\partial \mathbf{u}_k} + \sum_{n=0}^{N-1} \left(\frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}[n]} \frac{\partial \mathbf{x}[n]}{\partial \mathbf{u}_k} + \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}_k} \right),$$

其中状态相对于输入的梯度可以在“正向模拟”中计算出来。

$$\frac{\partial \mathbf{x}[n+1]}{\partial \mathbf{u}_k} = \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}[n]} \frac{\partial \mathbf{x}[n]}{\partial \mathbf{u}_k} + \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}_k}.$$

这些模拟梯度也可以用在连锁规则中，以提供任何约束的梯度。请注意，这里有很多术语需要保留，其数量为 (state dim) \times (control dim) \times (number of timesteps)。哎哟。还请注意，这些项中有许多是零；例如，在欧拉积分方案中 $\frac{\partial \mathbf{x}[n]}{\partial \mathbf{u}_k} = 0$ 如果 $0 \leq k < n$ 。（如果这看起来像我在这里混合了两个符号，请回顾一下我用 \mathbf{u}_k 代表决策变量， $\mathbf{u}[n]$ 代表模拟的第 n 步中使用的输入）。

无状态约束的直接射击的特殊情况

通过自己求解 $\mathbf{x}(\cdot)$ ，我们已经从优化中删除了大量的约束条件。如果没有额外的状态约束，而且我们需要计算的唯一梯度是目标的梯度，那么就会出现一个令人惊讶的高效算法。我将在这里给出步骤，但不做推导，而是在下面的Pontryagin部分进行推导。

1. 模拟前进。

$$\mathbf{x}[n+1] = f_d(\mathbf{x}[n], \mathbf{u}_n).$$

从 $\mathbf{x}[0] = \mathbf{x}_0$ 。

2. 向后计算。

$$\lambda[n-1] = \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])^T}{\partial \mathbf{x}[n]} + \frac{\partial f(\mathbf{x}[n], \mathbf{u}[n])^T}{\partial \mathbf{x}[n]} \lambda[n],$$

$$\text{从 } \lambda[N-1] = \frac{\partial \ell^*(\mathbf{x}[N])}{\partial \mathbf{x}[N]}$$

3. 提取梯度。

$$\frac{\partial J}{\partial \mathbf{u}[n]} = \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}[n]} + \lambda[n]^T \frac{\partial f(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}[n]}$$

$$\frac{\partial J}{\partial \mathbf{u}[n]} = \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}[n]} + \lambda[n]^T \frac{\partial f(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}[n]}$$

这里 $\lambda[n]$ 是一个与 $\mathbf{x}[n]$ 大小相同的向量，它的解释是 $\lambda[n] = -\frac{\partial J}{\partial \mathbf{x}[n]}$ 。管理 λ 的方程被称为**邻接方程**，它代表了对上述计算大量模拟梯度的一个巨大的效率改进。如果你有兴趣的话，是的，邻接方程正是神经网络文献中著名的**反向传播算法**，或者更普遍的反向模式**自动二分法**的定制版本。

获得良好的解决方案...在实践中。

当你开始在自己的问题上玩弄这些算法时，你可能会觉得自己正处于情绪过山车上。

你会有难以置信的幸福时刻

-- 解算器可能会对高度非琐碎的问题找到非常令人印象深刻的解决方案。但是你也会有沮丧的时候，解算器会返回一个糟糕的解决方案，或者干脆拒绝返回一个解决方案（说“不可行”）。令人沮丧的是，你无法区分一个真正不可行的问题，与解算器仅仅停留在局部最小值的情况。

因此，你的旅程的下一个阶段是开始尝试“帮助”解题者前进。有两种常见的方法。

首先是调整你的成本函数--有些人花了很多时间在目标上添加新的元素，或者调整目标中不同组成部分的相对权重。这是一个滑坡，我倾向于尽量避免它（可能是个错误；其他团体倾向于推出更有说服力的视频！）。

第二种方法是给你的求解器提供一个更好的初始猜测，使其处于“正确的”局部最小值附近。我认为这种方法更令人满意，因为对于大多数问题，我认为确实有一个“正确的”目标和约束条件的表述，而我们的目标应该是找到最优解。再一次，我们在这里看到了直接射击算法和直接转录/拼接算法之间的区别。对于拍摄，我们只能提供解算器

$\mathbf{u}(\cdot)$ 的初始猜测，而其他方法允许我们直接指定 $\mathbf{x}(\cdot)$ 的初始猜测。我发现，即使是非常简单的初始化，这也有很大帮助。在Acrobot和cart-pole的摆动问题的直接定位例子中，只需提供 $\mathbf{x}(\cdot)$ 的初始猜测为一条直线即可。

起点和目标之间的轨迹足以帮助解算器找到一个好的解决方案；事实上，这是必要的。

10.4 本地轨迹反馈设计

一旦我们从轨迹优化中得到一个局部最优轨迹，我们就找到了一个开环轨迹，该轨迹（至少是局部）使我们的最优控制成本最小。在数值公差范围内，这对 $\mathbf{u}_0(t)$, $\mathbf{x}_0(t)$ 代表了系统的可行解轨迹。但是，我们还没有做任何事情来确保这个轨迹是局部稳定的。

事实上，为了达到这一点，我们已经做了一些明显的近似：我们的轨迹优化的积分精度往往比正向模拟时使用的精度要低得多（我们在优化过程中倾向于采取更大的时间步长，以避免增加太多的决策变量），而且优化工具箱的默认收敛容限往往只满足动态约束，大约 10^{-6} 为 1 。因此，如果你要模拟优化后的**甚至可以直接从精确的初始条件**中获得控制轨迹。

从轨迹优化中，你可能会发现，状态轨迹与你的计划轨迹有偏差。

对此，我们可以做一些事情。可以在轨迹优化过程中评估轨迹的局部稳定性，并增加一个奖励开环稳定性的成本或约束条件（例如[9, 10]）。这可能是非常有效的（尽管它往往是昂贵的）。但是开环稳定性是一个相当局限的概念。一个可能更普遍有用的方法是设计一个反馈控制器来调节系统回到计划的轨迹上。

10.4.1 有限跨度LQR

我们已经[在LQR章节中](#)开发了[轨迹稳定的方法](#)。这是我最喜欢的轨迹反馈方法之一，因为它为控制器 $\mathbf{K}(t)$ 提供了一个（数值上的）封闭式解决方案，甚至还附带了一个时间变化的二次成本函数 $S(t)$ ，可以用来进行李亚普诺夫分析。

基本程序是沿着轨迹建立一个时变的线性化

$$\text{在错误的坐标中} \quad \bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}(0t) \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}(0t). \quad \text{和}$$

$\dot{\bar{\mathbf{x}}}(t) = \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t)$ 。这个线性化使用了我们在轨迹优化算法中一直使用的动力学的梯度。一旦我们有了时变线性化，我们就可以应用**Onite-horizon LQR**（详见[LQR章节](#)）。

这种方法的一个主要优点是，我们可以立即着手验证LQR政策下闭环系统的性能。具体来说，我们可以应用**Onite-time**可达性分析来获得证明所需不变性概念的“漏斗”--保证在计划轨迹附近开始的轨迹将保持在计划轨迹附近。请参阅[Onite-time可及性分析](#)部分以了解这些细节。我们将把所有这些想法放在下面的栖息案例研究中。

10.4.2 模型预测控制

使用凸优化解决轨迹优化的成熟性、稳健性和速度导致了一个美丽的想法：如果我们能够足够快速地优化轨迹，那么我们可以将我们的轨迹优化作为反馈策略。这个秘诀很简单。（1）测量当前状态，（2）从当前状态优化一个轨迹，（3）执行优化轨迹中的第一个动作，（4）让动态演化一步，然后重复。这个秘诀被称为**模型预测控制**（MPC）。

尽管该控制器具有非常强的计算性（该策略没有闭合形式的表示；它只是隐含地表示为优化的解决方案），但在MPC上有大量的理论和算法结果[.1112]。有几个核心思想是从从业者应该真正了解的。

一个核心思想是**退行期MPC**的概念。由于我们的轨迹优化问题是在一个**Onite-horizon**上制定的，我们可以把每个优化看作是对未来 N 个时间步的推理。如果我们的真正目标是在比 N 更长的时间范围内优化性能（例如，在有限时间范围内），那么标准的做法是在控制器的每次评估中继续求解 N 步时间范围。在这个意义上，所考虑的总视界在时间上继续向前移动（例如，后退）。

在后退地平线公式中必须注意一些问题，因为在每一个新的步骤中，我们都会在问题中引入新的成本和约束条件（那些与前一次求解的时间 $N+1$ 有关的成本和约束条件）--在解决凸优化问题的过程中不断前进，却突然遇到求解器返回“不可行”的情况，这将是糟糕的。我们可以设计出保证递归可行性的MPC公式--例如，保证如果在时间 n 找到一个可行的解决方案，那么求解器也将在时间 $n+1$ 找到一个可行的解决方案。

在稳定一个固定点 (\mathbf{x}^* , \mathbf{u}^*) 的优化中, 保证递归可行性的最简单的机制也许是给后退的水平线增加一个 final-value 约束, $\mathbf{x}[N]=\mathbf{x}^*$ 。这个想法很简单, 但很重要。考虑到绝对时间的轨迹/约束, 那么在算法的第 k 步, 我们在优化 $\mathbf{x}[k], \dots, \mathbf{x}[k+N]$, 和 $\mathbf{u}[k], \dots, \mathbf{u}[k+N-1]$; 让我们说, 我们已经为这个问题找到了一个可行的解决方案。后退地平线控制的危险在于, 当我们转移到下一步 ($k+1$) 时, 我们第一次在 $\mathbf{x}[k+N+1]$ 处对系统引入约束。但是, 如果我们在第 k 步的可行解有 $\mathbf{x}[k+N]=\mathbf{x}^*$, 那么我们知道, 设置 $\mathbf{x}[k+N+1]=\mathbf{x}^*$, $\mathbf{u}[k+N]=\mathbf{u}^*$ 就能保证为第 $k+1$ 步的新优化问题提供一个可行的解。有了可行性的保证, 求解者就可以自由地寻找一个成本较低的解决方案 (这就是可能现在就有了, 因为我们已经把 final-value 约束进一步移到了未来)。也有可能制定 MPC 问题, 即使在存在建模错误和干扰的情况下也能保证递归可行性 (参见[13])。

今天, 线性 MPC 的理论和实践方面已被充分理解, 它被认为是 LQR 的事实概括, 用于控制受 (线性) 约束的线性系统。

10.5 案例研究。可以像鸟儿一样降落在栖架上的滑翔机

2008 从那时起, 我的 2014 小组进行了一系列越来越复杂的调查[141516171819], 其中提出了一个问题: 无尾翼无人机能否像鸟一样降落在栖息地上?

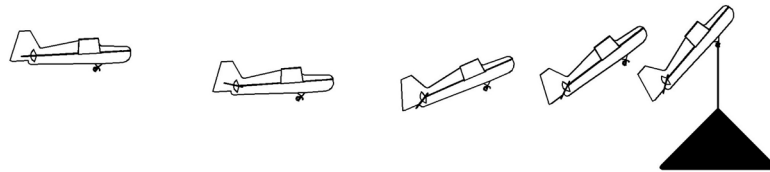


图--滑翔机降落在栖架上的10.3基本设置。

在一开始, 这是一项艰巨的任务。当鸟类在栖息地着陆时, 它们会向上俯冲, 并将它们的翅膀暴露在一个远远超过典型飞行包络线的 "攻击角" 下。飞机在传统上努力避免这种情况, 因为它会导致空气动力学上的 "失速" -- 在气流与机翼分离后突然失去升力。但这种升力的损失也伴随着阻力的大幅增加, 当鸟类迅速减速降落在栖息地时, 它们就利用了这一点。滞空后的空气动力学对控制来说是一个挑战, 因为 (1) 空气动力学是时间变化的 (以周期性的涡流脱落为特征) 和非线性的, (2) 至少在风洞中很难建立这种飞行制度的精确模型, 而且

(3) 失速意味着机翼上附着的气流的损失, 因此也意味着控制面的损失, 所以控制权可能会大量减少。

我们选择这个项目的初衷是认为这将是一个很好的无模型控制的例子 (如强化学习--因为模型是未知的)。然而, 最终, 这个项目让我真正了解了基于模型的轨迹优化和线性优化控制的力量。通过在运动捕捉环境中进行动态系统识别实验, 我们能够将令人惊讶的简单模型 (基于 ∞ 板理论) 与动态系统相连接[14], 也有更精确的模型, 使用 "类似神经网络" 的条款来捕捉模型和数据之间的残差[19]。这使得基于模型的控制变得可行, 但动力学仍然很复杂--虽然轨迹优化应该可行, 但我对只用线性反馈来调节这些轨迹的潜力感到相当怀疑。

我错了。一次又一次, 我看到时变的线性二次调节器采取了高度非微妙的纠正行动--例如, 在轨迹的早期向下倾斜以获得动能, 或向上倾斜以倾倒能量。

系统--以便在最后时刻降落在栖息地上。虽然动力学的线性近似的质量确实降低了，但控制器的有效性下降得不那么快（即使矢量场改变了，控制器需要推动的方向也没有改变）。这也是最初让我对理解非线性系统上的线性控制的吸引区域感兴趣的想法。

最后，这些实验非常成功。我们开始寻找我们可以建造的 "最简单 "的飞机，以捕捉基本的控制动态，减少复杂性，并仍然完成任务。我们最终建造了一系列 ∞ 平板泡沫滑翔机（没有螺旋桨），只有一个执行器来控制升降机。我们在机翼上增加了斜角，以帮助飞机保持在纵向平面内。这些飞机的简单性，加上它们可以通过相当简单的模型来理解的事实，使它们成为我最喜欢的典型的欠驱动系统之一。

图-10.4-来自[]的原始栖息实验。[14](#)] 中的原始栖息实验，用一根简单的绳子作为栖息地。主要视频是用高速摄像机拍摄的；整个栖息轨迹大约需要几秒钟。[8](#)。

10.5.1 平板滑翔机模型

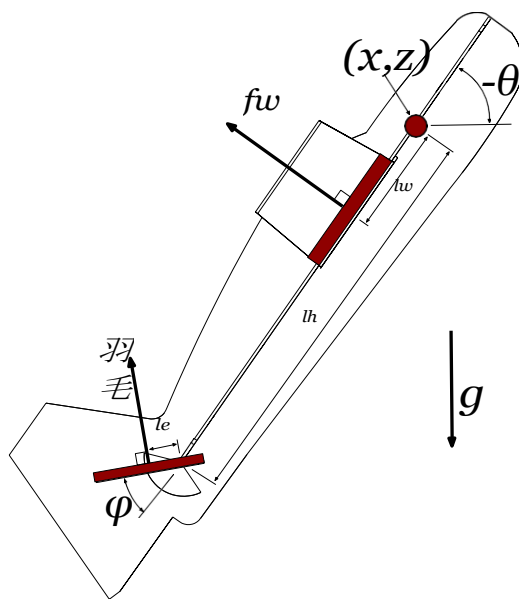


图10.5 - ∞ 平板滑翔机模型。请注意，传统上飞机坐标的选择是，正俯仰使机头向上；但这意味着正Z是向下的（为了有一个右旋的系统）。为了保持一致性，我选择坚持使用我们在整个文本中使用的飞行器坐标系--正Z是向上的，但正俯仰是向下的。

在我们的实验中，我们发现我们的飞机的动力学被所谓的" ∞ 平板模型"很好地捕捉到[14]。在 ∞ at plate理论中，机翼的升力和阻力由机翼压力中心的一个单一的力来概括，其大小为。

$$f(nS, \mathbf{n}, \mathbf{v}) = \rho S \sin^2 \alpha |\mathbf{v}| = -\rho S (\mathbf{n} \cdot \mathbf{v}) |\mathbf{v}|。$$

其中 ρ 是空气的密度， S 是机翼的面积， α 是表面的攻角， \mathbf{n} 是升力面的法向量， \mathbf{v} 是压力中心相对于空气的速度。这相当于有升力和阻力系数

$$C_{\text{lift}} = \sin 2\alpha \cos \alpha, \quad C_{\text{drag}} = \sin^2 \alpha。$$

在我们的滑翔机模型中，我们通过两个升力面的贡献来总结所有的空气动力，机翼（包括来自水平机身的一些贡献）用下标 w 表示，升降机用下标 e 表示，中心在 $\mathbf{p}_w = [x_w, z_w]^T$ 和 $\mathbf{p}_e = [x_e, z_e]^T$ ，由运动学给出。

$$\mathbf{p}_w = \mathbf{p} - l_w \begin{bmatrix} C_{\vartheta} \\ -S_{\vartheta} \end{bmatrix}, \quad \mathbf{p}_e = \mathbf{p} - l_h \begin{bmatrix} C_{\vartheta} \\ -S_{\vartheta} \end{bmatrix} - l_e \begin{bmatrix} C_{\vartheta+j} \\ -S_{\vartheta+j} \end{bmatrix}。$$

其中，我们车辆坐标系的原点， $\mathbf{p} = [x, z]^T$ ，被选择为质量中心。我们假设空气静止，所以机翼和升降机的 $\mathbf{v} = \dot{\mathbf{p}}$ 。

我们假设电梯是无质量的，致动器直接控制速度（注意，这破坏了我们的"控制 $\mathbf{a} \in \mathbb{R}^n$ "结构，但对于我们所处理的微小的业余舵机来说是更现实的）。这就给了我们一个状态变量 $\mathbf{x} = [x, z, \vartheta, \phi, \dot{x}, \dot{z}, \dot{\vartheta}]^T$ 和一个控制输入 $\mathbf{u} = \dot{\phi}$ 。由此产生的运动方程是

$$\begin{aligned} \mathbf{n}_w &= \begin{bmatrix} S_{\vartheta} \\ C_{\vartheta} \end{bmatrix}, \quad \mathbf{n}_e = \begin{bmatrix} S_{\vartheta+j} \\ C_{\vartheta+j} \end{bmatrix}。 \\ f_w &= f_n(S_w, \mathbf{n}_w, \dot{\mathbf{p}})_w, \quad f_e = f_n(S_e, \mathbf{n}_e, \dot{\mathbf{p}})_e。 \\ \ddot{x} &= \frac{1}{m_1} (f_{S_w} + f_{S_e\vartheta+\phi})。 \\ \ddot{z} &= \frac{1}{m_1} (f_{C_w} + f_{C_e\vartheta+\phi}) - g。 \\ \ddot{\vartheta} &= \frac{1}{I} (l_w f_w + (l_c + l_e) f_e)。 \end{aligned}$$

10.5.2 轨迹优化

 在Colab中打开

10.5.3 轨迹稳定化

10.5.4 弹道漏斗

10.5.5 超越单一的轨迹

围绕额定轨迹的线性控制器令人惊讶地有效，但这还不够。当我们讨论 "反馈运动规划" 时，我们将再次回到这个例子，以讨论如何找到一个能在更多初始条件下工作的控制器--最好是飞机能够到达目标的所有相关初始条件。

10.6 蓬莱阁'的最低原则

我们一直在开发的用于数值轨迹优化的工具与（分析）最优控制的定理密切相关。让我们用一个章节来体会这些联系。

一个轨迹， $\mathbf{x}(\cdot)$ ， $\mathbf{u}(\cdot)$ ，是局部最优的确切含义是什么？这意味着，如果我以任何方式扰动该轨迹(例如通过 ϵ 改变 \mathbf{u})，那么我的目标函数将产生更高的成本，或者违反约束。对于无约束的优化，局部优化的一个**必要条件**是，目标在解的梯度正好为零。当然，梯度也可以在局部最大值或鞍点消失，但它肯定必须在局部最小值消失。我们可以用**拉格朗日乘法器**把这个论点推广到约束性优化。

10.6.1

拉格朗日乘法器

的头等方程

让我们使用拉格朗日乘法器来推导出我们在离散时间内受限轨迹优化问题的必要条件

$$\begin{aligned} \min_{\mathbf{x}[\cdot], \mathbf{u}[\cdot]} \quad & \ell(\mathbf{x}[N]) + \sum_{n=0}^{N-1} \ell(\mathbf{x}[n], \mathbf{u}[n])。 \\ \text{受制于} \quad & \mathbf{x}[n+1] = \mathbf{f}_d(\mathbf{x}[n], \mathbf{u}[n])。 \end{aligned}$$

形成拉格朗日。

$$L(\mathbf{x}[\cdot], \mathbf{u}[\cdot], \lambda[\cdot]) = \ell(\mathbf{x}[N]) + \sum_{n=0}^{N-1} \ell(\mathbf{x}[n], \mathbf{u}[n]) + \sum_{n=0}^{N-1} \lambda^T[n] (f_d(\mathbf{x}[n], \mathbf{u}[n]) - \mathbf{x}[n+1])。$$

并将导数设为零，得到上面为射击算法描述的邻接方程方法。

$$\begin{aligned} \forall n \in [0, N-1], \quad \frac{\partial L}{\partial \lambda[n]} &= f_d(\mathbf{x}[n], \mathbf{u}[n]) - \mathbf{x}[n+1] = 0 \Rightarrow \mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]) \\ n \forall \in [0, N-1], \quad \frac{\partial L}{\partial \mathbf{x}[n]} &= \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}} + \lambda^T[n] \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}} - \lambda^T[n-1] = 0 \\ &\Rightarrow \lambda[n-1] = \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])^T}{\partial \mathbf{x}} + \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])^T}{\partial \mathbf{x}} \lambda[n]. \\ \frac{\partial L}{\partial \mathbf{x}[N]} &= \frac{\partial \ell^T}{\partial \mathbf{x}} - \lambda^T[N-1] = 0 \Rightarrow \lambda[N-1] = \frac{\partial \ell^T}{\partial \mathbf{x}} \\ n \forall \in [0, N-1], \quad \frac{\partial L}{\partial \mathbf{u}[n]} &= \frac{\partial \ell(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}} + \lambda^T[n] \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}} = 0。 \end{aligned}$$

因此，如果我们给定一个初始条件 \mathbf{x}_0 和一个输入轨迹 $\mathbf{u}[\cdot]$ ，我们可以通过模拟系统向前求解 $\mathbf{x}[\cdot]$ ，向后求解邻接方程来验证它是否满足最优性的必要条件。 $\lambda[\cdot]$ 的时间，并验证 $\frac{\partial L}{\partial \lambda}$ 对 λ 所有 n 。

$\frac{\partial L}{\partial \mathbf{x}} = 0$ 和 $\frac{\partial L}{\partial \lambda} = 0$ 从变迁微积分的一些基本结果中得出。

$$\frac{\partial L}{\partial \mathbf{u}} = 0$$

10.6.2 连续时间内最优性的必要条件

听到这些必要条件在连续时间里有一个类似物，你不会感到惊讶。我在此说明，不做推导。考虑到初始条件 \mathbf{x}_0 ，连续动态 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ ，以及瞬时成本 $\ell(\mathbf{x}, \mathbf{u})$ ，对于在 $t \in [t_0, t_f]$ 上定义的轨迹 $\mathbf{x}(\cdot)$ ， $\mathbf{u}(\cdot)$ 来说，它必须满足以下条件才是最优的

$$\begin{aligned} \forall t \in [t_0, t_f], \quad \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u})。 & \mathbf{x}(0) &= \mathbf{x}_0 \\ \forall t \in [t_0, t_f], \quad -\lambda &= \frac{\partial \ell^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} \lambda, & \lambda(T) &= \frac{\partial \ell^T}{\partial \mathbf{x}} \\ \forall t \in [t_0, t_f], \quad \frac{\partial \ell}{\partial \mathbf{u}} &+ \lambda^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}} &&= 0。 \end{aligned}$$

事实上，该声明甚至可以被推广到 \mathbf{u} 有约束的情况下。这个结果被称为庞特里亚金的最小原则--为一条轨迹成为最优提供必要条件。

定理--10.1 庞特里亚金的最小原则

改编自[20]。考虑到初始条件 \mathbf{x}_0 、连续动态 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ 和瞬时成本 $\ell(\mathbf{x}, \mathbf{u})$ ，对于在 $t \in [t_0, t_f]$ 上定义的轨迹 $\mathbf{x}^*(\cdot)$ ， $\mathbf{u}^*(\cdot)$ 来说，它必须满足以下条件才是最优的

$$\begin{aligned} \forall t \in [t_0, t_f], \quad \dot{\mathbf{x}}^* &= \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*)。 & \mathbf{x}^*(0) &= \mathbf{x}_0 \\ \forall t \in [t_0, t_f], \quad -\lambda^* &= \frac{\partial \ell^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} \lambda^*, & \lambda^*(T) &= \frac{\partial \ell^T}{\partial \mathbf{x}} \\ \forall t \in [t_0, t_f], \quad \mathbf{u}^* &= \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmin}} [\ell(\mathbf{x}^*, \mathbf{u}) + (\lambda^*)^T \mathbf{f}(\mathbf{x}^*, \mathbf{u})]。 \end{aligned}$$

请注意，该定理最后一行中被最小化的条款通常被称为最优控制问题的哈密尔顿。

$$H(\mathbf{x}, \mathbf{u}, \lambda, t) = \ell(\mathbf{x}, \mathbf{u}) + \lambda^T \mathbf{f}(\mathbf{x}, \mathbf{u})。$$

它有别于经典力学的哈密顿，但受到经典力学的启发。记住 λ 有一个解释为 $\frac{\partial L}{\partial x}$ ，你也应该从以下方面认识它

监狱管理局。

10.7 变化和扩展

10.7.1 二维平坦度

在一些非常重要的情况下，基于对变量的巧妙改变，非凸的轨迹优化可以转回为凸的轨迹优化。这个领域的一个关键想法是 "differential flatness" 的概念，它在哲学上与我们讨论的杂技机器人和车杆系统的 部分反馈线性化 的想法非常接近。但我们将在此探讨的最著名的 "差动性" 的应用，实际上是针对四旋翼机器人的。

部分反馈线性化最重要的经验之一是，如果你有 m 个执行器，那么你基本上可以精确地控制系统的 m 个数量。Differential flatness 利用了这个相同的想法（选择 m 个输出），但方向相反。这个想法是，对于某些系统，如果你在这 m 个坐标中给我一个轨迹，它实际上可能决定了所有的状态/执行器必须一直在做什么。在这种情况下，你只能执行一个可能的轨迹子集，这可以使轨迹规划变得更加容易！

让我们从一个例子开始...

例子（10.3 平面四旋翼的微分拍子）。

前面描述的 平面四旋翼模型有3个自由度（ x 、 y 、 ϑ ），但只有两个执行器（每个推进器一个）。我的主张是，如果你只给我一个质心位置的轨迹： $x(t)$ ， $y(t)$ ， $\forall t \in [t_0, t_f]$ ，那么我就能推断出 $\vartheta(t)$ 在同一区间内必须是什么，才是可行的。此外，我甚至可以推断出 $u(t)$ 。有一个技术条件要求：你给我的 $x(t)$ 和 $y(t)$ 的轨迹需要是连续二阶的（四次）。

要看到这一点，请回顾一下这个系统的运动方程是由以下内容给出的。

$$\ddot{x} = -(u_1 + u_2) \sin \vartheta. \quad (1)$$

$$\ddot{y} = (u_1 + u_2) \cos \vartheta - mg \quad (2)$$

$$I \ddot{\vartheta} = r(u_1 - u_2) \quad (3)$$

请注意，从(1)和(2)，我们有 $\frac{\ddot{x}}{\ddot{y} + mg} = \frac{(u_1 + u_2) \sin \vartheta}{(u_1 + u_2) \cos \vartheta} = \tan \vartheta$ 。

换句话说，给定 $\ddot{x}(t)$ 和 $\ddot{y}(t)$ ，我可以求出 $\vartheta(t)$ 。我可以将这种关系（在时间上）再二分两次，得到 ϑ 。使用(3)与(1)或(2)给我们提供了两个 u_1 和 u_2 的线性方程，这很容易解决。

现在你可以看到为什么我们需要原始轨迹是平滑的--解决方案对 $u(t)$ 的影响取决于 $\vartheta(t)$ ，它取决于 \ddot{x} 和 \ddot{y} ；我们需要这些导数沿着整个轨迹存在。

更重要的是--如果我们暂时忽略输入限制--任何顺利的smoothly的

$x(t)$ 、 $y(t)$ 的轨迹是可行的，所以如果我可以简单地找到一个避开障碍物的轨迹，那么我就设计了我的状态轨迹。正如我们将看到的，即使是高阶片断多项式的优化实际上也是一个简单的问题（它被证明是一个二次方程），假设约束条件是凸的。在实践中，这意味着一旦你确定了是向左还是向右绕过障碍物，轨迹设计就很容易和快速。

我在这里为你编了一个简单的例子。



在Colab中打开

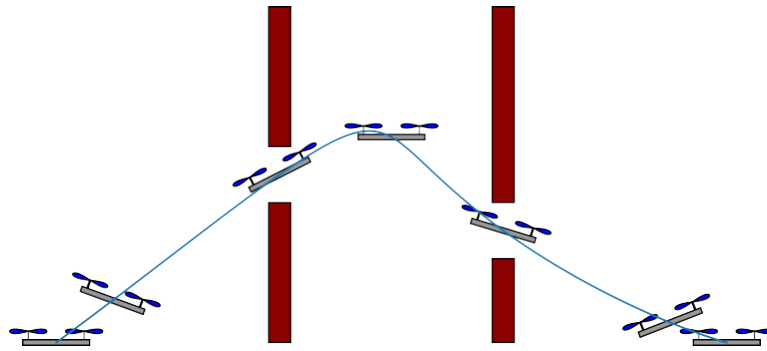


图--平面四旋翼飞机的10.6二重 ∞ --通过解决一个简单的优化，找到 $x(t)$ 和 $y(t)$ 的平滑轨迹，我可以倒出 $\theta(t)$ ，甚至 $u(t)$ 。

上面的例子表明，平面四旋翼系统是不透明的。 ∞ 在输出 $x(t)$ 、 $y(t)$ 中。更一般地说，如果我们有一个动态系统

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}).$$

而我们设计一些输出坐标（基本上是 "任务空间"）。

$$\mathbf{z}(t) = h(\mathbf{x}, \frac{d\mathbf{u}}{dt}, \dots, \frac{d^k \mathbf{u}}{dt^k}).$$

这样，我们可以把 \mathbf{x} 和 \mathbf{u} 纯粹写成输出的函数和它的时间导数。

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(\mathbf{z}, \frac{d\mathbf{z}}{dt}, \dots, \frac{d^k \mathbf{z}}{dt^k}) \\ \mathbf{u}(t) &= \mathbf{u}(\mathbf{z}, \frac{d\mathbf{z}}{dt}, \dots, \frac{d^k \mathbf{z}}{dt^k}). \end{aligned}$$

那么我们就说系统 f 在输出 $\mathbf{z}[\dots]$ 方面是不透明的。²¹要证明一个系统在这些输出中是differentially ∞ at的，需要把解决 $\mathbf{x}(t)$ 和 $\mathbf{u}(t)$ 的函数写成仅是 $\mathbf{z}(t)$ 及其时间导数的函数。

我不知道有什么数字口诀可以表明一个系统在本质上是 ∞ 的，也不知道有什么口诀可以发现潜在的 ∞ 输出，但我承认我没有研究过，也没有找过这些口诀。这将是很有趣的!我认为differential ∞ atness是一个必须为你的系统找到的属性--通常通过一点机械的直觉和大量的代数。一旦找到，它就会非常有用。

例10.4（三维四旋翼的微分拍子）。

最著名的二重性 ∞ 的例子可能是在全三维四旋翼飞机上。[\[22\]](#)表明，三维四旋翼在输出方面是二重 ∞ 性的。

$\{x, y, z, \vartheta_{yaw}\}$ 。他们利用这一点，以戏剧性的效果，表演各种杂技。

由此产生的视频非常棒（在接下来的几年里，四旋翼飞机在学术界和工业界的广泛普及可能应得到很大的赞扬）。

图-10.7-积极的四旋翼飞行器轨迹，使用来自[\[...\]](#)的differential flatness。 [22](#)].

关于这些例子，有几件事需要注意，只是为了让我们也了解其局限性。首先，上面的技术对于设计轨迹来说是很好的，但要稳定这些轨迹还需要额外的工作（我们将讨论这个话题

在后面的注释中会有更详细的介绍)。轨迹稳定极大地受益于良好的状态估计，而上面的例子都是在运动捕捉领域进行的。此外，"简单"版本的轨迹设计足够快，可用于穿过移动的环形物，但仅限于凸优化公式--这意味着我们必须事先硬编码，以决定在每个障碍物周围是向左还是向右/向上还是向下。

10.7.2

迭代

LQR和Differential动态

编程

还有一种轨迹优化的方法（至少对于初值问题），最近变得相当流行。迭代LQR（iLQR）[23]也被称为顺序线性四边形最优控制）[24]。这个想法很简单：给定一个输入和状态轨迹的初始猜测，对动力学进行线性近似，对成本函数进行二次近似。然后计算和模拟时变的LQR控制器，以确定新的输入和状态轨迹。重复进行，直到收敛。

迭代LQR背后的动机相当吸引人--它利用了李嘉图方程的惊人结构，在对李嘉图方程进行一次后向传递后，再进行一次前向模拟，就能得出对轨迹的二阶更新。根据经验，收敛速度很快，而且很稳健。众多的应用...[2526]。


iLQR的一个关键限制是，它只支持无约束的轨迹优化--至少是通过Riccati解。有约束的优化的自然扩展是用迭代线性模型预测控制（MPC）优化来取代Riccati解决方案；这将导致一个二次方程，并且非常接近SQP中的情况。但在目前的机器人文献中，一个更常见的方法是使用惩罚方法将任何约束添加到目标函数中，特别是使用增强拉格朗日。

二阶动态编程（DDP）[27]的灵感来自于解决HJB方程的局部二次近似的想法。由此产生的算法几乎与iLQR相同；唯一的区别是DDP使用植物动力学的二阶近似，而不是一阶（线性）的。传统的观点是，采取这些额外的植物导数可能不值得计算它们的额外成本；在大多数情况下，iLQR收敛得同样快（尽管细节取决于问题）。不幸的是，尽管这些算法之间的区别是明确的和没有争议的，你会发现许多论文在实际使用iLQR时错误地写成了DDP。

10.7.3 非凸性约束的混合整数凸优化

例子10.5(绕过障

碍物的混合整数规划)

在Colab中打开

[28], [29], [3031].

最近的工作使图形搜索的优化方法与我们的连续优化之间有了更强的联系[32].

10.7.4 明确的模型预测控制

10.8 练习题

练习 (10.1直接拍摄与直接抄写的对比)

在这个编码练习中，我们详细探讨了[直接转录比直接拍摄方法的计算优势](#)。这个练习可以完全在这个[python笔记本](#)中完成。为了简化分析，我们将把这两种方法应用于一个 ∞ -horizon LQR问题。要求你完成四段代码。

- 使用给定的直接射击的实现来分析这种方法的数值调节。
- 完成给定的直接转录法的实施。
- 验证一下，随着时间跨度的增长，从直接描述的成本接近LQR的成本。
- 分析直接转录的数字调节。
- 实施动态编程递归（又称 "Riccati递归"），以有效地解决直接转录法的线性代数问题。

移 迹 练习10.2(轨道转 的 优化)

在这个练习中，你将只在[这个笔记本](#)上工作。你被要求通过非线性轨迹优化，找到一条能有效地将火箭从地球运到火星，同时避开小行星云的路径。优化问题的骨架已经有了，但还缺少几个重要的部分。更多的细节在笔记本中，但你需要。

- 执行最大推力约束。
- 强制执行最大速度约束。
- 确保火箭不与任何小行星相撞。
- 增加一个目标函数，使燃料消耗最小化。

练习 (10.3迭代线性二次调节器)

该练习在[本笔记本](#)中是自成一体的。在这个练习中，你将推导并实现迭代线性二次调节器 (iLQR)。你将通过为自主车辆规划轨迹来评估它的功能。你将需要。

- 为连续动力学设计一个积分器。
- 计算一个给定初始状态和控制轨迹的轨迹。
- 总结出该轨迹的总成本。
- 推导出二次Q函数的系数。
- 对最佳控制法进行优化。
- 推导出价值函数向后的更新。
- 实施iLQR的前向传递。
- 实施iLQR的后向传递。

参考文献

1. John T. Betts, "轨迹优化的数值方法调查"。

指导、控制和动力学杂志》，第193-207页2,21,。 1998.

2. John T. Betts, "Practical Methods for Optimal Control Using Nonlinear Programming", Society for Industrial and Applied Mathematics, 2001.
3. Yang Wang and Stephen Boyd, "Fast model predictive control using online optimization", *IEEE Transactions on control systems technology*, vol. 18, no. pp2,, 2009.
4. C.R. Hargraves 和 S. W. Paris, "Direct Trajectory Optimization using Nonlinear Programming and Collocation", *J Guidance*, Vol. 10, no.4, pp. 338-342, July-August, 1987.
5. Divya Garg and Michael Patterson and Camila Francolin and Christopher Darby and GeoÆrey Huntington and William Hager and Anil Rao, "Direct trajectory optimization and costate estimation of Å Ænite-horizon and inÆnite-horizon optimal control problems using a Radau pseudospectral method", *Computational Optimization and Applications*, vol. pp49,, 2011.
6. I.Michael Ross 和 Mark Karpenko, "A review of pseudospectral optimal control: {From} theory to Æight", *Annual Reviews in Control*, vol. 36, no. 2, pp. 182-197, dec, 2012.
7. TC Lin 和 JS Arora, "DiÆerential dynamic programming technique for constrained optimal control", *Computational Mechanics*, vol. 9, no. 1, pp.27-40, 1991.
8. Marc Toussaint, "A Novel Augmented Lagrangian Approach for Inequalities and Convergent Any-Time Non-Central Updates", , 2014.
9. Katja D. Mombaur and Hans Georg Bock and Johannes P. Schloder and Richard W. Longman, "Open-loop stable solutions of periodic optimal control problems in robotics", *Z. Angew.数学. Mech.(ZAMM)*,第85卷,第2页7,。 2005.
10. Aaron M Johnson and Jennifer E King and Siddhartha Srinivasa, "Convergent planning", *IEEE Robotics and Automation Letters*, vol. 1, no. 1044-10512,, 2016.
11. Carlos E Garcia and David M Prett and Manfred Morari, "Model predictive control: theory and practice-a survey", *Automatica*, vol. 25, no.3, pp. 335-348, 1989.
12. Eduardo F Camacho and Carlos Bordons Alba, "Model predictive control",Springer Science/& Business Media, 2013.
13. Alberto Bemporad 和 Manfred Morari , "鲁棒模型预测控制。 A survey", *Robustness in identifiVcation and control*, vol. 245, pp, 1999.
14. Rick Cory和Russ Tedrake, "固定翼{无人机}的实验栖息", *AIAA 指导、导航和控制会议论文集*, 第1-12页, [2008.[链接](#)]。
15. John W. Roberts和Rick Cory以及Russ Tedrake, "On the Controllability of Fixed-Wing Perching", *美国控制会议 (ACC) 论文集*, [2009.[链接](#)]。
16. 里克-科里, "Supermaneuverable Perching", 博士论文, 麻省理工学院, 六月, [2010.[链接](#)]。
17. Joseph Moore, "Powerline Perching with a Fixed-wing {UAV}", , May, 2011.

[[链接](#)]

18. Joseph Moore和Russ Tedrake, "Control Synthesis and Verification for a Perching {UAV} using {LQR}-Trees", *In Proceedings of the IEEE Conference on Decision and Control*, pp. December8,, [2012.[链接](#)]
19. Joseph Moore, "Robust Post-Stall Perching with a Fixed-Wing UAV", 博士论文, 麻省理工学院, 9月, [2014.[链接](#)]。
20. Dimitri P. Bertsekas, "动态编程和最优化控制", Athena Scientific, 2000.
21. Richard M. Murray and Muruhan Rathinam and Willem Sluis, "Differential ∞ 机械控制系统的脆弱性。原型系统的目录"。
ASME国际机械工程大会和博览会。 1995.
22. D.Mellinger 和 V.Kumar , "四旋翼飞机的最小快照轨迹生成和控制", *IEEE2011 国际机器人和自动化会议 (ICRA)* 论文集, 第2520-2525页。 2011.
23. Weiwei Li和Emanuel Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems.", *International Conference on Informatics in Control, Automation and Robotics*, pp.222-229, 2004.
24. Athanasios Sideris和James E. Bobrow, "A Fast Sequential Linear Quadratic Algorithm for Solving Unconstrained Nonlinear Optimal Control Problems", , February, 2005.
25. Farbod Farshidian and Edo Jelavic and Asutosh Satapathy and Markus Giffthaler and Jonas Buchli, "Real-time motion planning of legged robots:A model predictive control approach", *IEEE2017-RAS第17届国际仿人机器人会议 (Humanoids)*, 第577-584页。 2017.
26. Y.Tassa and T. Erez and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization", *Intelligent Robots and Systems (IROS), IEEE/2012RSJ International Conference on*, pp.4906-4913, 2012.
27. David H. Jacobson 和 David Q. Mayne, "Differential Dynamic Programming", American Elsevier Publishing Company, Inc., 1970.
28. A.A. ~Richards和J.P.~How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming", *Proceedings of the American 2002Control Conference*, vol. pp3,,1936-1941, 2002.
29. Daniel Mellinger and Alex Kushleyev and Vijay Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams", *2012 IEEE international conference on robotics and automation*, pp.477-483, 2012.
30. Robin Deits和Russ Tedrake, "Efficient Mixed-Integer Planning for 杂乱环境中的{UAVs}", 《*IEEE国际机器人自动化会议论文集*》, 2015年。
。 *国际机器人和自动化会议 (ICRA)* , 2015。 [[链接](#)]
31. Benoit Landry and Robin Deits and Peter R. Florence and Russ Tedrake, "Aggressive Quadrotor Flight through Cluttered Environments Using Mixed Integer Programming", *Proceedings of International Conference on Robotics and Automation (ICRA)*, May, [2016.[链接](#)]。
32. Tobia Marcucci and Jack Umenberger and Pablo A. Parrilo and Russ Tedrake, "Shortest Paths in Graphs of Convex Sets", *arXiv preprint*, [2021.

链接]。

33. O.Junge and J. E. Marsden and S. Ober-Bloebaum, "Discrete mechanics and optimal control", *Proceedings of the 16th IFAC World Congress*, 2005.

[上一章](#)

[目录](#)

[下一章](#) [可访](#)

[问性](#)

© Russ Tedrake, 2021

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 11

政策搜索

 在Colab中打开

在我们关于栖息飞机的案例研究中，我们解决了一个具有挑战性的控制问题，但我们的控制设计方法只基于线性最优控制（围绕优化轨迹）。我们还讨论了一些非线性最优控制的方法，这些方法可以超越小的离散状态空间的范围。These were based on estimating the cost-to-go function, including value iteration using function approximation and approximate dynamic programming as a linear program or as a sums-of-squares program.

估算成本-收益函数（又称价值函数）的方法有很多值得喜欢的地方。成本-去向函数将长期规划问题减少为一步到位的规划问题；它编码了所有关于未来的相关信息（仅此而已）。HJB给我们提供了成本-去向函数的优化条件，给我们提供了一个强大的算法处理方法来工作。

在本章中，我们将探讨另一个非常自然的想法：让我们用一些决策变量来给控制器设置参数，然后直接搜索这些决策变量，以实现任务和/或优化性能目标。这种方法的一个具体动机是（公认的有点传闻的）观察，通常情况下，简单的政策在复杂的机器人和潜在的复杂的成本-运行函数上可以表现得非常好。

我们将这一大类方法称为“策略搜索”，或者，当使用优化方法时，我们有时会使用“策略优化”。这个想法并没有得到控制界相当多的关注，可能是因为我们知道许多相对简单的情况下，它的效果并不好。但是最近由于强化学习（RL）中“策略梯度”算法的经验性成功，它又变得非常流行。本章包括对这些RL策略梯度算法的“基于模型”版本的讨论；我们将在未来一章中描述其“无模型”版本。

11.1 问题的制定

考虑一个静态的全状态反馈政策。

$$\mathbf{u} = \pi(\alpha \mathbf{x})。$$

其中 π 可能是一个非线性函数，而 α 是描述控制器的参数向量。控制可能以时间为输入，甚至可能有它自己的内部状态，但让我们从这个简单的形式开始。

利用我们对使用加法成本的最佳控制的规定，我们可以从任何初始条件下使用，例如，评估这个控制器的性能。

$$J(\alpha \mathbf{x}) = \int_0^{\infty} \ell(\mathbf{x}(t), \mathbf{u}(t)) dt。$$

受制于 $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ 。 $\mathbf{u} = \pi_{\alpha}(\mathbf{x})$ 。 $\mathbf{x}(0) = \mathbf{x}_0$ 。

为了给每组政策参数 α 提供一个标量成本，我们还需要一个要素：不同初始条件的相对重要性。

我们将在讨论[随机最优控制](#)时进一步阐述，一个非常自然的选择--保留了HJB的递归结构--是优化成本的期望值，给定一些初始条件的分布。

$$\min_{\alpha} E[\mathbf{x}_0 \sim J_{\alpha}(\mathbf{x})]。$$

其中 \mathbf{x}_0 是初始条件的概率分布， $\mathbf{x}(0)$ 。

11.2 线性二次调节器

为了开始思考直接在策略参数中搜索的问题，从一个我们知道并能很好理解的问题开始是非常有帮助的。在线性、时间不变量系统的LQR问题中，我们知道最优策略是一个线性函数： $\mathbf{u} = -\mathbf{K}\mathbf{x}$ 。到目前为止，我们总是间接地获得 \mathbf{K} --通过解决一个Riccati方程来Find the cost-to-go and then backing out the optimising policy.

在这里，让我们研究这样的情况：我们将 \mathbf{K} 的元素作为决策变量进行参数化，并试图直接优化预期成本--去向。

11.2.1 政策评估

首先，让我们评估一下给定 \mathbf{K} 的目标。这一步被称为 "[策略评估](#)"。如果我们使用均值为零、协方差为 $\mathbf{\Omega}$ 的高斯作为初始条件的分布，那么对于LQR，我们有

$$E \left[\int_0^{\infty} [\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}] dt \right]。$$

受制于 $\dot{\mathbf{x}} = \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u}$ 。 $\mathbf{u} = -\mathbf{K} \mathbf{x}$ 。 $\mathbf{x}(0) \sim \mathcal{N}(0, \mathbf{\Omega})$ 。

这个问题也被称为 H_2 最优控制问题[1]，因为这里的预期成本是线性系统2从干扰输入（这里只是产生初始条件的脉冲）到性能输出（这里是指

例如， $\mathbf{z} = [\mathbf{Q} \mathbf{x}^2]$ 。 $\mathbf{R} \mathbf{u}^2$ ）。

为了评估一个政策 \mathbf{K} ，让我们首先利用矩阵跟踪的特性，稍微重新排列成本函数。

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{x}^T \mathbf{K}^T \mathbf{R} \mathbf{K} \mathbf{x} = \mathbf{x}^T (\mathbf{Q} + \mathbf{K}^T \mathbf{R} \mathbf{K}) \mathbf{x} = \text{tr}((\mathbf{Q} + \mathbf{K}^T \mathbf{R} \mathbf{K}) \mathbf{x} \mathbf{x}^T)。$$

以及积分和预期值的线性。

$$E \left[\int_0^{\infty} \text{tr}((\mathbf{Q} + \mathbf{K}^T \mathbf{R} \mathbf{K}) \mathbf{x} \mathbf{x}^T) dt \right] = \text{tr}((\mathbf{Q} + \mathbf{K}^T \mathbf{R} \mathbf{K}) E \left[\int_0^{\infty} \mathbf{x} \mathbf{x}^T dt \right])。$$

对于任何给定的初始条件，闭环动力学的解决方案是由矩阵指数给出的。

$$\mathbf{x}(t) = (\mathbf{A} - \mathbf{BK})^t \mathbf{e}\mathbf{x}(0)。$$

对于初始条件的分布，我们有

$$E [\mathbf{x}(t)\mathbf{x}(t)^T] = e^{(\mathbf{A}-\mathbf{BK})t} E [\mathbf{x}(0)\mathbf{x}(0)^T] e^{(\mathbf{A}-\mathbf{BK})^T t} = e^{(\mathbf{A}-\mathbf{BK})t} \mathbf{\Omega} e^{(\mathbf{A}-\mathbf{BK})^T t}。$$

这只是 t 的一个（对称）矩阵函数。这个函数的积分，称为 \mathbf{X} ，代表闭环响应的预期 "能量"。

$$\mathbf{X} = E \left[\int_0^{\infty} \mathbf{x}\mathbf{x}^T dt \right]。$$

假设 \mathbf{K} 是稳定的， \mathbf{X} 可以被计算为Lyapunov方程的（唯一）解。

$$(\mathbf{a} - \mathbf{bk})\mathbf{x} + \mathbf{x}(\mathbf{a} - \mathbf{bk})^T + \omega = 0。$$

（你可以[在这里](#)看到一个密切相关的推导）。最后，总的政策评估是由

$$E\mathbf{x} \sim N(0, \omega) | [(\mathbf{K}\mathbf{x})] = \text{tr}((\mathbf{Q} + \mathbf{K}^T \mathbf{R} \mathbf{K}) \mathbf{X})。 \quad (1)$$

11.2.2 \mathbf{K} 中的一个非凸目标

不幸的是，李亚普诺夫方程代表了一个非线性约束（关于 \mathbf{K} ， \mathbf{X} 这一对）。事实上，众所周知，当有3个或更多的状态变量时，即使是稳定线性系统的控制器集在参数 \mathbf{K} 中也是非凸的[2]。

例子（11.1 稳定化 \mathbf{K} 的集合可以是非凸的）

下面的例子是在[2]。考虑一个离散时间线性系统， $\mathbf{A} = \mathbf{I}_{3 \times 3}$ ， $\mathbf{B} = \mathbf{I}_{3 \times 3}$ 。

$$\mathbf{K}_1 = \begin{bmatrix} 1 & 0 & -10 \\ -1 & & 1 \\ 0 & & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{和} \quad \mathbf{K}_2 = \begin{bmatrix} 1 & 100 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

都是这个系统的稳定控制器（ $\mathbf{A} - \mathbf{BK}$ 的所有特征值都在复平面的单位圆内）。然而，控制器 $\hat{\mathbf{K}} = (\mathbf{K}_1 + \mathbf{K}_2) / 2$ 有两个特征值在单位圆之外。

由于达到 ∞ 总成本的控制器集合是非凸的，显然我们在这里考虑的成本函数也是非凸的。

作为一个旁观者，对于这个问题，我们实际上知道一个变量的变化，使问题凸。让我们引入一个新的变量 $\mathbf{Y} = \mathbf{K}\mathbf{X}$ 。由于 \mathbf{X} 是PSD，我们可以把 $\mathbf{K} = \mathbf{Y}\mathbf{X}^{-1}$ 退回去⁻¹。现在我们可以重写这个优化。

$$\begin{aligned} \min_{\mathbf{X}, \mathbf{Y}} \quad & \text{tr} \mathbf{Q} \mathbf{X} \mathbf{Q} + \text{tr} \mathbf{R} \mathbf{Y} \mathbf{X} \mathbf{Y}^{-1} \mathbf{R} \\ \text{受制于} \quad & \mathbf{A} \mathbf{X} - \mathbf{B} \mathbf{Y} + \mathbf{X} \mathbf{A}^T - \mathbf{Y}^T \mathbf{B}^T + \mathbf{\Omega} = 0, \\ & \mathbf{X} \succeq 0. \end{aligned}$$

目标中的第二项看起来是不凸的，但实际上是凸的。在

为了把它写成一个SDP，我们可以用多一个松弛变量 \mathbf{Z} 和一个舒尔补码来完全取代它[3]:

$$\begin{aligned} \min_{\mathbf{X}, \mathbf{Y}, \mathbf{Z}} \quad & \text{tr } \mathbf{Q} \mathbf{X} \mathbf{Q}^T + \text{tr } \mathbf{Z} \\ \text{受制于} \quad & \mathbf{A} \mathbf{X} - \mathbf{B} \mathbf{Y} + \mathbf{X} \mathbf{A}^T - \mathbf{Y}^T \mathbf{B}^T + \mathbf{\Omega} = \mathbf{0}, \\ & \begin{bmatrix} \mathbf{Z} & \mathbf{R} \mathbf{Y}^T \\ \mathbf{Y}^T \mathbf{R}^T & \mathbf{X} \end{bmatrix} \succeq \mathbf{0}. \end{aligned}$$

然而，我们最初的问题是问在原始参数化中直接搜索， \mathbf{K} 。如果目标在这些参数中是非凸的，那么我们应该如何进行搜索？

11.2.3 没有局部最小值

虽然凸性是保证优化景观不存在任何局部最小值的必要条件，但它实际上并不是必须的。[2]表明，对于这个LQR目标，所有的局部最优实际上是全局最优。这一分析在[4]中对这一分析进行了扩展，给出了一个更简单的分析，并包括收敛率。

如何证明一个优化景观没有局部最小值（即使它可能是非凸的）？最流行的工具之一是用著名的Polyak-Łojasiewicz (PL) 不等式来证明梯度支配性[5]。对于一个优化问题

$$\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$$

我们首先假设函数 f 是 L 平滑的（Lipschitz梯度）。

$$\forall \mathbf{x}, \mathbf{x}', \quad \|\nabla f(\mathbf{x}') - \nabla f(\mathbf{x})\|_2 \leq L \|\mathbf{x}' - \mathbf{x}\|_2.$$

那么我们说，如果以下情况在某些 $\mu > 0$ 成立，则该函数满足PL-inequality。

$$\forall \mathbf{x}, \quad \frac{1}{2} \|\nabla f(\mathbf{x})\|_2^2 \geq \mu (f(\mathbf{x}) - f^*).$$

其中 f^* 是在最佳状态下获得的值。换句话说，目标的梯度必须比二次函数的梯度增长得快。但是请注意，这里的距离是以 $f(\mathbf{x})$ 而不是 \mathbf{x} 来衡量的；我们并不要求（也不暗示）最优解是唯一的。这显然意味着对于任何最小值， \mathbf{x}' 与 $\nabla f(\mathbf{x}') = \mathbf{0}$ ，因为左边是零，我们必须让右边也是零，所以 \mathbf{x}' 也是一个全局最优值： $f(\mathbf{x}') = f^*$ 。

例子（11.2 一个没有局部最小值的非凸函数）。

考虑一下函数

$$f(x) = x^2 + \sin^2(x).$$

我们可以通过观察来确定这个函数是不凸的，对于 $a = \pi$ ， $b = 3\pi$ ，我们有

$$f\left(\frac{a+b}{2}\right) = \frac{\pi^2}{4} + 3 \approx 5.47 > \frac{f(a) + f(b)}{2} = \frac{5\pi^2}{16} + \frac{3}{2} \approx 4.58.$$

我们可以利用梯度建立PL条件

$$\nabla f(x) = 2x + \sin 6(x) \cos(x).$$

我们可以确定，这个函数是L平滑的，L=由8

$$|\nabla f(b) - \nabla f(a)| = |2b - 2a + \sin 6(b-a)| \leq 8 |b - a|.$$

因为 $\sin(x) \leq x$ 。最后，我们从PL-不等式中得到了梯度支配性。

$$\frac{1}{2}(2x + \sin(6x) \cos(x))^2 \geq \mu(x^2 + \sin(3^2x)).$$

(我用一个小的 [dReal](#) 程序证实了这一点) 0.175。

[5] 给出了在PL条件下，[梯度下降法](#)收敛到最优状态的收敛率。[4]表明，我们在此研究的LQR成本相对于K的梯度，在成本-去向函数的任何子级集合上都满足PL条件。

11.2.4 真正的梯度下降

上述结果表明，人们可以使用梯度下降来获得LQR的最佳控制器 \mathbf{K}^* 。对于我们'目前所看到的变化（我们知道模型），我绝对建议解决Riccati方程是一个更好的算法；它更快，更稳健，没有像步长这样的参数需要调整。但是，当我们把梯度下降看作是一种不太完美的算法的模型时，梯度下降就变得更加有趣/可行了，例如，植物模型没有给定，梯度是由噪声样本估计的。

在这里，由于我们有能力整合线性植物/控制器、二次成本和高斯初始条件，我们可以准确地计算出价值函数，这是一种罕见的奢侈，在(1).我们还可以计算出真正的梯度--这是我们应该在方法中争取的精确性的巅峰，但很少会再达到。梯度的计算方法是

$$\frac{\partial E[\mathbf{J}(\mathbf{x})]}{\partial \mathbf{K}} = 2 (\mathbf{R}\mathbf{K} - \mathbf{B}^T \mathbf{P}) \mathbf{X}。$$

其中 \mathbf{P} 满足另一个Lyapunov方程。

$$(\mathbf{a} - \mathbf{b}\mathbf{k})^T \mathbf{p} + \mathbf{p}(\mathbf{a} - \mathbf{b}\mathbf{k}) + \mathbf{q} + \mathbf{k}^T \mathbf{r} \mathbf{k} = 0.$$

请注意，强化学习中使用的术语策略梯度通常指的是我在上面暗示的稍微不同的一类算法。在这些算法中，我们使用政策的真实梯度（仅），但通过抽样估计梯度中的其余项。这些方法通常需要许多样本来估计我们在这里计算的梯度，而且应该只比本章中的算法更弱（不那么有效）。研究LQR的梯度下降收敛性的论文也开始探索这些情况。我们将很快研究这些所谓的无模型"策略搜索算法。

11.3 更多对比结果和对比实例

LQR / H_2 控制是一个很好的案例，我们知道，对于直接在 \mathbf{K} 中参数化的目标，所有的局部最优都是全局最优。[6] 将这一结果扩展到 H_2/H_∞ 混合控制。[7]给出了最近对表格（ \emptyset nite）MDP情况的处理。

对于LQR，我们也知道控制器的其他参数化，使目标实际上是凸的，包括LMI公式和Youla参数化。它们在策略搜索环境中的效用最初是在[8].

不幸的是，我们并不期望这些好的属性在一般情况下都能成立。有一些附近的问题，已知在原始参数中是不凸的。静态输出反馈的情况是一个重要的例子。如果我们把植物模型扩展到包括（可能是有限的）观察： $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ ， $\mathbf{y} = \mathbf{C}\mathbf{x}$ ，那么直接搜索控制器， $\mathbf{u} = -\mathbf{K}\mathbf{y}$ ，已知是NP-hard[9].这时，稳定 \mathbf{K} 矩阵的集合可能不仅是非凸性的，而且实际上是

断了联系。我们可以通过一个简单的例子（有一次在与Alex Megretski的谈话中给我的）看到这一点。

例子11.3(静态
	输出
反馈	的参数
化)	

考虑单输入、单输出的LTI系统

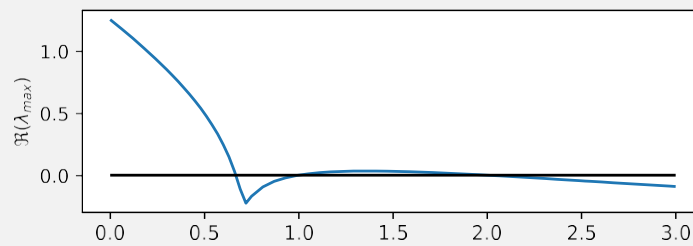
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}。 \quad \mathbf{y} = \mathbf{C}\mathbf{x}。$$

与

$$\mathbf{A} = \begin{bmatrix} 0 & 0.2 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 1 & 3 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

这里，线性静态输出-反馈策略可以写成 $u = -ky$ ，有一个标量参数 k 。

这里是闭环系统的最大特征值（实部）与 k 的函数关系图，只有当这个最大值小于零时，系统才是稳定的。你会发现稳定的 k 的集合是一个不相连的集合。



还有许多其他已知的反例。[\[10\]](#)给出了一个特别简单的双状态MDP。其中一个大的开放性问题⁴是深层网络参数化是否是一个好的例子。

11.4 基于轨迹的政策搜索

虽然我们不能指望梯度下降在一般情况下收敛到全局最小值，但对于更复杂的非线性控制问题，尝试使用梯度下降来确定策略仍然是非常合理的。在一般情况下，这意味着优化的第1步是

$$\min_{\alpha} E [\mathbf{x} \sim \mathbf{x}_0 J_{\alpha}(\mathbf{x})].$$

是估计

$$\frac{\partial}{\partial \alpha} E_{\mathbf{x} \sim \mathbf{x}_0} [J_{\alpha}(\mathbf{x})].$$

在LQR问题中，我们能够准确地计算这些项；最大的简化来自于[线性系统对高斯初始条件的响应保持高斯](#)的事实。但对于更一般的非线性系统来说，这并不正确。那么，我们该怎么做呢？

最常见的/一般的技术（尽管它不是很有效），是使用大量的样本（Monte-Carlo）来近似预期的去向成本。

$$E [\mathbf{x} \sim \mathbf{x}_0 J_{\alpha}(\mathbf{x})] \approx \frac{1}{N} \sum_{i=0}^{N-1} J_{\alpha}(\mathbf{x}_i), \quad \mathbf{x}_i \sim \mathbf{x}_0.$$

梯度很容易遵循。

$$\frac{\partial}{\partial \alpha} E [\mathbf{x} \sim \mathbf{x}_0 J_{\alpha}(\mathbf{x})] \approx - \sum_{i=0}^{N-1} \frac{\partial J_{\alpha}(\mathbf{x}_i)}{\partial \alpha}, \quad \mathbf{x}_i \sim \mathbf{x}_0.$$

我们对这一估计的准确性的信心将随着我们增加 N 而提高；见

例如，[1]的4.2章节。[11](#)]中关于置信区间的细节。我们中最乐观的人会说，只有一个有噪声的梯度估计值是很难做到的。
 --这导致了随机梯度下降，它可以有一些非常理想的特性。但它确实使算法更难分析和调试。

使用Monte-Carlo估计器，总梯度更新只是关于特定初始条件 $\alpha_{\pi(x_i)}$ 的梯度之和。但对于finite-horizon目标来说，这些梯度正是我们在轨迹优化的背景下已经研究过的。它们可以用邻接法有效地计算出来。两者的区别在于，在这里我们把 α 看作是反馈控制器的参数，而之前我们把它们看作是轨迹的参数，但这对连锁规则没有任何区别。

例子（具有真实梯度的11.4LQR与近似梯度的对比）。

在Colab中打开

11.4.1 黑暗中的目标

11.4.2 全球优化的搜索策略

为了对抗局部最小值，...进化的策略。...

11.5 政策迭代

在李亚普诺夫分析一章中，我们还探索了少数[控制设计](#)的技术。[其中一个](#)甚至直接将控制器参数化（作为一个多项式反馈），并使用交替的方式在策略评估和策略优化之间切换。[Another](#) generated lower-bounds to the optimal cost-to-go.

即将推出...

参考文献

1. Ben M. Chen, "H2 {最优} {控制}", *Encyclopedia of {系统} and {控制}*, 第515--520页。 2015.
2. Maryam Fazel and Rong Ge and Sham M. Kakade and Mehran Mesbahi, "Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator", *International Conference on Machine Learning*, 2018.

3. Erik A Johnson和Baris Erkus, "通过LMI合成的智能阻尼器的耗散性和性能分析", *结构控制和健康监测。Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures*, vol. no14,, pp, 3,2007.
4. Hesameddin Mohammadi and Armin Zare and Mahdi Soltanolkotabi and Mihailo R Jovanović, "Global exponential convergence of gradient methods over the nonconvex landscape of the linear quadratic regulator", *IEEE2019 58th Conference on Decision and Control (CDC)* , pp.7474-7479, 2019.
5. Hamed Karimi和Julie Nutini以及Mark Schmidt, "polyak-Łojasiewicz条件下梯度和近似梯度方法的线性收敛", *欧洲机器学习和数据库知识发现联合会议*, 第795-811页。 2016.
6. 张凯庆和胡斌以及Tamer Basar, "Policy Optimization for \mathcal{H}_2 Linear Control with \mathcal{H}_∞ Robustness Guarantee:Implicit Regularization and Global Convergence", *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 179-190, 10-11 June2020,, 2020.
7. Alekh Agarwal and Sham M. Kakade and Jason D. Lee and Gaurav Mahajan, "论政策梯度方法的理论。优化、近似和分布转移", , 2020.
8. John Roberts and Ian Manchester and Russ Tedrake, "Feedback Controller Parameterizations for Reinforcement Learning", *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, [2011.[link](#)] 。
9. Vincent Blondel和John N Tsitsiklis, "NP-hardness of some linear control design problems", *SIAM journal on control and optimization*, vol. no35,, pp6,,2118-2127, 1997.
10. Jalaj Bhandari和Daniel Russo, "全球{透明性}。{保证}{对于}的保证{政策} {梯度}{方法}", *arXiv:1906.01786 [cs, stat]*, oct, 2020.
11. Reuven Y Rubinstein和Dirk P Kroese, "Simulation and Monte Carlo method",John Wiley / Sons,vol. No. 10,2016.

低动能机器人技术

走路、跑步、游泳、飞行和操纵的算法

吕斯-特德雷克

© Russ Tedrake, 2021

最后修改 2021-6-13.

如何引用这些笔记，使用注释，并给予反馈。

注意：这些是用于在麻省理工学院教授的课程的工作笔记。它们将在2021年春季学期中被更新。讲座视频可在YouTube上找到。

[上一章](#)

[目录](#)

[下一章](#)

章节 12

作为搜索的运动规划

术语 "运动规划" 是一个无可奈何的通用术语，它几乎肯定包含了我们已经讨论过的动态编程、反馈设计和轨迹优化算法。然而，还有一些我们尚未讨论的算法和想法，它们是从将运动规划表述为搜索问题的想法中发展起来的--例如，在一个太大的图形中搜索从起点到目标的路径，用动态编程完全解决。有些算法，但肯定不是全部，为了找到任何存在的路径而牺牲了最优性，而规划器 "完整" 的概念--保证找到一条存在的路径--受到高度重视。这正是我们本章的目标，增加一些额外的工具，能够为我们最复杂的几何学、高度非凸的机器人控制问题提供某种形式的解决方案。

[1]是一本非常好的关于一般规划算法，特别是运动规划算法的书。与其他规划问题相比，运动规划通常指的是规划域是连续的问题（如连续状态空间、连续行动空间），但许多运动规划算法的起源可以追溯到离散域的想法（如图搜索）。

在本章中，我们将考虑以下问题的表述：给定一个由非线性动力学（连续或离散时间）定义的系统

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \text{ 或 } \mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n])。$$

并给定一个起始状态 $\mathbf{x}(0)=\mathbf{x}_0$ 和一个目标区域 G ，如果存在这样的轨迹，请找出从 \mathbf{x}_0 到 $\mathbf{x} \in G$ 的任何有限-time轨迹。

12.1 人工智能作为搜索

悠久的历史.....一些人认为，创造智能机器的途径是收集大量的知识本体，然后进行非常高效的搜索。（更现代的人工智能观点是专注于机器学习，正确的答案可能涉及到两者的部分）。萨缪尔跳棋选手、深蓝下棋、定理证明、Cyc、IBM Watson.....。

其中一个关键的想法是使用 "启发式方法" 来指导搜索。"是否有可能在不访问每个节点的情况下，从起点到目标的最优路径？"。A*，可接受的启发式方法。例如：谷歌地图。

在线规划。D*，D*-Lite。

12.2 随机化运动规划

如果你还记得我们最初是如何将动态编程引入图搜索的，你会记得在离散化状态空间方面存在一些挑战。让我们假设，我们已经将连续空间离散为我们图中的一些 $\mathcal{O}nite$ 的离散节点集。即使我们愿意将机器人的行动空间离散化（这在实践中甚至可能是可以接受的），我们也有一个问题，即图中一个节点的离散行动，在某个 $\mathcal{O}nite$ 区间 h 上的整合，极不可能正好落在图中另一个节点之上。为了解决这个问题，我们不得不开始研究在节点之间插值函数估计的方法。

如果你试图求解整个状态空间的成本-去向函数，内插法可以很好地工作，但它与试图在空间中找到一条单一路径的搜索方法不太兼容。如果我从节点1开始，在节点和2节点之间降落，那么我应该从3,哪个节点继续扩展？

避免这个问题的一个方法是在搜索执行过程中建立一个 **搜索树**，而不是依靠预先设计好的网格离散化。这棵树将包含扎根于连续空间的节点，这些节点正好是系统可以到达的点。

连续空间的任何 $\mathcal{O}xed$ 网格离散化，甚至行动空间的 $\mathcal{O}xed$ 离散化的另一个问题是，除非我们对我们的连续系统有具体的几何/动态见解，否则很难提供一个 **完整的** 规划算法。即使我们能证明在树/图上不存在通往目标的路径，我们怎么能确定连续系统没有路径？如果我们把系统离散化，或者更多的离散化，也许会出现一个解决方案？

解决这第二个挑战的一个方法是抛弃 $\mathcal{O}xed$ 离散的概念，而用随机抽样来代替（另一个方法是在算法运行时自适应地增加离散的分辨率）。随机抽样，例如行动空间的随机抽样，可以产生连续空间的 **概率完整的算法**--如果问题的解决方案存在，那么概率完整的算法将以1的概率找到该解决方案，因为样本的数量达到了不确定。

考虑到这些动机，我们可以建立一个也许是最简单的概率上完整的算法，在一个连续的状态和行动空间中找到一条从起始状态到某个目标区域的路径。

例子（用随机树进行12.1规划）

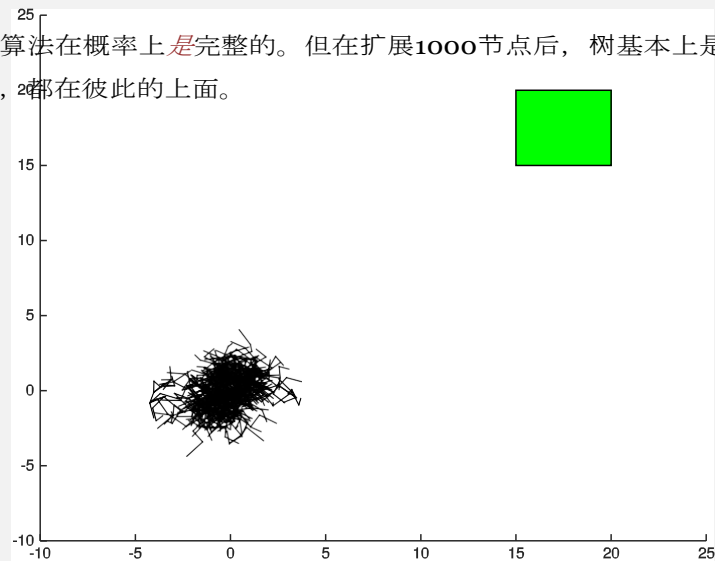
让我们把包含树的数据结构表示为 T 。该算法非常简单。

- 用起始状态初始化树。 $T \leftarrow x_0$ 。
- 在每个迭代中。
 - 从树中随机选择一个节点， x_{rand} ， T
 - 从可行的行动分布中选择一个随机的行动， u_{rand} 。
 - 计算动力学： $x_{new} = f(x_{rand}, u_{rand})$
 - 如果 $x_{new} \in G$ ，则终止。找到解决方案！
 - 否则将新的节点添加到树上， $T \leftarrow x_{new}$ 。

可以证明，这种算法实际上是概率上完整的。然而，如果没有强大的启发式方法来指导选择预定扩展的节点，它可能是极其不完善的。对于一个简单的例子，考虑系统 $\mathbf{x}[n]=\mathbf{u}[n]$, $\mathbf{x}\in\mathbb{R}^2$, $\mathbf{u}\in[-1,1]$ 。我们将从原点开始，把目标区域为 $\forall i, -15x_i\leq 20$ 。你自己试试吧。

```
T = struct('parent',zeros(1,1000),'node',zeros(2,1000)); % 预先分配内存给 T
for i=2:size(T.parent,2)
    T.parent(i) = randi(i-1)
    %
    x_rand = T.node(:,T.father(i));
    u_rand = *2rand(2,1)-1;
    x_new = x_rand+u_rand;
    如果15 (<=x_new(1) && x_new(1)<20= && <=15x_new(2) && x_new(2)<=20)
        disp('Success!'); break;
    end
    T.node(:,i) = x_new;
结束
clf;
line([T.node(1,T.parent(:2end));T.node(:,12:end)], [T.node(:,152T.parent(:2end));T.node(2,2:e
nd patch[15,1520,20202015], 'g' )
轴 ([-10,25,-10,25])。
```

同样，这种算法在概率上 是完整的。但在扩展1000节点后，树基本上是一个乱七八糟的节点点，都在彼此的上面。



我们现在还没有接近目标，而且这也不是一个很难的问题。

虽然生成可行点树的想法有明显的优势，但我们已经失去了在一个节点（因此也是一个空间区域）被探索过后交叉AE的能力。看来，为了使随机算法有效，我们至少需要某种形式的启发式方法来鼓励节点扩散和探索空间。

12.2.1 快速探索的随机树 (RRTs)

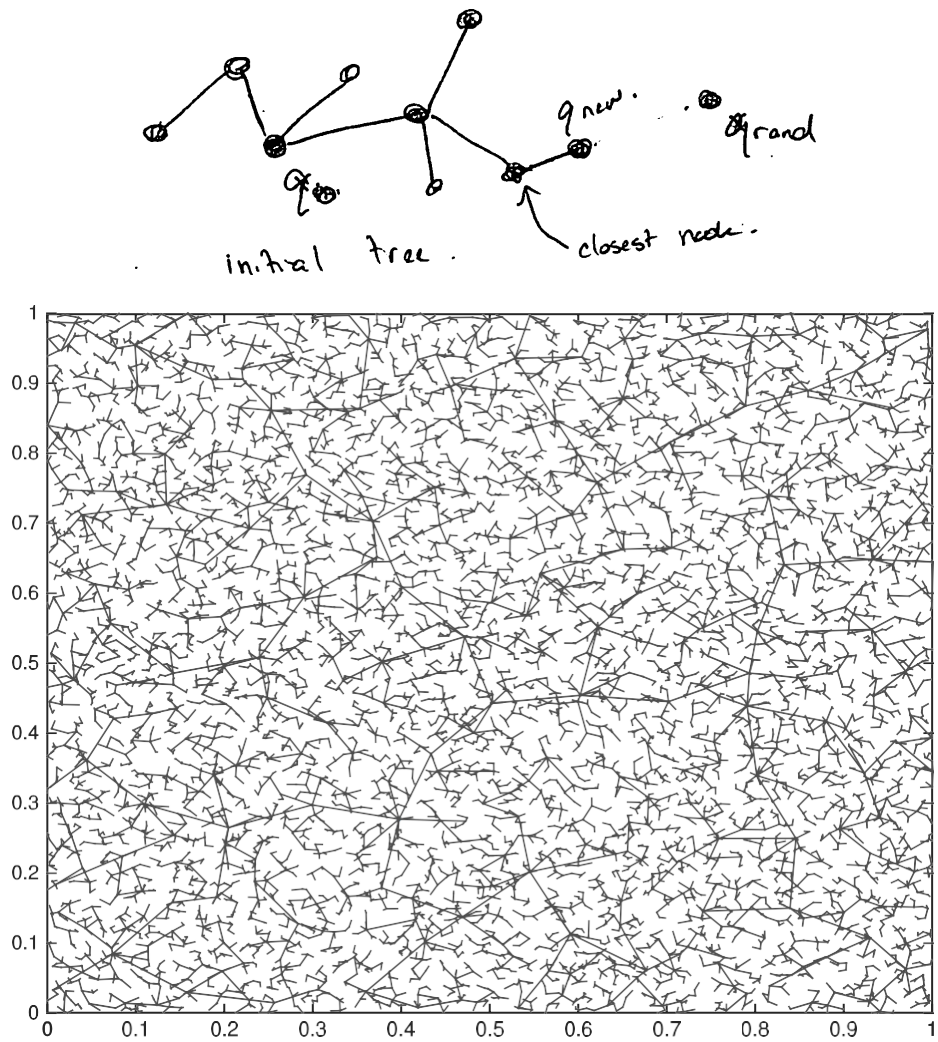


图 -12.3 ([点击这里观看动画](#))

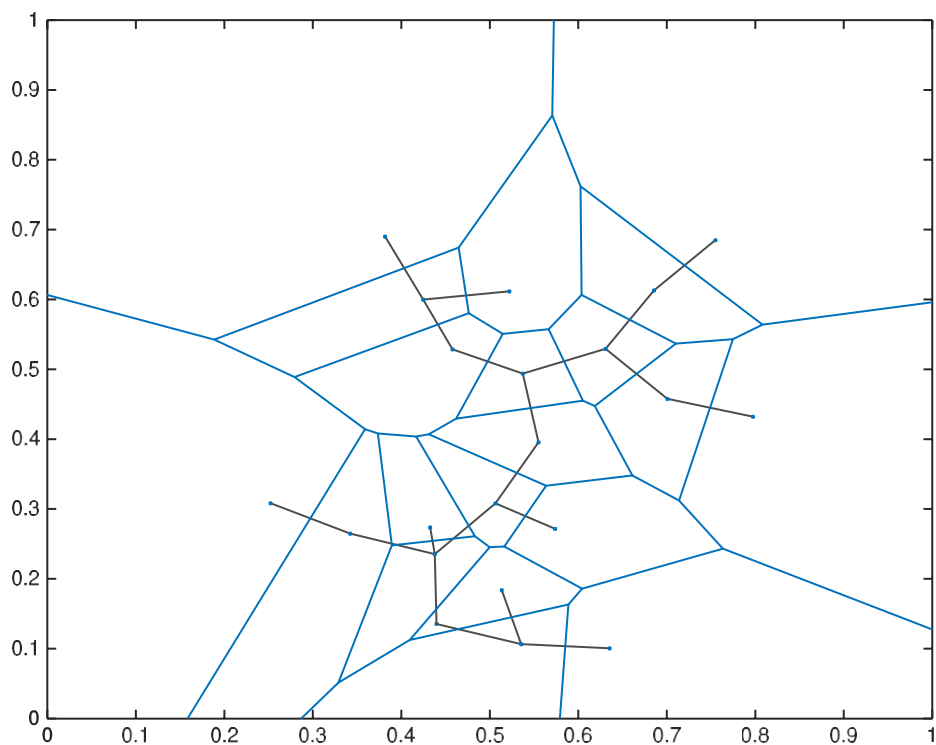


图 -12.4 ([点击这里观看动画](#))

12.2.2 带动力的机器人的RRTs

12.2.3 变化和扩展

用PRMs、RRT...*、RRT-sharp、RRTx

进行多查询规划。 ...

动力学-RRT*, LQR-RRT(*) 复杂度界限和分

散性限制

12.2.4 讨论

还不确定随机性是否是这里的根本，或者是否是在我们更好地理解几何和动态规划之前的一个临时 "拐杖"。

12.3 分解方法

单元分解...混合整数规

划.

复杂环境的近似分解（如IRIS）。

12.4 练习题

演习 (12.1 RRT 规划)

在这个 [笔记本](#) 中，我们将为快速探索随机树 (RRT) 编写代码。在这个实现的基础上，我们还将实现 RRT*，这是 RRT 的一个变种，可以向最优解收敛。

- a. 实施 RRT
- b. 实施 RRT*。

参考文献

1. Steven M. LaValle, "Planning Algorithms", 剑桥大学出版社。2006.

[上一章](#)

[目录](#)

[下一章](#) [可访](#)

[问性](#)

© Russ Tedrake, 2021