第 5 章　自适应和模型预测控制

在本章，我们讨论在值空间近似框架中的一些核心的控制系统设计方法论。特别地，在下面两节中，我们将讨论具有未知或变化的问题参数的问题，并简要综述一些主要类型的自适应控制方法。然后我们将聚焦在基于在线重规划的机制上，包括使用滚动。这里的思想是使用值空间近似机制/牛顿步替代对控制器的完整重新优化，以应对系统参数的变化；我们已在第 1 章注意到这种可能性。后续，在 5.3 节和 5.4 节，我们将讨论模型预测控制法及其与值空间近似、牛顿法、自适应控制及随之而来的稳定性问题的关联。

5.1 具有未知参数的系统——鲁棒和 PID 控制

我们到目前为止讨论处理的问题具有已知和不变的数学模型，即系统方程、费用函数、控制约束和扰动的概率分布是精确已知的。数学模型可以通过显式的数学公式和假设获得，或者通过可以模拟所有在模型中涉及的数学运算的计算机程序来获得，包括计算期望值所使用的蒙特卡洛仿真。从我们的视角，数学模型是通过闭式数学表达式或者通过计算机仿真并没有差别：我们讨论的方法在两种情形下都是可用的，只是它们对于给定问题的适用性可能受到数学公式的可用性的影响。

然而在实际中，常见的是系统涉及的参数或者不精确已知或者随时间变化。在这样的情形下，重要的是设计出可以应对参数变化的控制器。这样的方法论通常称为*自适应控制*，这是一个复杂而多面的主题，有许多应用和相当长的历史。[①]

我们也应当注意到未知的问题环境是强化学习的人工智能视角的重要部分。特别地，引用 Sutton 和 Barto 的书 [SuB18] 中的话，“从与环境的交互中学习是几乎在所有学习和智能理论之下的一个奠基性的思想。”与环境交互的思想通常关联到通过探索环境来识别其特征。在控制理论中这经常被视作*系统辨识*方法论的一部分，旨在构造动态系统的数学模型。系统辨识过程经常与控制过程结合来处理未知或者变化的问题参数。这是随机最优和次优控制中最有挑战性的领域，自 20 世纪 60 年代早期以来就被深入研究了。

鲁棒和 PID 控制

给定假设标称的动态规划问题模型进而获得的控制器的设计，一种可能性是简单地忽略问题参数的变化，然后尝试设计一个在整个参数变化范围内都足够的控制器。这有时被称为*鲁棒控制器*。鲁棒控制器不需要跟踪变化的问题参数。它被设计成能够应对参数的变化，且在实际中它经常倾向于处理最坏的情形。

处理连续状态问题的一种重要的经得起时间考验的鲁棒控制方法是 PID（*比例-积分-微分*）*控制器*；例如见 Aström 和 Hagglund 的书 [AsH95]、[AsH06]。特别地，PID 控制旨在当系统参数在相对较为宽广的范围内变化时将单入单出动态系统的输出保持在一个设定点的周围或者遵循一个给定的轨迹。在其最简单的情形中，PID 控制器有三个标量参数，可以通过多种方法确定，其中一些是手动/启发式的方法。PID 控制广泛应用且有许多成功的应用，尽管其应用范围主要局限在相对简单的单入单出连续状态控制系统上。

---

① 设计自适应控制器的难度经常被低估。除去其他，它们让离线训练和在线对弈之间的权衡变得更加复杂，我们在第 1 章中与阿尔法零的联系中讨论到这一点。值得记住尽管学会高质量地下棋是相当有挑战性的，但是对弈的规则是稳定的且在下棋的过程中不会发生不可预期的变化！具有系统参数变化的问题可能更加有挑战性！

综合系统辨识与控制

鲁棒控制机制，例如 PID 控制，并不努力尝试保持数学模型并在未知的模型参数变化时进行跟踪。取而代之，我们可以在控制器中引入一种机制测量或者估计未知的或者变化的系统参数，从而恰当地变化控制以应对。[①]

需要注意的是，更新问题参数未必需要特殊的算法。在许多情形中问题参数的取值为有限多个可能值，且事先已知（例如参数取值可能对应于车辆的特定控制、机械臂的移动、飞行器的特定飞行模式等）。一旦控制机制确定了问题参数的变化，就可以将变化融入值空间近似的机制中，在策略滚动的情形中，可以切换到对应的事先设计好的基础策略。

在本章后续内容中（包括我们在 5.3 节中关于 MPC 的讨论），我们将假设存在一种机制学习（可能是不完美的并且通过某些未指定的步骤）随时间变化的系统的模型。我们将宽泛地称这一学习过程为经典的名称*系统辨识*，但是我们将不介绍特定的辨识方法，记住这样的方法可能是不精确的且有挑战性的，但是也可能是快速的和简单的，这取决于手上的问题。

一个明显合理的机制是将控制过程分成两个阶段，*系统辨识阶段*和*控制阶段*。在第一个阶段未知的参数被估计出来，而控制并不考虑估计的临时结果。第一个阶段的最终的参数估计值然后用于在第二个阶段实现最优的或者次优的策略。

这一交替的估计和控制可以在系统的运行中重复多次以考虑后续的参数变化。注意不需要引入估计阶段和控制阶段的硬切分。它们可以同时进行，只要有必要，新的参数估计可以在后台生成，然后引入控制过程中，见图 5.1.1。

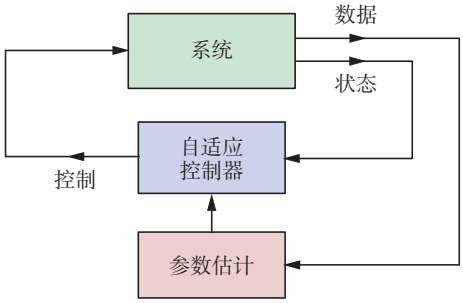

图 5.1.1 并发参数估计和系统控制的机制示意图。系统参数被在线估计出来，在需要的时候估计值被传给控制器（例如，在估计值显著变化之后）。这一结构也称为间接自适应控制。

该方法的不足之处是有时不易确定何时终止一个阶段并开始另一个。还有一个更加基本的难点是控制过程可能让一些未知的参数对于估计过程不可见。这被称为*参数可辨识性*问题，在几个参考文献中针对自适应控制进行了讨论。处理可辨识性问题的在线参数估计算法，已经在控制理论文献中详细讨论了，但是对应的方法论是复杂的且超出了本书的范

① 在自适应控制文献中，涉及参数估计的机制有时称为*间接法*，而不涉及参数估计的机制（例如 PID 控制）被称为*直接法*。引用 Aström 和 Wittenmark 的书 [AsW08] 中的原话“间接法是那些使用估计的参数计算所需要的控制器参数的方法”（见图 5.1.1）。本节后续描述的方法以及在下一节讨论的基于滚动的自适应控制方法应该被视作间接法。

畴。然而，假设可以让估计阶段某种意义上可用，那么我们可以用在线重新规划过程的形式使用新估计出来的参数重新优化控制器。

不幸的是，这类在线重新规划存在另一个难点：使用新辨识出来的系统模型在线重新计算最优的或者近优的策略可能是困难的。特别地，耗时的或者需要大量数据的方法，如涉及训练神经网络或者离散、整数控制约束的方法，是不可能被使用的。更简单的一种可能性是使用滚动，我们将在下一节讨论。

5.2 值空间近似、滚动和自适应控制

我们现在将考虑处理未知或者变化参数的一种方法——基于滚动和在线重新规划。我们已经在第 1 章中注意到这种方法，在那里我们强调了快速的在线策略改进的重要性。

假设某些问题参数随时间变化，而可能在一定的数据采集和估计的延迟之后控制器注意到这一变化。问题参数重新计算或者变得已知的方法对于接下来的讨论是不重要的。它可能涉及有限形式的参数估计，其中未知的参数被在一些时段上的数据采集“跟踪”，其主要目的是处理参数可辨识性问题；或者可能涉及控制环境的新的特征，如服务器的数量变化或者服务系统中任务数量的变化。

我们于是忽略参数估计的具体细节，关注基于新获得的参数在线重新规划控制器。这一修订可以基于任意的次优方法，但是采用某个基础策略的滚动尤其引人注目。这一基础策略可以是固定的鲁棒控制器（例如某种形式的 PID 控制）或者它可以随时间变化不断更新（在后台，在某个未指定的原理的基础之上），此时滚动策略将同时在应对变化的基础策略和应对变化的参数中修订。

这里滚动的优势是简单、可靠和相对快速。特别地，它不需要复杂的训练过程，如基于使用神经网络或者其他的近似架构，所以针对参数变化没有新的策略被显式计算出来。一种替代方法是，在当前状态可用的控制通过单步或者多步最小化进行比较，其费用函数近似由基础策略提供（参见图 5.2.1）。

需要考虑的另一个问题是滚动策略的稳定性和鲁棒性。通常可以在宽松的假设条件下证明，如果基础策略在一个参数取值范围内是稳定的，那么滚动策略也是稳定的；这可以从图 3.4.3 中推断出来。相关的思想在控制理论文献中有相当长的历史；见 Beard[Bea95]，Beard、Saridis 和 Wen[BSW99]，Jiang 和 Jiang[JiJ17]，Kalise、Kundu 和 Kunisch[KKK20] 以及 Pang 和 Jiang[PaJ21]。

在自适应控制的背景中使用滚动的主要要求是滚动控制的计算应当对于两个阶段之间的执行是足够快的。我们注意到加速、截断或者简化版本的滚动，以及并行计算，可以用于满足这一时间约束。

在给定问题参数的取值集合时，若滚动控制比最优控制更易计算，那么在这些情形中通过滚动和在线重新规划进行自适应控制是有意义的。这样的情形涉及非线性系统和难以处理的约束条件（例如整数约束）。

我们之前讨论过简单的一维线性二次型问题。下面的例子展示了如何在这类问题中使用滚动进行在线重新规划。这个例子的目的是用解析的方式展示以标称参数集合下的最优策略为基础策略进行滚动，当参数从其标称值发生变化时，滚动方法的性能依然良好。这

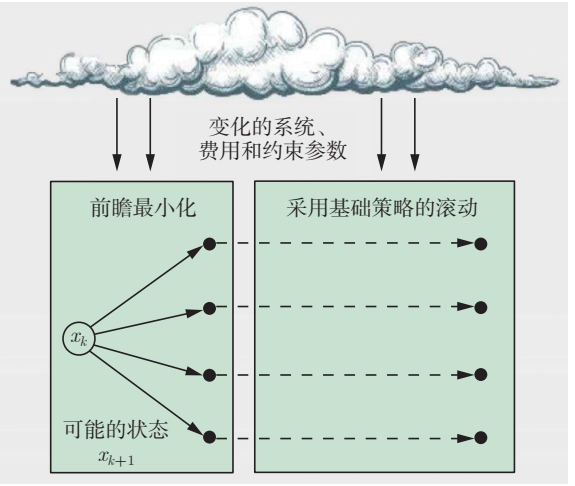

图 5.2.1 基于滚动的在线重新规划的自适应控制示意图。单步前瞻最小化后基于保持不变的基础策略进行仿真。系统、费用和约束参数随时间变化，其最新值用于前瞻最小化和滚动操作。采用多步前瞻最小化和末端费用近似的截断滚动也是可能的。基础策略也可基于多种准则修订。对于本节的讨论，我们可以假设所有变化的参数信息由某个超出我们控制范畴的计算和传感器“云”提供。

一性质对于线性二次型问题在实用中并没有什么用，因为当参数变化时，可以闭式地计算出新的最优策略，但是可以展示滚动方法在其他情形下（例如有约束的线性二次型问题）的鲁棒性。

例 5.2.1（线性二次型问题基于滚动的在线重新规划）

考虑涉及如下线性系统的确定性无折扣无限时段线性二次型问题

$$x_{k+1} = x_k + bu_k$$

和二次费用函数

$$\lim_{N\to\infty} \sum_{k=0}^{N-1} (x_k^2 + ru_k^2)$$

这是之前一节的一维问题的特例，$a = 1$ 和 $q = 1$。最优费用函数给定如下

$$J^*(x) = K^* x^2$$

其中 K^* 是黎卡提方程的唯一正解

$$K = \frac{rK}{r + b^2 K} + 1 \tag{5.1}$$

最优策略的形式为

$$\mu^*(x) = L^* x \tag{5.2}$$

其中

$$L^* = -\frac{bK^*}{r + b^2 K^*} \tag{5.3}$$

作为一个例子，考虑对应于标称问题参数 $b = 2$ 和 $r = 0.5$ 的最优策略：这是式 (5.2)∼式 (5.3) 的策略，其中 K 作为式 (5.1) 的二次黎卡提方程在 $b = 2$ 和 $r = 0.5$ 时的正解。对于这些标称参数值，我们有

$$K = \frac{1+\sqrt{6}}{4} \approx 1.11$$

从式 (5.3) 也可得到

$$L = -\frac{2+\sqrt{6}}{5+2\sqrt{6}} \tag{5.4}$$

我们将考虑 b 和 r 的取值的变化而保持 L 恒定为之前的取值，在 b 和 r 变化时比较下面三个费用函数的二次费用系数。

（a）最优费用函数 K^*x^2，其中 K^* 由式 (5.1) 的黎卡提方程的正解给定。

（b）对应于如下基础策略

$$\mu_L(x) = Lx$$

的费用函数 K_Lx^2，其中 L 由式 (5.4) 给定。这里，我们有 [参见 4.1 节]

$$K_L = \frac{1+rL^2}{1-(1+bL)^2} \tag{5.5}$$

（c）对应于用策略 μ_L 作为基础策略获得的滚动策略

$$\tilde{\mu}_L(x) = \tilde{L}x$$

的费用函数 $\tilde{K}_Lx^2$。使用之前推导的公式，我们有 [参见式 (5.5)]

$$\tilde{L} = -\frac{bK_L}{r+b^2K_L}$$

和 [参见 4.1 节]

$$\tilde{K}_L = \frac{1+r\tilde{L}^2}{1-(1+b\tilde{L})^2}$$

图 5.2.2 对 r 和 b 在一个范围内取值时展示了系数 K^*、K_L 和 $\tilde{K}_L$。我们有

$$K^* \leqslant \tilde{K}_L \leqslant K_L$$

差分 $K_L - K^*$ 指示了策略 μ_L 的鲁棒性，即，通过忽略 b 和 r 的取值变化并继续使用策略 μ_L 导致的性能损失，注意 μ_L 对于标称值 $b = 2$ 和 $r = 0.5$ 是最优的，但是对于 b 和 r 的其他取值是次优的。差分 $\tilde{K}_L - K^*$ 指示了因为通过滚动进行在线重新规划而不是使用最优重新规划导致的性能损失。最终，差分 $K_L - \tilde{K}_L$ 指示了因为使用滚动进行在线重新规划而不是保持策略 μ_L 不变带来的性能改进。

注意到图 5.2.2 展示了误差率

$$\frac{\tilde{J} - J^*}{J - J^*}$$

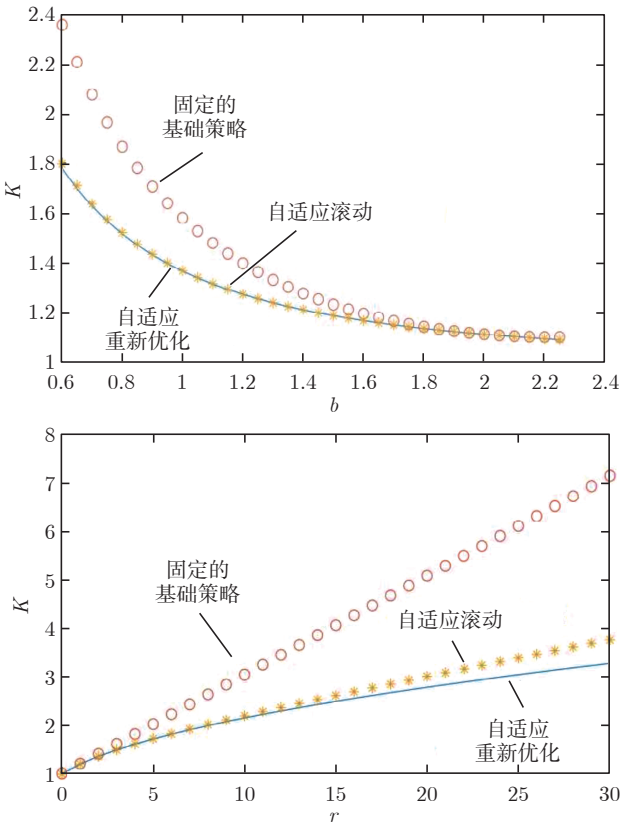

图 5.2.2 在变化的问题参数下使用滚动的示意图。对于 $r = 0.5$ 且 b 变化和 $b = 2$ 且 r 变化的这两个情形展示了二次费用系数 K^*（最优的，用实线表示）、K_L（基础策略，用圆圈表示）和 $\tilde{K}_L$（滚动策略，用星号表示）。L 取值固定在 $b = 2$ 且 $r = 0.5$ 时的最优值 [参见式 (5.4)]。即使当基础策略距离最优策略很远时，滚动策略性能仍非常接近最优。注意，如图所示，我们有

$$\lim_{J \to J^*} \frac{\tilde{J} - J^*}{J - J^*} = 0$$

其中对于给定的初始状态，$\tilde{J}$ 是滚动性能，J^* 是最优性能，J 是基础策略性能。这是在滚动之下的牛顿法的超线性二次收敛速率的结果，并且保证了滚动性能以远快于基础策略性能的速度接近最优值。

的变化情况，其中对于给定的初始状态，$\tilde{J}$ 是滚动性能，J^* 是最优性能，J 是基础策略性能。因为滚动方法之下的牛顿法的超线性二次收敛速率，这一比例随着 $J - J^*$ 变得更小而趋向 0。

5.3 值空间近似、滚动和模型预测控制

这一节简要讨论 MPC 方法，着重关注其与值空间近似和滚动算法的关系。我们将主要关注无折扣无限时段确定性问题，涉及如下系统

$$x_{k+1} = f(x_k, u_k)$$

其状态 x_k 和控制 u_k 是有限维的向量。假设每阶段的费用非负

$$g(x_k, u_k) \geqslant 0, \forall (x_k, u_k)$$

（如正定二次费用）。存在控制约束 $\mu_k \in U(x_k)$，且为了简化后续讨论，我们将一开始不考虑状态约束。假设系统可以无费用地保持在原点，即

$$f(0,\bar{u}_k)=0, g(0,\bar{u}_k)=0, \text{ 对某个控制}\bar{u}_k \in U(0)$$

对于给定的初始状态 x_0，我们希望获得一个序列 $\{u_0,u_1,\cdots\}$ 满足控制约束且最小化总费用。

这是控制系统设计中的一个经典问题，称为**镇定问题**，其目标是在面对扰动和参数变化时，将系统的状态保持在原点（或者更一般的某个所希望的设定点）附近。在该问题的一种重要的变形中，存在额外的状态约束，形式为 $x_k \in X$，且要求不仅是在当前时刻而且是在未来时刻保持状态在 X 内部。我们稍后在这一节处理这个问题。

经典形式的 MPC——视作滚动算法

我们首先关注一种经典形式的 MPC 算法，其形式由 Keerthi 和 Gilbert[KeG88] 给出。在这一算法中，在每个遇到的状态 x_k，我们施加如下计算出来的控制 $\tilde{u}_k$，见图 5.3.1。

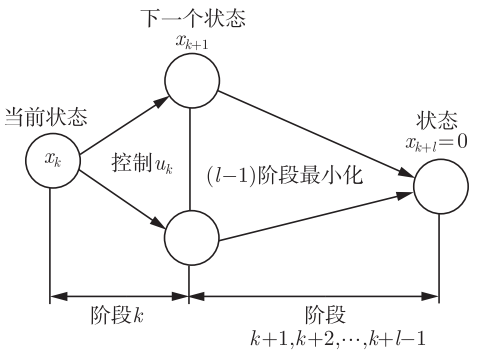

图 5.3.1　用经典形式的 MPC 在状态 x_k 求解这一问题的示意图。我们最小化下面 l 个阶段的费用函数，并施加约束，要求 $x_{k+l}=0$。然后应用最优序列的首个控制。在滚动的语境下，对 u_k 的最小化是单步前瞻，对驱使 x_{k+l} 到 0 的 $u_{k+1},u_{k+2},\cdots,u_{k+l-1}$ 的最小化是基础启发式规则。

（a）我们求解一个 l 阶段最优控制问题，使用相同的费用函数并且要求在 l 步之后状态被驱使到 0，即 $x_{k+l}=0$。这个问题是

$$\min_{u_t,t=k,k+1,\cdots,k+l-1} \sum_{t=k}^{k+l-1} g(x_t,u_t) \tag{5.6}$$

系统约束

$$x_{t+1}=f(x_t,u_t), t=k,k+1,\cdots,k+l-1 \tag{5.7}$$

控制约束

$$u_t \in U(x_t), t=k,k+1,\cdots,k+l-1 \tag{5.8}$$

末端状态约束

$$x_{k+l}=0 \tag{5.9}$$

这里 l 是一个整数且满足 $l>1$，这在很大程度上通过实验的方式选取。

（b）如果 $\{\tilde{u}_k,\tilde{u}_{k+1},\cdots,\tilde{u}_{k+l-1}\}$ 是这个问题的最优控制序列，我们施加 $\tilde{u}_k$ 并且丢弃其他的控制 $\tilde{u}_{k+1},\tilde{u}_{k+2},\cdots,\tilde{u}_{k+l-1}$。

（c）在下一个阶段，一旦下一个状态 x_{k+1} 出现之后，我们重复这一过程。

为了建立之前的 MPC 算法与滚动之间的联系，我们注意到由 MPC 隐式使用的单步前瞻函数 $\tilde{J}$[参见式 (5.6)] 是某种稳定的基础策略的费用函数。这个策略在 $l-1$ 个阶段（而不是 l 个阶段）后将系统状态驱使到 0，并且在此之后让状态保持在 0，满足状态和控制约束，并且最小化所关联的 $(l-1)$ 阶段费用。这一 MPC 的滚动视角首先在作者的论文 [Ber05] 中进行了讨论。这对于用牛顿法解释近似动态规划、强化学习、滚动方法是有用的。特别地，一个重要的结论是 MPC 策略是稳定的，因为正如我们已经在 3.2 节中讨论的，采用稳定基础策略的滚动获得一个稳定策略。

我们也可以等价地将之前的 MPC 算法视作采用 $\bar{l}$ 步前瞻的滚动，其中 $1<\bar{l}<l$，并且采用了在 $l-\bar{l}$ 阶段后将状态驱使到 0 并且在之后将状态保持在 0 的基础策略。这提出了涉及采用末端费用函数近似的截断滚动的 MPC 的变形，我们稍后讨论。

注意当问题参数变化时，考虑在线重新规划是自然的，正如我们之前的讨论。特别地，一旦系统的新估计或费用函数参数变得可用，MPC 可以通过将新的参数估计引入上面（a）中的 l 阶段的优化问题来应对。

5.4 末端费用近似——稳定性问题

在一种常见的 MPC 变形中，式 (5.6) 的 l 阶段 MPC 问题中在 l 步之后将系统状态驱使到 0 的要求，可以替换为末端费用 $G(x_{k+l})$。所以在状态 x_k，我们求解如下问题

$$\min_{u_t,t=k,k+1,\cdots,k+l-1}\left[G(x_{k+l})+\sum_{t=k}^{k+l-1}g(x_t,u_t)\right] \tag{5.10}$$

而不是式 (5.6) 的问题，其中要求 $x_{k+l}=0$。这一变形也可以被视作采用单步前瞻的滚动，以及一个基础策略，后者在状态 x_{k+1} 应用最小化

$$G(x_{k+l})+\sum_{t=k+1}^{k+l-1}g(x_t,u_t)$$

的序列 $\{\tilde{u}_{k+1},\tilde{u}_{k+2},\cdots,\tilde{u}_{k+l-1}\}$ 的第一个控制 $\tilde{u}_{k+1}$。在跳出滚动的语境中这也可以被视作采用了 l 步前瞻最小化和值空间近似（由 G 近似末端费用）。所以之前的 MPC 控制器可能拥有比 G 更接近 J^* 的费用函数。正如我们在第 3 章中讨论过的，这优于在值空间近似之下的牛顿法的超线性二次收敛速率。

一个重要的问题是，如何选择末端费用近似让最终的 MPC 控制器是稳定的。我们在 3.3 节中关于值空间近似机制的稳定域的讨论适用于这里。特别地，在本节的非负费用假设下，若 $TG\leqslant G$，或者等价地（通过使用在第 3 章中介绍的抽象动态规划的符号）

$$(TG)(x)=\min_{u\in U(x)}\{g(x,u)+G\left(f(x,u)\right)\}\leqslant G(x),\forall x \tag{5.11}$$

正如在 3.2 节中所注释的，那么 MPC 控制器将是稳定的。这一条件对于 MPC 控制器的稳定性是充分但非必要的。图 5.4.1 提供了图示化的解释。其展示了条件 $TG \leqslant G$ 意味着对所有 $l \geqslant 1$ 有 $J^* \leqslant T^l G \leqslant T^{l-1} G$（书 [Ber12] 和 [Ber18a] 提供了对于这一事实的数学证明）。这进一步意味着 $T^l G$ 对所有的 $l \geqslant 0$ 都位于稳定域之内。

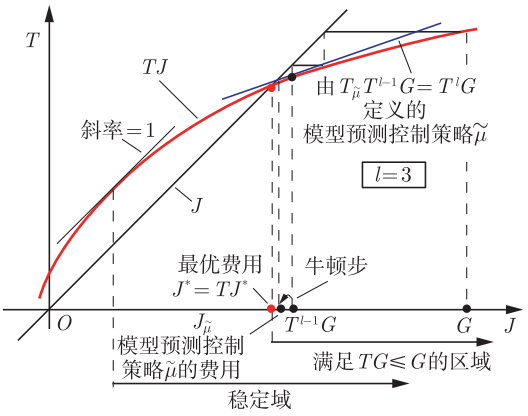

图 5.4.1　条件 $TG \leqslant G$ 或者等价地

$$(TG)(x) = \min_{u \in U(x)} \{g(x,u) + G\left(f(x,u)\right)\} \leqslant G(x), \forall x$$

的示意图。当末端费用函数近似 G 满足这一条件时，其保证了采用 l 步前瞻最小化的 MPC 策略 $\tilde{\mu}$ 的稳定性，定义为

$$T_{\tilde{\mu}} T^{l-1} G = T^l G$$

其中对于一般的策略 μ，T_μ 定义如下（使用第 3 章的抽象动态规划的符号）

$$(T_\mu J)(x) = g(x, \mu(x)) + J\left(f(x, \mu(x))\right), \forall x$$

在本图中，$l = 3$。

我们也期待随着前瞻最小化的长度 l 增大，MPC 控制器的稳定性提升。特别地，给定 $G \geqslant 0$，对于充分大的 l，所得的 MPC 控制器可能是稳定的，因为 $T^l G$ 通常收敛到 J^*，后者位于稳定域内。已知这类结果在 MPC 框架中在多种条件下成立（见 Mayne 等的论文 [MRR00]、Magni 等的论文 [MDM01]、MPC 书 [RMD17] 和作者的书 [Ber20a]3.1.2 节）。在这一上下文中，我们在 4.4 节和 4.6 节的稳定性的讨论也是相关的。

在另一种 MPC 的变形中，在末端费用函数近似 G 之外，我们使用截断滚动，这涉及连续 m 步运行某个稳定的基础策略 μ，见图 5.4.2。这非常类似标准截断滚动，唯一的区别在于当控制空间是无限时式 (5.10) 的前瞻最小化问题的计算求解可能变得复杂。正如在 3.3 节中讨论的，增加截断滚动的长度可扩大 MPC 控制器的稳定域。原因是通过增大截断滚动的长度，我们将牛顿步的起点推向稳定策略的费用函数 J_μ，后者位于稳定域之内（因为 $TJ_\mu \leqslant T_\mu J_\mu = J_\mu$）；也见 4.7 节中关于线性二次型问题的讨论。基础策略可以用于处理状态约束；见 Rosolia 和 Borelli 的论文 [RoB17]、[RoB19] 及作者的强化学习一书 [Ber20a] 中的讨论。

MPC 的具有多个末端状态和基础策略的滚动变形

在 Li 等的论文 [LJM21] 中提出了另一种 MPC 的变形，与其在 l 步结束时将状态驱动到 0，我们考虑在 l 步时段的末尾有多个末端系统状态，以及为了滚动使用多个基础策