having done that doing the short break here we go bring a number to be that you may address for the TA they're teaching the area as a possible [Music] this [Music] yeah [Music] [Music]

DP ALGORITHM

okay maybe we can get started

我们要开始了

so I will pick up again on this dynamic programming algorithm

我要再重复一遍动态规划算法的内容

we define JK's of xk it will be the optimal cost the tail problems starting at xk and given the solution to all the tail problems of starting at k plus one we can generate by the principle of optimality essentially the optimal cost of the tail problem starting at time k go one step and then go optimally on the tail problem at time k plus one and minimize choose u to minimize the sum of the two

我定义函数J\_k(x\_k)为从状态x\_k开始的尾问题的最优成本，并且给定了一个计算所有从状态x\_{k+1}开始的尾问题的最优解，就可以根据最优性准则生成从x\_k开始的尾问题的最优成本，再观察J\_{k+1}，就可以选择控制u去最小化这两项的和

ok so this is an algorithm that a computer can understand but it can be a very very difficult algorithm to execute

这是一个计算机很容易理解的算法，但是它执行起来非常困难

and that's the problem here with dynamic programming recognized in the very early days

这就是动态规划很早以前就被意识到的问题

let's see what are the difficulties with this algorithm

让我们来看看这个困难是什么

well first of all the curse of dimensionality exponential growth of the computation and storage requirements as the number of state variables and control variables increases

首先，计算量与存储量的维数灾是随着状态变量与控制变量成指数增长的

so for example if you have motion of a vehicle in some three dimensional space we said the six state variables okay six state variables if you discretize each axis with a hundred points this is a hundred to the sixth discretization points

举个例子，你想要在某个三维空间中开车，系统状态有六个维度，如果你对每一个维度离散化为100个值，你就有100^6的离散点

so at each one of these tail to the six discretization points you have to calculate this expectation okay and then minimize overall you that's a lot of minimizations to do and this is only one time step if you go back many many steps the requirements increase proportionally

所以每一个六维状态的子问题，你必须计算他们的期望成本，然后在所有的控制u中选择令成本最小的，这就需要做很多次最小化操作，这仅仅是一个阶段需要进行的操作，如果你要对很多阶段进行优化，资源的需求就会增加

so there's this curse of dimensionality which manifests itself on many many computations to be done here and there is also a problem of storage because if you have you have to store the say a hundred to the six numbers to store this function JK plus 1 so again and a hundred to the sixth for problem is relatively small if we will look later at some problems that are much much bigger than that so that's the curse of dimensionality it's a real formidable object object or obstacle

所以这就是维数灾的体现，他要进行非常多次的计算，存储这些数据同样存在问题，如果你需要存储100^6个最优成本函数J\_{k+1}的数据，这还不算什么，如果我们需要处理大得多的问题，需要存储的数据量就很可怕了

PRACTICAL DIFFICULTIES OF DP

okay the curse of modeling here in this algorithm we need to have we transition the probability distribution of wk and that's easy that's not so easy to get you need to have an equation here FK that may also be difficult to have in a give a practical situation so it would be nice if we did not have to calculate this entire thing in detail then calculate the whole thing and this is what you learning does somebody mentioned Q learning earlier later we're going to see that the Q learning technique is a dynamic program entitled up of techniques that is model free you don't need to have a model of the system or a model of the uncertainties to apply

还有一个困难，是建模困难，在动态规划算法中我们需要直到状态转移的概率分布，或者是随机变量w\_k的概率分布，这并不容易，你需要知道方程f\_k，在一个给定的情境下，这很难，所以如果我们不需要计算这些东西，就可以直接计算整个成本函数。你接下来学到的有些人提到的Q-learning就是一个动态规划就能够完成这件事情，这是一个不需要模型的方法，你不需要知道不确定问题的系统模型

it here's another difficulty there may be real-time solution constraints oftentimes you need you have really a family of problems and you don't know which problem you're going to solve until the very last minute so for example suppose you want to dispatch some trucks to make deliveries in various parts of a certain area you don't know you may have a good map of the area but you don't know where the deliveries are going to be until the very last minute when the orders come in so this has to do also with real-time operation the problem being revealed at you in real-time however if you have real-time situations required fast solution and the dynamic programming algorithm is not fast you can may take a very long time to produce a solution is basically an offline algorithm and many times you have online situations moreover the problem data may change as the system is controlled for example in the city in deliveries there may be some new deliveries that are scheduled as your trucks are moving ok so the problem of the data of the problem may change on you and then you need to do online replanting and the dynamic programming algorithm is not suitable for this sort of thing

另一个问题就是实时问题，现在你有很多问题需要解决，但是在这个问题出现之前你并不知道你接下来要解决的是哪一个问题，比如你想派遣一些卡车到一个地区的不同地方运货，你有这些地区的最优派遣方案，但是在派遣需求产生之前你不知道哪些位置需要货，所以这些需求产生的时候你需要进行实时操作，动态规划没有这么快，你可以使用非常长的时间计算一个offline的方案，但是很多情况下你要工作在online的条件下，还有一个问题，就是这种问题的数据可能会随着控制被系统执行而改变，比如在一个城市内运货，有很多新的运输任务在卡车正运输其他货物时被调度，所以这些问题的数据可能会变，你需要实时地调整这些方案，动态规划就不适用于这样的场景

all of these difficulties are our motivation for considering approximations to the algorithm and also simulation to deal with this curse of modeling

这些问题让我们有足够的动机去考虑动态规划的近似算法，或通过仿真去解决建模困难的问题

A MAJOR IDEA: COST APPROXIMATION

so here's a major idea of approximation it's not the only one but it's an important one

这是一个主要的近似思路，不是唯一的一个，但是很重要的一个

instead of using the optimal cost to go function JK plus one in the algorithm we replace it by an approximation that is more easily computable in other words we use a sub optimal policy mu bar not new star nu bar which at time K attains the minimum of this expression now how is this expression different than the one we had before instead of having j k plus one we have J tilde k plus one an approximation so the thing that gives us trouble this JK plus one we approximate it with something and I haven't told you how and then use the minimizing control as a substitute for the object this is called approximation in value space okay we approximate the optimal values here by something and we go from there

我们使用一种更便于计算的近似函数来代替最优成本函数J\_{k+1}，即使用时间k时的次优策略mu bar 获得期望的最小值，这种表述和我们之前提到的描述是不一样的，时间k+1时我们使用J\_k tilde 来近似原来的J\_k，我不告诉你们近似的细节，思路是用这种近似来求最优控制u作为原问题的最优解的替代解，这叫做值空间近似，我们用一种方式近似这个最优值，然后在这个近似的基础上进行求解

how can we compute an approximation to the optimal cost to go well we may replace the problem with a simpler problem and use as J tilde K to be the optimal cost of that simpler problem that simpler problem may be obtained by discarding some dynamics by simplifying the trying to the probability distributions of the problem make the problem simpler to to to solve exactly perhaps by dynamic programming be helped by other means and stick it in here to get the benefit of feedback

我们可以使用一个更简单的，用J\_k tilde 来近似cost to go的问题，代替原问题，简化可以通过丢掉一些动态性与概率分布完成，这样我们就可以使用动态规划以及动态规划的反馈信息的优势来求解这个更简单的问题，也可以获得近似最优成本

here's another example J tilde K is a high dimensional object and you remember that the state space may be very large there may be many many states so this is a function that is represented by a list of many many values okay it may be possible to represent this function approximately with a parametric form a function that depends on some parameters a small number of parameters which are tuned or trained by some heuristic or systematic scheme and this is going to be a major object for us to look into in a major portion of new Rosner learning the approximate dynamic programming neural dynamic programming deals with parametric approximations of the cost to go function JK

这里有另一个例子，J\_k tilde 是一个高维目标函数，你还记得状态空间非常大的情况么，这种情况下函数的数量会非常多，如果使用一个在比较小的参数空间内进行近似的函数，就可以通过使用启发式或者其他方法对对参数进行训练或者调节来近似cost to go，这也是一种很主要的近似方法，强化学习、近似动态规划和神经动态规划都通过参数近似来减小求解难度

he is a third approach which is simple but very effective often called the roll-out approach in rollout we use as J as an approximation J tilde the cost of some other policy not the optimal on some suboptimal policy some heuristic policy which is calculated which is such that you can calculate its cost either analytically or by simulation so you have a heuristic instead of the optimal cost you use the cost of agrestic in there

这是第三种近似方法，虽然简单但是经常很高效，被叫做roll-out方法，roll-out方法使用J tilde 即一些次优策略来近似最优策略的成本，当然这不是最优成本，你可以通过分析或者仿真计算他的成本，比如，使用一个启发式策略来代替最优策略

the rollout as I mentioned is a is a very effective technique very simple but we're not going to go into it much in this course because we just don't have time but let me spend one slide on it because I think it's an important algorithm

我提到的roll-out是一种非常简单并且非常高效的技术，但是我们不会在这个课程里深入地介绍他，因为我们没有足够的时间，不过我还是要用一整张slide来介绍一下，因为我觉得它是一个很重要的算法

ROLLOUT ALGORITHMS

okay so here's our all-out algorithm I have some heuristic policy which I call the base policy and I calculate somehow the cost of that policy associated with the next states that I may land using control UK and I minimize this expression so I calculate this in some way

下面我要介绍roll-out算法，我设计了一些启发式策略，我把它叫做基准策略，然后我根据采取某个控制后下一个阶段可能出现的状态计算这个策略的成本，再对bellman表达式进行优化。

and here's the important property the roll out policy the roll out the policy that you are going to obtain from this minimization is always no worse and usually much better than the base policy so starting from any heuristic if you can calculate this cost you can stick it in here and get a better heuristic often significantly better offer pretty close to optimal and this is suitable for online operation because as you go on then what you need in order to apply controller time UK you need to calculate this expression here and you can do it by forward simulation often so you are at IMAX that at time estate XK you calculate you consider all possible use and then from the state whether you land you run the heuristic from there and get a number that will give you a number for every control and you pick the control that minimizes

这是roll-out策略一个很重要的性质，我们根据roll-out策略最小化得到的控制一般不会更坏，通常会比基准策略更好，所以从这个启发式策略出发，如果你算出这个策略的成本，那么你就可以执行这个策略，这通常可以获得一个更有意义且更接近最优解的控制，这种方法适合于在线操作，因为这样你可以采取控制u\_k，你需要通过前向仿真计算法的成本，然后根据执行某个控制可能出现的状态计算可能产生的成本，然后选择一个成本最小的控制

there is a potential difficulty calculating the cost of a heuristic may itself be computationally intensive and one possibility is to use Monte Carlo simulation if the problem is stochastic however if the problem is deterministic then this calculating this quantity does not require Monte Carlo simulation it can be done by a single run of the deterministic heuristic and this is a this is this this is important in discrete optimization problems where you have combinatorial problems you're breaking them up into decisions over stages and you have a heuristic which are at certain point you've calculated the solution a partial solution to calculate the remainder of the solution you use this heuristic and calculating the associated cost requires only a single trajectory simulation

这种方法有一个潜在的问题，计算启发式策略的成本的时候可能需要进行大量的计算，如果你需要解决的是随机问题，你可以尝试使用蒙特卡罗模拟来计算成本，如果这是一个确定性的问题，就不需要使用蒙塔卡罗模拟而是直接计算成本，这种方法是一种解决离散优化问题的很重要的方法，如果你需要解决一个离散优化问题，你可以把决策变成多阶段的决策，然后使用启发式策略在一个状态轨迹仿真的过程中计算指定点的剩余路径成本获得部分控制方案

there is also a methodology that's very important in control theory model predicted control control system design in these days is done to a large extent with this model predictive control and control turns out to be a special case of rollout with a special heuristic in the special context of this NPC I'm not going to get into this like I can post literature on the website but an interesting connection because MPC has been hugely successful in control theory and it and then this connection indicates of how things may go from here and why rollout is also is could be a very effective technique it's a success stories were all out as well as the control theory

在控制理论中有一个非常重要的方法-模型预测控制，控制系统的设计由于模型预测控制获得了很大的发展，模型预测控制就是一种在特殊的系统中特定情况下使用特定启发式策略的roll-out算法，我不会在这里深入介绍模型预测控制，我会把相关文章放到我的网站上，模型预测控制与roll-out有一个很有趣的联系，因为它已经在控制理论中获得了很大的成功，所以我们说roll-out也可以是一种非常有效的技术

INFINITE HORIZON PROBLEMS

okay so now I'm going to turn to infinite horizon problems for final equalizer problems we have one algorithm we can consider approximations but basically that's it there's not much more Theory for finite horizon problems the interesting theory comes foot to infinite for infinite horizon problems

现在我们要开始讲无穷维问题，在有穷维问题中，我们可以考虑近似我们要解决的问题，但是这只是一个比较基本的方法，没有太多理论来解决有穷维问题，在无穷维问题中有一些很有趣的理论

points an infinite horizon problem is the same as the basic finite horizon operation problem except that the number of stages is infinite and the system and cost is stationary there is no dependence on J if I can turn you back to this model here sorry for the flipping backwards okay so this is our generic fine horizon model if there is no dependence on K here no dependence on K here and the horizon is infinite we get an infinite horizon problem

无穷维问题的形式和基本的有穷维问题是相同的，除了阶段的数量是无穷的和系统与成本是平稳的，平稳指的是他们不依赖于时间k，我把slide往前翻，可以看到通用的有穷维模型，如果它不依赖于时间k，那么这就是一个无穷维问题

okay now we want to minimize in a major class of problems we want to minimize the total infinite horizon cost so we form the finite horizon cost and we introduce a discount factor alpha to the K alpha may be either or less than if it's less than then it discounts it diminishes future cause it makes this expression bounded it keeps it real number

现在我想要最小化一个主要类型的问题，最小化总成本，所以我把有穷维问题的成本函数加进了折扣因子alpha^k，这个因子小于等于一，如果它小于等于一，就可以保证成本的期望有界并且是一个实数

so plugging in a policy for any finite horizon gives us a number and take this number to go to n to go to infinity the sequence of numbers to go to infinity and that gives you the infinite horizon cost associated with a policy and with an initial state

在任意有穷维问题中采用一个策略，会获得一个数，然后我们用这个数去计算成本，当n趋于无穷的时候，这些数的个数也是无穷多个的，这样你就可以获得一个无穷维问题在已知初始状态时的策略成本

and there are several categories of problems here one is discounted problems where alpha is less than and G is bounded okay it's values right within a bounded range and this makes this series convergent okay G is bounded alpha goes down as a geometric progression makes the series convergent and therefore the optimal could be the cost of any policy is finite

无穷维问题有几种不同的类别，一种是折扣问题，当alpha小于1时g是一个有界的实数，即等号右边的值是有界的，这可以保证这个数列是收敛的，即g的值随着时间增长成几何级数减小，这样不管使用什么策略，我们最后得到的最优成本都是一个有界的数值

there are other ways to keep across finite and not explode to infinity even with alpha equals to 1 if there is a termination state consider the following problem you want to go to some gold stay for something where you want that you want to reach and go there in a in with minimum cost now this is really a finite horizon problem because you will be at the termination state within a finite number of steps but the horizon is not fixed rather it is random and the corresponding cost is finite if the expected time by which we reach the world is also finite

另一种方法是保持这个问题是有界的，不让他爆炸到无穷期的问题，alpha 等于1，如果这个问题有一个终止状态，思考一下下面这个问题，你想要去一个地方取东西，这时候就需要你最小化去这个地方的成本，这是要给有穷维的问题，因为你会经过有限步的移动后停在终止状态上，但是阶段不是固定的因为这是一个随机问题，同时由于这是一个有限期问题，所以总成本会是一个有界的实数，同时你到达重点的期望步数也是一个有限的数

so that's a major class of problems so Kasich's showed this path problems and you make me familiar with a deterministic version of this problem the deterministic shortest path problem within a graph go from a node of a graph to another destination node in the graph traversing arcs which some is whose heart length is minimum so stochastic extension it's an important model but we're not going to discuss it very much in this course because we just don't have a time

这是一个很重要的问题类型，可以从确定性的视角出发，使用与确定性最短路径问题相似的方法解决，在一个图上，你需要从一个节点经过很多弧以后到一个确定的节点，最小化经过的弧的总长度，把这个问题随机化就产生了一个重要的模型，但是由于时间关系，我们不会过多地讨论这个模型

problems involving unbounded cost per stage either the scandal or undiscounted are more difficult than these two and we are not going to cover and actually there is only now emerging theory of approximations for problems like that

每阶段平均成本没有上界与下界的问题无论截断还是没有折扣都会比上面两个问题更困难，我不会讨论这个类型的问题，事实上，这种问题的近似理论还在研究中

it's also another major infinite horizon model average cost problems for problems where there is no discounting in this term here the sum of the finite horizon cost may explode to infinity you may divide by over n here and make the average cost finite these problems are a little bit more difficult than v's but we are not going to cover them again because we don't have time they have a substantial theory associated with them

这是另一个主要的无穷维平均成本问题，这类问题没有折扣因子，有限期问题的总成本可能会没有上界，你可以把它除以n让平均成本有界，这种问题比上面的问题难一点点，有一个实质性的理论与他们相关，但是由于时间问题我还是不会过多讨论他，

these are the four major classes of infant horizon problems there are some others but these are the major ones ok

这就是四种主要的无限期问题，其实还有其他类型的无限期问题，但这四种是主要的四种

now why do we want to deal with infinite horizon in reality we never have infinite horizon life is finite you know it's always a finite number of steps

我们的生活中从来没有过无限期的问题，几乎所有问题都可以在有限的步骤内解决，那么我们为什么想要解决无限期问题呢

there is some mathematical benefit in dealing with infinite horizon problems solutions are more elegant we can restrict attention very often to stationary policies policies that do not change over time what you apply today is the same rule that you'll apply tomorrow and the day after tomorrow basically because you are facing the same tail problem the tail problem is from now to infinity it doesn't become shorter at any with any passing day as a result policies turned out to be stationary so you have to remember only one meal as opposed to a sequence of memes also there are some interesting new algorithms that arise in this infinite tourism context and very often the solution turns out to be more elegant and for the mathematicians among u there's much more interesting mathematics in this area than there is for finite horizon problems also the theory of approximations is typically developed for infinite horizon problems so that's an extra motivation for us to focus on infinite horizon

因为无限期问题在数学上很优雅，我们可以把注意力放在一个平稳策略上，平稳策略指的是不随时间改变的策略，你今天采取的策略，在明天，后天都是可以继续用的，因为你面临的是同样的子问题，他们都是从当前状态一直执行到无穷时间为止，因此你可以使用相同的策略，只需要记住控制序列中的一个控制就可以了。有一些有趣的新算法，他们在这个无限期的问题中被提出，这些算法的解在数学的角度上更优雅，在这个领域有更多有趣的数学理论吗，相比于有限期问题，很多近似理论都是为无限期问题开发出来的，这也是我们关注无限期问题的一个原因

有一些有趣的新算法, 出现在这个无限的旅游环境中, 往往解决方案是更优雅, 并为数学家们在这个领域有更有趣的数学比有对于有限的地平线问题, 近似的理论通常是为无穷地平线问题而开发的, 所以这是我们专注于无穷地平线的额外动力。

DISCOUNTED PROBLEMS/BOUNDED COST

okay so we have gone from final Horizon to infinity horizon

现在从有限期问题到无限期问题的变化已经介绍完了

and now let's focus on discounted problems with bounded cost

现在我们要关注有上界成本的折扣无限期问题

this is the most well behaved class of infinite horizon problems

这是一个类表现最好的无限期问题

we have a stationary system notice no dependence on K XK and XK plus and XK plus 1 they all live in the same space everything stationary here wk also has the same distribution for each K it's a stationary system

现在有一个不依赖于时间k的平稳系统，平稳系统指的是，如果一个系统中的每一个元素都是平稳的，这就是平稳系统，你可以看到x\_k和x\_{k+1}，他们都在同一个系统状态空间内取值，同样，w\_K在每一个时间k也都具有相同的概率分布

and the cost of a policy which is a sequence of functions is obtained like so it is the sum of the discounted cost from to infinity

一个策略的成本是一个序列函数，就像这样，它是从当前到时间区域无穷的时候的累加成本

so for each PI you plug it in here you get a random variable you average it you get a number and take the limit as n goes to infinity you get you get a number again

所以如果你采用了一个策略pi，你能够获得一个随机变量，然后取它的平均值，当时间n趋于无穷的时候，你可以得到一个新的数

if G is bounded for some capital M when you assume that G is bounded for some capital M we assume that the values of the G takes but its absolute value is bounded by M then it is easy to verify that this quantity here is a real value function is bounded

如果我们假设g的上界是M，那么我们可以假设g的绝对值的上界是M，那么可以很轻易地确定这个数是一个有上界的实函数

in fact and and and that's the nice thing about this model the boundedness of G guarantees

事实上，这个模型中的g有上界是一个很好的事情，

at all costs the course of all the policies are bounded by this expression in tract

这个课程中提到的所有策略轨迹都被这个表达式限制住了上界

M is an upper bound on G 1 over 1 minus a is the sum of a geometric progression

M是上界，1-a是几何级数累加的结果

the space is that X and u live are arbitrary only the boundedness of G is important

这个问题中状态x和控制u可以在任意形式的空间内，只有g有上界是必须的

some of you that may have mathematical interest for continuous spaces there are some that very fine mathematical points having to do with measure theory but in practice they are not so important and we're not going to deal with them an important special case is when all the spaces are finite okay

我知道你们有的人会对连续空间的数学方法感兴趣，这有一些很好的数学方法可以研究，但是实际应用中，是不是连续并不重要，我们我们解决一个问题时关心的比较重要的问题是是不是所有的空间都是有限的

the finite spaces case X takes only a finite number of values so the system is a Markov chain in a finite state Markov chain

有限状态空间X里面有有限个状态，所以这个系统是一个有限空间马尔科夫链

so finite number of states and we move randomly within the states of that chain the control takes also a finite number of values and usually such a problem is called the Markovian decision problem or MDP

状态空间有限时，我们可以看到链的状态在状态空间内随机移动，控制同样在一个有限个元素的空间内取值，这种问题通常被叫做马尔科夫决策过程，或者叫MDP

some people use MDPs the name MVP for arbitrary spaces but they tend to qualify it as much as infinite spaces and repeat

有些人研究的MDP定义在任意空间内，但是他们倾向于让MDP空间是一个无穷空间

when I talk about MVP I mean finite spaces here if you want to solve the problem algorithmically ultimately you have to work with a finite spaces approximation you have to discretize the problem have discretized space space to control space and so on so in the end it's going to be a finite space SMD field which is why this model is important and people artificial intelligence almost uniformly use finite spaces and DP as the starting point for their analysis and their algorithms okay

在我提到MDP的时候，我说的是有限空间的MDP，如果你想要最终把问题解决，你必须对有限空间近似做工作，你必须把问题离散化让状态和控制在有限空间内取值，让这个问题变成有限空间MDP，这就是为什么系统模型这么重要，人工智能领域的研究者都从有限空间MDP开始他们的研究

SHORTHAND NOTATION FOR DP MAPPINGS

so this is the problem we are going to consider first and the analysis of this problem becomes very complicated because of notation notation dynamic programming is a bit of a curse maybe you should add it to the curse of dimensionality or the curse of modeling

我们要考虑和分析这个问题由于动态规划的符号变得非常复杂，这也是动态规划的一个问题，也许你可以把他加到维数灾或者建模困难里面

the curse of notation long expressions which clutter your clutter be your vision and there is a shorthand notation which facilitates very much the B both the analysis in the insight that you get into problems

复杂的符号会对你造成影响，我们使用更简短的符号可以让解决和分析这个问题变得更简单

and it may be a little challenge because it's abstract but it's really very convenient if you want to do a short course there's no other way to do it than to resort to some kind of shorthand notation so let's introduce the shorthand notation okay

当然简单的符号会带来一点挑战因为他有一点抽象，但是他真的非常方便，如果你想做一个尖端的课程，没有比使用简短的符号更好的方案了，下面让我来介绍一下这些简短的符号

for any function of X remember we're looking at cost function any function of X

对于任何以x为变量的函数，还记得以x为变量的成本函数么

let's consider the product of the dynamic programming step starting with this function J can you recognize here the current stage cost and the cost to go alpha is the discount factor

让我们考虑一下动态规划的函数J的值，它等于当前阶段的成本加上cost-to-go乘以折扣因子alpha的值

and many more plays so TJ is another function so beauty as a box you put J N and T produces is an operator that produces another function TJ and its values are given by this expression more concretely

TJ是一个很漂亮的函数，就像一个盒子，T是一个算子，你把J\_n和T相乘，就可以得到另一个函数TJ，他的值可以通过这个具体的表达式计算获得

TJ is the optimal cost function for a one-stage problem with stage cost g and terminal cost function alpha J

TJ是最优成本函数，他的表达方式是当前阶段的成本加上cost-to-go，即折扣因子alpha乘以J的值

now for discounted problems she operates on bounded functions to produce other bounded functions okay

对于一个有上界的问题，这个算子可以产生另一个有上界的函数

and we can introduce a similar mapping for any stationary policy remember a stationary policy is one that does not depend on time

我接下来要介绍一个简单的平稳策略的映射，还记得平稳策略么，就是时间无关的策略

you plug it in here in place of u and you get another black box that generates from given a function J it generates another function team use of J by means of this formula

你用平稳函数代替控制u，你就获得了一个黑盒子，在给定J的时候，通过计算这个表达式的均值，就可以生成J

now the key point is that the details of the problem do not really matter what really matters is the structure of the map in T and T\_mu the dynamic programming mapping

我们要关心的主要问题不是这个问题的细节，而是映射的结构，就是T和T\_mu的映射，也叫做动态规划映射

all the interesting structure from a theoretical point of view in an algorithmic point of view all the interesting structure of the problem is captured by T and T mu

从理论和算法的角度来看，所有的有趣的结构都是由T和T\_mu产生的

and the entire theory of discounted problems can be developed in shorthand instead of spending like pages 15 you can do it in three slides or four slides maybe four this is important to get used to this idea of a black box operator that takes J's and produces T news of J's or PC but actually it's not so hard because this is really the operator of the dynamic programming algorithm that's about goes backwards using this operator many times

整个折扣问题都可以被简短的符号代替，如果你使用原始的定义需要15页纸来表达你的想法，使用简短的符号可能三页或者四页纸就能完成了，使用这个黑盒算子的想法可以让J和算子T产生另一个函数，这并不是很难因为这只是动态规划的一个算子，后向操作会多次使用这个算子

so TNT new provide a powerful unifying framework for dynamic programming and this is the essence of this book but listed in your list of reference to the abstract dynamic programming book that I wrote last year instead of focusing on specific problems focusing on properties of T and T new that give you the basic theory now we are going to use this abstract notation to develop the theory of infinite horizon discounted problems

T和T\_mu为动态规划提供了一个统一的强力的框架，这是一本书的精髓，也就是我在你的参考列表里列出的那本去年写的抽象动态规划，这本书没有关注具体的问题，而是关注T和T\_mu提供的基本理论和性质，现在我们要来看一看如何使用这个抽象的理论来分析无限期折扣问题

FINITE-HORIZON COST EXPRESSIONS

but let's see how you can use it for final quarter produce okay

让我们看看他是如何工作的

so let's consider an end stage policy with terminal cost J and discounted cost per stage okay

我们考虑一个带有终端成本J和每个状态的折扣成本的终端状态策略

so the cost here of this policy starting at X0 is the cost of the end stages discounted plus a terminal cause that's also discounted by alpha

这个成本是从x0开始的策略的成本，等于终端状态的成本加各阶段的折扣成本

now you can write this as okay take the first stage cost so take it out of here and then take a common factor alpha and then what you have is that this can be written as the cost of the first stage plus the cost of the remaining stages

现在你可以把它写成第一个阶段的成本加上折扣因子乘以剩余成本的形式

so using our our notation you this this is this is obtained by applying tinu to take by ones of tail

最后你可以使用我们刚才提到的简短的符号，应用T\_mu0把这个表达式写成很简短的形式

where PI 1 is the tail policy this is really expression to the principle of optimality the original problem can be obtained by from the cost of the tail problem of one stage less by applying this in new operator

pi\_1是原问题依据最优性原理的子问题表达形式，即除了最后一个阶段的控制之外的所有控制策略

and if you take this and apply successfully n times you come up with an expression of the cost of any policy that's very succinct very compact you start with J the terminal cost this is the first step of the dynamic programming algorithm the second step operates on whatever you get from the first step in going forward all the way to the initial State

如果你成功地获得了时间n的策略，很简洁紧凑地表达了任意策略的成本，你就可以从终端成本开始，向前一步使用动态规划算法利用你在第一个阶段的结果计算第二个算子，然后逐渐向前直到初始阶段

if you have a stationary policy as opposed to a time-varying policy then this expression simplifies further it can be written in this very compact form starting with J apply n times the composition of n times of this operation and get the cost of the stationary policy over n stages so this is T nu T nu square Tim youth to the cubed power to the end by the n fold convolution of T new the composition continue

如果你想要一个平稳的政策, 而不是一个时变的政策, 那么这个表达式可以进一步简化，它可以这种非常紧凑的形式, 从 J 开始应用n次算子T, 并获得这个平稳策略n 阶段的成本，T\_mu^N是N个T\_mu进行组合得到的

and similarly the optimal end stage cost function is T to the end t is the minimization operation

and you have this formula which is just the dynamic programming algorithm

相似地，最优化终端阶段成本函数是对T进行优化获得的这个表达式就是动态规划

dynamic program now it can be written in this short expression n minus 1 tail problem optimal cost is used to obtain the end stages tail problem course by applying dynamic programming mapping

动态规划现在被写成了很简短的形式，n-1子问题的最优成本被用于计算终端子问题

okay just to take you back tea involves the minimization team you involves just a poly senior

回到上面的定义，T是最小化T\_mu的结果

okay so this map is give you a succinct expression a succinct form of the finite correlation theory

所以这个映射是给你一个简洁的有限关联理论

“SHORTHAND” THEORY – A SUMMARY

and now we are going to use it to express the entire theory of infinite horizon discounted problems in one slide

现在我们要使用这种简洁的表达形式用一个slide来描述一下整个无限期折扣问题问题的理论

we're talking about the infinite horizon problem for any policy PI the the cost of that policy starting from state X is obtained by taking the limit of the finite horizon cost which is given by this expression and for a stationary policy you have even something it's even simpler and j now is zero okay the terminal cost is zero for the event horizon problem we don't have

我们太谈论无限期问题的时候，对于任何从状态x开始的策略pi的成本都可以通过被这个表达式，即折扣因子限制成有限期问题的表达式计算得到，如果你希望得到一个平稳策略，这个表达式会更简洁，J\_0表示终端成本，我们把这个终端成本定义为0

now the optimal cost function satisfies a fundamental equation for infinite correlation problems in this fundamental equation is precisely this J star is the optimal cost and it is a fixed point of this mapping T okay in other words if you plug in J star within T you get back J star moreover for every policy mu again you have balanced equation J nu is the solution of this equation in fact turns out to be the unique solution of this equation

现在, 最优成本函数满足了无限期问题的一个基本方程,这个 J\* 是最优成本, 它是这个映射的一个不动点，换句话说, 如果你把乘子T作用于J\*，他的值还是J\*，此外, 对于每一个策略mu，都有一个平衡方程，其中J\_mu是这个方程的解，事实上，J\_mu也是这个方程的唯一解。

another result has to do with the optimality condition a policy a stationary policy is optimal if and only if it attains the minimum on this bellman equation

如果一个平稳策略是最优的，只有它满足bellman方程的时候才成立

the seer may have a better slide okay

这有一个更好的slide

okay there's an algorithm called value iteration value iteration is simply the dynamic programming algorithm applied an infinite number of times starting with heavy J compute apply dynamic programming once twice many times and take HK to go to infinity and then you get the optimal cross function another way to say this is that the limit of the optimal finite horizon cost is the optimal infinite horizon cost

有一个简单地动态规划算法的应用叫做值迭代算法, 从J开始应用动态规划k次，在k的值趋于无穷的时候，你就可以得到最优函数了，另一种说法是，最优有限期问题的最优解等于最优无限期问题的最优解

and the other major algorithm is the so called policy duration algorithm given a stationary policy new K find the cost of that policy by solving this equation then you do a policy improvement five new J plus 1 such that new k plus 1 attains the minimum in the minimization operation involved here and so from UK you get new K plus 1 mu K plus 1 and so on and it's a basic algorithm that has the property that it converges to the optimal policy

另一个主要的算法被叫做策略迭代，给定一个平稳策略mu\_k，通过求解第一个方程计算这个策略的成本，然后改善这个策略，改善策略是这样进行的，最小化等号右边的表达式，然后找到一个新的策略mu\_{k+1}，一直这样持续下去直到收敛，这是一个基本算法，他有收敛性保证能够收敛到最优策略

these are the two major results I haven't proved any of those so far but I'm I want to tell you that with shorthand notation you have it all in one page

这是两个主要的结果，到目前为止我还没有证明他们，但是我想告诉你的是，使用简洁的符号，你可以用一页纸描述他们

break Q&A

okay so maybe we can take a little break now and then I'm going to go into into how we address these results and what we get what we get from the theory are they questions is there any patient for the polish duration algorithm under what conditions it is valid is like a question the initial policy can be arbitrary any policy will be sufficient within this discounted problem context any policy yes and you get convergence in a finite number of steps here of course we remember discounted problems finite spaces discard the problems that's what I'm talking about for finite space is discounted problems it's a finite number of states fine number of controls to find a number of policies stationary policies so you terminate in a finite number of steps yes okay what happens well those are all outfit into this it turns out that the rollout is simply one iteration of this algorithm you start with a base for this some heuristic and through this policy evaluation and improvement you get a rollout policy okay so your question has to do with error bounds unfortunately there is very little in the way of error bounds all you really have is that the base policy the rollout policy is better okay is upper bounded by the cost of the of the base policy and it's very difficult except in some special cases people have been this recent research in specific types of problems like knapsack type problems or other problems of you know specialized problems where people have been able to obtain bounds for the performance of rolla but in general there is no result it's the claim to success is based entirely on experimental evidence here one predicted control well okay I mentioned earlier the the result that the rollout policy is better than the base policy in model predictive control this results translates the following if you start with a stable policy you're going to get by NPC you're going to get hey also another stable policy and that's that's not very much but but but still is important because it provides a guarantee of stability of the what you get Randy's this of course under some assumptions people have been using MPC in other contexts but there are no guarantees at all yeah if you do multiple rollout and you'll be doing this policy direction but all out is just one step of this but it's not so easy to do this by simulation at least in the traction that people use rollout no two states are allowed is computationally very intensive yes yes if you could do it but unfortunately this policy evaluation has to be done efficiently for this algorithm to to be viable and in rollout the policy evaluation is very time-consuming because you do it by simulation Monte Carlo simulation so you have to - to roll Monte Carlo simulation within Monte Carlo simulation and pretty soon it becomes not viable but if you were to do it theoretically would lead to the object I wouldn't say that value iteration does not use policies okay it applies a minimization operation does not use any policy evaluation nor does it compute intermediate policies yes okay let me take you back to to this slide and roll out okay in rollout we have some base policy some heuristic and we simulate it in order to calculate this expression here so each time we need to calculate this expression for a given value of FK and so on for given value of next state from that state we simulate the the the base policy many many times and then we take the expected value of that that's that it's very simple just you have a random number generator calculate a lot of Monte Carlo trajectories starting from this and you get a number that's the number that you plug in to this algorithm but you have to do it for every possible value of U okay for every possible here for every possible value of U and so that's a lot of computation ok let's take a break because we're running a little late and I'm going to get into I want you to take if you could use the break to eyeball these two expressions here and because they are going to figure importantly in the in the next few slides seven minutes please okay [Music]