# Using the Foundations and Trends® LaTeX Class

## Instructions for Creating an FnT Article

**Suggested Citation:** Alet Heezemans and Mike Casey (2018), "Using the Foundations and Trends® LaTeX Class", : Vol. xx, No. xx, pp 1–18. DOI: 10.1561/XXXXXXXXX.

**Alet Heezemans**
now publishers, Inc.
alet.heezemans@nowpublishers.com

**Mike Casey**
now publishers, Inc.
mike.casey@nowpublishers.com

now
the essence of knowledge
Boston — Delft

# Contents

# Using the Foundations and Trends® LaTeX Class

Alet Heezemans[1] and Mike Casey[2]

[1]*now publishers, Inc.; alet.heezemans@nowpublishers.com*
[2]*now publishers, Inc.; mike.casey@nowpublishers.com*

ABSTRACT

This document describes how to prepare a Foundations and Trends® article in LaTeX . The accompanying LaTeX source file FnTarticle.tex (that produces this output) is an example of such a file.

Alet Heezemans and Mike Casey (2018), "Using the Foundations and Trends® LaTeX Class", : Vol. xx, No. xx, pp 1–18. DOI: 10.1561/XXXXXXXXX.

# 1

# Introduction to Dynamic Programming

## 1.1 Dynamic Programming Setting

Consider a *generic* optimization problem

$$\min_{u \in U} g(u)$$

where $u$ is the decision variable, $g(u)$ is the cost function, and $U$ is the constraint set. There are several categories of problems:

- Discrete (i.e., $U$ is finite), or continuous

- Linear (i.e., $g$ is lienar and $U$ is polyhedral) or nonlinear

- Stochastic or deterministic: For stochastic problems, the cost involves a stochastic parameter $\omega$, which is averaged:

  $$g(u) = \mathbb{E}_{\omega}\{G(u, w)\}$$

  where $\omega$ is a random parameter.

The dynamic programming (dp) is able to deal with complex stochastic problems where $\omega$ becomes available in stages, and the decisions are also made in stages based on the information.

### 1.1.1 Basic Structure of Stochastic DP

The DP usually invoves the following elements:

- Discrete-time system:

$$x_{k+1} = f_k(x_k, u_k, \omega_k), \quad k = 0, 1, \dots, N-1$$

  where

  - $k$ denotes the stage / discrete time
  - $x_k$ denotes the state for stage $k$, which *summarizes past information* that is relevant for future decision
  - $u_k$ denotes the control variable, i.e., a decision to be selected at time $k$ from a given set. When making the decision, the state $x_k$ is known.
  - $w_k$ denotes the *random parameter* / *disturbance* / *noise* at stage $k$.
  - $N$ is the *Horizon* or the number of times that the control is applied.

- A cost function that is *addictive* over time:

$$\mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, \omega_k) \right\}$$

**Remark 1.1.** We may use the transition matrix as the alternative *discrete-system* description, i.e., suppose the transition matrix $P(x_{k+1} \mid x_k, u_k)$ is available, then

$$x_{k+1} = \omega_k, \text{ with } P(\omega_k \mid x_k, u_k) = P(x_{k+1} \mid x_k, u_k)$$

Several assumptions are made for common dp problem:

- The set of possible values for control $u_k$ depend at most on $x_k$ but not on *prior* $x$ or $u$

- The probability distribution of $\omega_k$ does not depend on past values $\omega_{k-1}, \dots, \omega_0$, i.e., $\omega_k \sim P(\cdot; x_k, u_k)$, but may depend on $x_k$ and $u_k$, since otherwise past values of $\omega$ would be useful for future optimization by applying *state augmentation.*

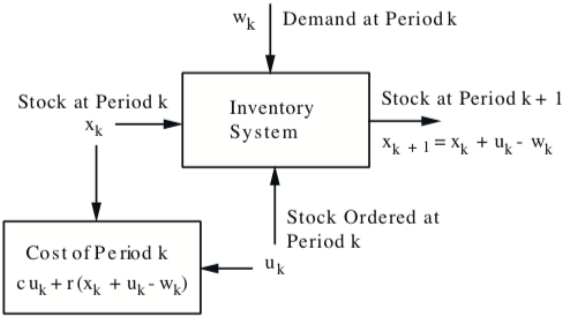

**Figure 1.1:** Inventory Model

### 1.1.2 Inventory Control Example

The inventory model is summarized in the figure below: In this case, the discrete-time system is

$$x_{k+1} = f_k(x_k, u_k, \omega_k) = x_k + u_k - \omega_k$$

with the cost function that is*additive* over time:

$$\mathbb{E}\left\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, \omega_k)\right\} = \mathbb{E}\left\{\sum_{k=0}^{N-1}(c \cdot u_k + r \cdot (x_k + u_k - \omega_k))\right\}$$

Our goal is to optimize the cost function over *policies*, and the policies $u_k = \mu_k(x_k)$ maps states into controls. The discrete-time system for this problem is deterministic, but the cost function is a expected value for a stochastic rv.

### 1.1.3 Deterministic Finite-State Problem

Consider a deterministic dp problem. We aim to find the optimal sequence of four operations $A, B, C, D$, where $A$ must *precede* $B$ and $C$ must *precede* $D$. Assume the startup cost to be $S_A, S_C$, and the setup transition cost from operation $m$ to operation $n$ is $C_{mn}$.

We may list all possible scheduling in the figure below:

The discrete-time system for this problem is deterministic, and the cost function is deterministic as well.

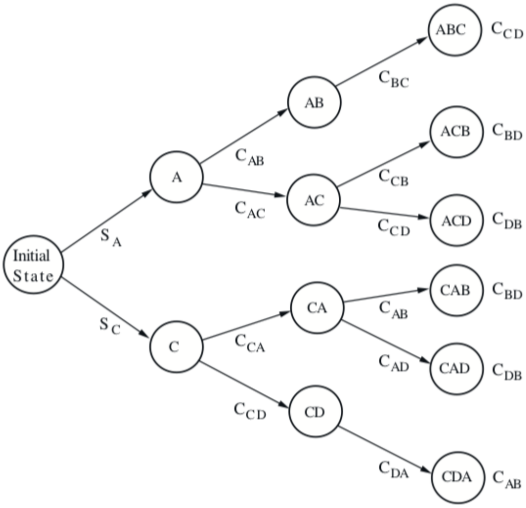

**Figure 1.2:** Scheduling Problem

### 1.1.4 Stochastic Finite-State Problem

The goal is to find the two-game chess match strategy.

- The *timid* play draws with probability $p_d > 0$ and loses with probability $1 - p_d$

- The *bold* play wins with probability $p_w < \frac{1}{2}$ and loses with probability $1 - p_w$.

We may list all possible strategies result in the figure (1.3). The discrete-time system for this problem is stochastic, and the cost function is stochastic as well.

### 1.1.5 Formal Setting of DP

The formal setting of DP is as follows:

- The system transition satisfies

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, 1, \ldots, N-1$$

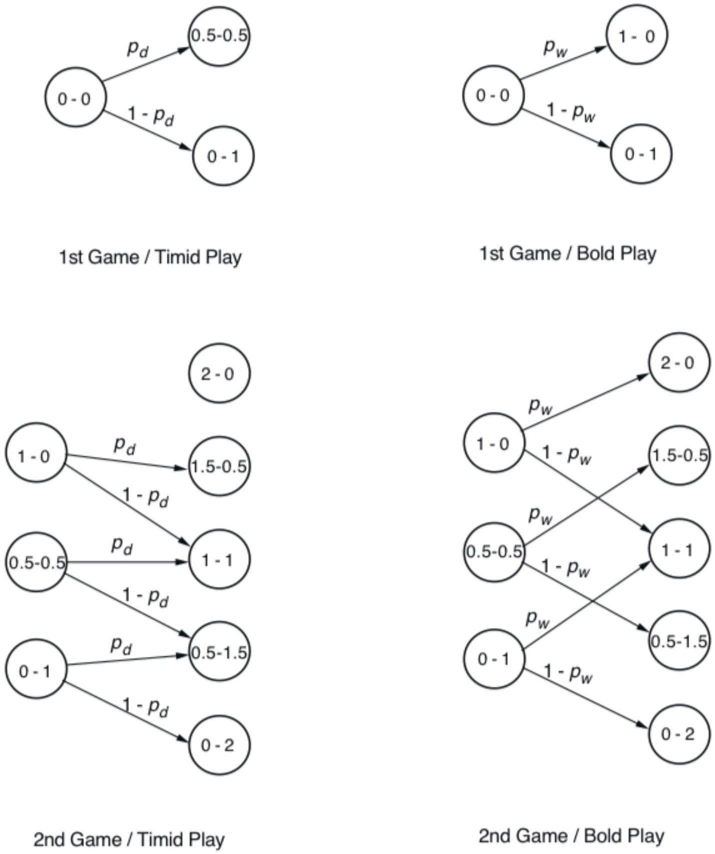

**Figure 1.3:** Two-Game Chess Match Strategies

- The control constraints only depends on $x_k$, i.e., $u_k \in U_k(x_k)$
- The probability distribution of $\omega_k$ is $P_k(\cdot \mid x_k, u_k)$
- The policy is a collection of functions $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, where $\mu_k$ maps states $x_k$ into controls $u_k = \mu_k(x_k) \in U_k(x_k)$ for all $x_k$
- The expected cost of $\pi$ starting at $x_0$ is given by

  $$J_\pi(x_0) = \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), \omega_k) \right\}$$

  The goal is to minimize the cost function:

  $$J^*(x_0) = \min_\pi J_\pi(x_0),$$

  where the optimal policy $\pi^*$ satisfies

  $$J_{\pi^*}(x_0) = J^*(x_0).$$

  Note that $\pi^*$ is independent of $x_0$.

### 1.1.6 Significance of Feedback

The closed-loop policies refer to that the controller can adapt to the unexpected values of the state, while the open-loop policies cannot get access to this information.

- For deterministic problems, the *open-loop policies* is as good as the *closed-loop policies*
- For stochastic problems, the reduction in the cost gained by closed-loop is called the *value of information.*

For example. consider the chess match problem,

1. The probability of win using optimal open-loop policy is

   $$p_w^2 + p_w(1-p_w) \cdot \max(2p_w, p_d)$$

2. The probability of win using optimal closed-loop policy is

   $$p_w^2(2-p_2) + p_w(1-p_w) \cdot p_d$$

3. Therefore, we see that the value of information is sometimes positive:

   $$\text{Value of Information } = p_w(1-p_w) \min(p_w, p_d - p_w)$$

**Variants of DP** In the second-half of this couse, we may consider several variants of DP problems:

- Continuous-time problems
- Imperfect state information problems
- Infinite horizon problems
- Suboptimal Control

## 1.2 Dynamic Programming Solving

### 1.2.1 Principle of Optimality

Suppose $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ be optimal policy. Consider the *tail subproblem*, i.e., we are at $x_i$ at time $i$ and wish to minimize the *cost-to-go* from time $i$ to time $N$:

$$\mathbb{E}\left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), \omega_k) \right\}$$

Consider the *tail policy* $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$.

**Theorem 1.1** (Principle of Optimality)**.** The tail policy is optimal for the tail subproblem, i.e., the optimization of the future does not depend on what we did in the past.

**Remark 1.2.** Consider an auto travel analogy. Suppose that the fastest route from A to C passes the place B. The principle of optimality highlights that *the B to C portion of the route is also the fastest route of the trip that starts from B and ends at C*.

The DP aims to solve all tail subproblems of $N$ horizons. The DP algorithm is based on the idea as follows: It proceeds sequentially, by solving all the tail subproblems of a given time length, based on the solution of the tail subproblems of shorter time length.

### 1.2.2 DP algorithm for scheduling problem

Consider the scheduling problem discussed in subsection (1.1.3). The startup cost and transition cost is summarized in the figure (1.4).

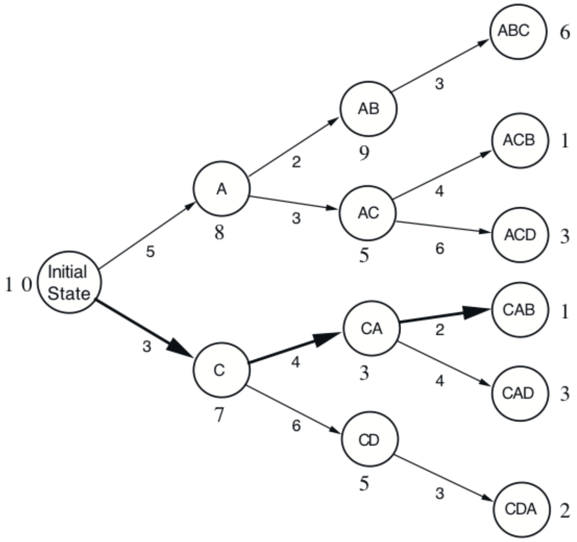

**Figure 1.4:** Scheduling Problem Cost Description

**Tail Problem of Length 1:** The subproblems involving one operations are trival to solve. State ABC,ACB,ACD,CAB,CAD,CDA correspond to the tail cost $6, 1, 3, 1, 3, 2$.

**Tail Problem of Length 2:** Solving the subproblems involving two operations is based on the tail problem of length 1. State $AB, AC, CA, CD$ corresponds to the tail cost $9, \min(4{+}1, 6{+}3) = 5, \min(2{+}1, 4{+}3) = 3, 5$.

**Tail Problem of Length 3:** Solving the subproblems involving three operations is based on the tail problem of length 2. State $A, C$ corresponds to the tail cost $\min(2 + 9, 3 + 5) = 8, \min(4 + 3, 6 + 5) = 7$.

**Tail Problem of Length 4:** Solving the subproblems involving four operations is based on the tail problem of length 3. The initial state corresponds to the tail cost $\min(5 + 8, 3 + 7) = 10$.

After computing the optimal cost, we construct the optimal solution by starting at the initial node and proceeding forward, where each time we choose the operation that starts the *optimal* schedual for the corresponding *tail subproblem*. Therefore, at each state-time pair, we need to record both the optimal cost-to-go and the optimal decision.

### 1.2.3 Stochastic Inventory Example

Consider the inventory problem in subsection (1.1.2). The tail subproblems of length 1 aims to solve

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1}\geq 0} \mathbb{E}_{\omega_{N-1}}\{c\cdot u_{N-1}+r\cdot(x_{N-1}+u_{N-1}-\omega_{N-1})\},\ \forall x_{N-1}$$

The tail subproblems of length $N-k$ aims to solve

$$J_k(x_k) = \min_{u_k\geq 0} \mathbb{E}_{\omega_k}\{c\cdot u_k + r\cdot(x_k+u_k-\omega_k) + J_{k+1}(x_k+u_k-\omega_k)\},\ \forall x_k$$

Here $J_0(x_0)$ is the optimal cost with initial state $x_0$.

### 1.2.4 Formal Dynamic Programming Algorithm

The dynamic programming aims to find the policy by starting with

$$J_N(x_N) = g_N(x_N),$$

and go backwards in time from period $N-1$ to period 0:

$$J_k(x_k) = \min_{u_k\in U_k(x_k)} \mathbb{E}_{\omega_k}\left\{g_k(x_k,u_k,\omega_k) + J_{k+1}(f_k(x_k,u_k,\omega_k))\right\},\ k=0,\dots,N-1. \tag{1.1}$$

In such case, $J_0(x_0)$ is generated at the last step, and is equal to the optimal cost $J^*(x_0)$. Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes the RHS of (1.1) for each $x_k$ and $k$, then $\pi^* = \{\mu_0^*,\dots,\mu_{N-1}^*\}$ is optimal.

*Proof.* Proof by induction that $J_k(x_k)$ is equal to $J_k^*(x_k)$, which is defined as the optimal cost of the tail subproblem that starts tat time $k$ at state $x_k$. □

### 1.2.5 Linear-Quadratic Analytic Example

Consider a certain material is passed through a sequence of two ovens shown in Figure (1.5).

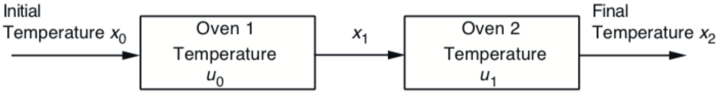

**Figure 1.5:** Linear-Quadratic Example

- The system is given by

$$x_{k+1} = (1-a)x_k + au_k, \quad k = 0, 1,$$

  where $a$ is a given scalar from the interval $(0, 1)$.

- Define the cost function

$$r(x_2 - T)^2 + u_0^2 + u_1^2$$

  where $r$ is a given positive scalar.

- We may apply the dp algorithm as follows:

$$\begin{aligned} J_2(x_2) &= r(x_2 - T)^2 \\ J_1(x_1) &= \min_{u_1} \left[ u_1^2 + r((1-a)x_1 + au_1 - T)^2 \right] \\ J_0(x_0) &= \min_{u_0} \left[ u_0^2 + J_1((1-a)x_0 + au_0) \right] \end{aligned}$$

### 1.2.6 State Augmentation

When the assumptions of the basic problem are violated, then we need to consider reformulating the state. The DP algorithm still applies, but the problem gets bigger.

**Example** Consider the system becomes

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, \omega_k)$$

Introducing the additional state variable $y_k = x_{k-1}$ and view $\tilde{x_k} = (x_k, y_k)$. Therefore, the new system takes the form

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, \omega_k) \\ x_k \end{pmatrix}$$

The DP algorithm applies to the reformulated problem:

$$J_k(x_k, x_{k-1}) = \min_{u_k \in U_k(x_k)} \mathbb{E}_{\omega_k} \left\{ g_k(x_k, u_k, \omega_k) + J_{k+1}(f_k(x_k, x_{k-1}, u_k, \omega_k), x_k) \right\}$$

# 2

# Dynamic Programming for Deterministic System

## 2.1 Deterministic Finite-State Problem

Consider a shortest path problem shown in Figure (2.1). We can convert the problem into a dynamic programming system as follows:

- The states correspond to the nodes in each stage.
- The controls correspond to the acrs in each stage.
- The control sequences for this open-loop system correspond to

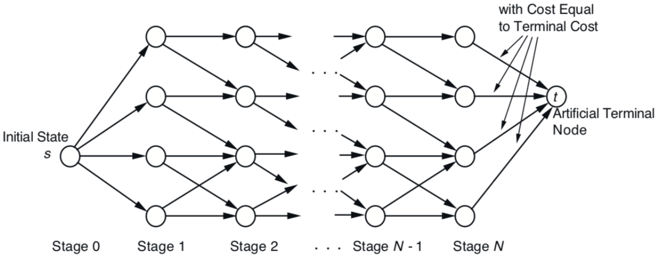

**Figure 2.1:** Illustration of shortest path problem

paths from initial state to the terminal states.

- Let $a_{ij}^k$ denote the cost of transition (length of the arc) from the state $i \in S_k$ to state $j \in S_{k+1}$ at time $k$
- The cost of the control sequence is the cost of the corresponding path (total length of the path)

**DP algorithms** The backward algorithm applies as follows:

$$\begin{aligned} J_N(i) &= a_{ii}^N, \quad i \in S_N \\ J_k(i) &= \min_{j \in S_{k+1}} [a_{ij}^k + J_{k+1}(j)], \quad i \in S_k, k = N-1, N-2, \ldots, 0. \end{aligned}$$

Here the length of the shortest path from $s$ to $t$ equals to the $J_0(s)$.

Note that the optimal path $s \to t$ is also an optimal path $t \to s$ for a *reversed* shortest path problem, where the direction of each arc is reversed with its length unchanged.

Therefore, we can apply the backward DP algorithm for the reversed problem, which is so called the *forward* DP algorithm:

$$\begin{aligned} \tilde{J}_N(j) &= a_{sj}^0, \quad j \in S_1, \\ \tilde{J}_k(j) &= \min_{i \in S_{N-k}} \left[ a_{ij}^{N-k} + \tilde{J}_{k+1}(i) \right], \quad j \in S_{N-k+1} \end{aligned}$$

The optimal cost for the whole problem is $\tilde{J}_0(t) = \min_{i \in S_N} \left[ a_{it}^N + \tilde{J}_1(i) \right]$. The interpretation for this forward DP algorithm is that we can view $\tilde{J}_k(j)$ as the *optimal cost-to-arrive* to state $j$ from initial state $s$.

**Remark 2.1.** There is no *forward* DP algorithm for *stochastic problems*. This is becasue that we cannot restrict ourselves to open-loop sequences for the stochastic problems. In other words, due to the uncertainty, the concept of *optimal-cost-to-arrive* at the state $x_k$ does not make sense, i.e., it's impossible to guarantee (with probability 1) that any given state can be reached.

Fortunately, even for stochastic problems, the concept of *optimal cost-to-go* from any state $x_k$ makes clear sense.

## 2.2 Generic Stochastic Path Problems

Consider first a general deterministic shortest path(***s***) problems as follows:

- Let $\{1, \ldots, N, t\}$ denote the nodes of a graph, where $t$ is the destination.

- The $a_{ij}$ denotes the cost of moving from node $i$ to node $j$

- Our goal is to find a shortest (minimum cost) path from each node $i$ to node $t$

- Assumption: *All cycles have non-negative* length, i.e., the optimal path *need not take more than* $N$ *moves.*

- We can reformulate this problem as one where we require *exactly* $N$ moves by allowing *degerate moves* from a node $i$ to itself with cost $a_{ii} = 0$. Moreover, we let

  $$\begin{aligned} J_k(i) &= \textit{optimal cost of getting from } i \textit{ to } t \textit{ in } N-k \textit{ moves} \\ J_0(i) &= \textit{cost of the optimal path from } i \textit{ to } t \end{aligned}$$

- Therefore, we apply the DP algorithm as follows:

  $$J_k(i) = \min\nolimits_{j=1,\ldots,N} \left[ a_{ij} + J_{k+1}(j) \right], \quad k = 0, 1, \ldots, N-2,$$

  with $J_{N-1}(i) = a_{it}, i = 1, \ldots, N$.

For example, consider the shortest paths problem shown in Fig (2.2). Assume the destination node is the node 5. The state $i$ means the current position is in the node $i$. In such case, note that

$$\begin{aligned} J_{N-1}(i) &= a_{it}, \quad i = 1, \ldots, N \\ J_k(i) &= \min_{j=1,\ldots,N} \left[ a_{ij} + J_{k+1}(j) \right], \quad k = 0, 1, \ldots, N-2 \end{aligned}$$

The solution for this problem is shown in Fig. (2.2b), where the arrow indicate the optimal move to node 5 from the position at current stage.

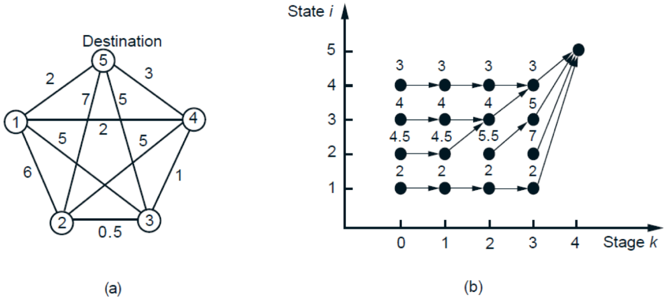

**Figure 2.2:** One Example of shortest paths problems

### 2.2.1 HMM setting and solving

Now consider the *Hidden Markov Models*. Suppose that the Markove chain is with transition probability $p_{ij}$, but the state transitions are hidden from our view. For each transition, we obtain an *observation* instead. Let $r(z; i, j)$ denote the probability of observation taking value $z$ given the transition $i \to j$. The *trajectory estimation problem* is that, given the observation sequence $Z_N = \{z_1, \ldots, z_N\}$, what's the *most likely* state transition sequence $\hat{X}_N := \{\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N\}$, i.e., the one equals to

$$\arg \min_{\text{all } X_N} p(X_N \mid Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}$$

We can rewrite the probability $p(X_N, Z_N)$ as follows:

$$\begin{aligned} p(X_N, Z_N) &:= p(x_{0:N}, z_{1:N}) \\ &= \pi_{x_0} \cdot p(x_{1:N}, z_{1:N} \mid x_0) \\ &= \pi_{x_0} p(x_1, z_1 \mid x_0) p(x_{2:N}, z_{2:N} \mid x_{0:1}, z_1) \\ &= \pi_{x_0} p_{x_0 \to x_1} r(z_1; x_0, x_1) p(x_{2:N}, z_{2:N} \mid x_{0:1}, z_1) \end{aligned}$$

Following the similar idea, we compute

$$\begin{aligned} p(x_{2:N}, z_{2:N} \mid x_{0:1}, z_1) &= p(x_2, z_2 \mid x_{0:1}, z_1) p(x_{3:N}, z_{3:N} \mid x_{0:2}, z_{1:2}) \\ &= p_{x_1 \to x_2} r(z_2; x_1, x_2) p(x_{3:N}, z_{3:N} \mid x_{0:2}, z_{1:2}) \end{aligned}$$

Therefore, we derive

$$p(X_N, Z_N) = \pi_{x_0} \prod_{k=1}^{N} p_{x_{k-1} \to x_k} r(z_k; x_{k-1}, x_k)$$

Thus the problem is equivalent to

$$\text{minimize} \quad -\ln(\pi_{x_0}) - \textstyle\sum_{k=1}^{N} \ln \left(p_{x_{k-1}\to x_k} r(z_k; x_{k-1}, x_k)\right)$$

over all possible sequences $\{x_0, x_1, \ldots, x_N\}$. Therefore, due to the structure of our objective function, we can apply a distributed algorithm (e.g., dynamic programming) to solve it.

## 2.3 General Shortest Path Algorithms

Next, we discuss some non-DP shortest path algorithms. Note that they all can *only* be used to solve deterministic finite-state problems. The advantage is that they avoid computing the optimal cost-to-go for *every* state. This is essential for problems with *huge* state spaces. (see examples in combinatorial optimization)

### 2.3.1 Label Correcting Methods

The idea of this kind of algorithm is to *progressively* discover shorter paths from the origin to *any other* node $i$, and to maintain the length of the shortest path found *so far* in a variable $d_i$ (called the *label* of $i$).

- If a shorter path to $i$ is found, then $d_i$ is reduced, and the algorithm checks if the labels $d_j$ can be reduced, where $j$ is the children of $i$, i.e., whether labels $d_j$'s can be reduced by setting them into $d_i + a_{ij}$.

- This algorithm makes use of a list of nodes called *OPEN* (or *candidate list*). This list contains nodes that are currently active such that they are candidates for further examination by the algorithm, and possible inclusion in the shortest path.

  Initially, OPEN contains only $s$. Each node that has entered the OPEN at least once (except $s$) is assigned a *parent* (refers to some node). The parent nodes are not necessary for the computation of shortest distance, but for tracing the shortest path to the origin after the algorithm terminates.

## Notations

- $d_i$ denotes the *label* of $i$, i.e., the length of the shortest path from $s$ to $i$ found so far. Initially $d_s = 0, d_i = \infty, \forall i \neq s$
- UPPER: The label $d_t$ to the destination
- OPEN list: contains nodes that are currently active in the sense that they are candidates for further examination (initially OPEN $= \{s\}$)

The label correcting methods work as follows:

**Algorithm 1** The Label Correcting Methods

```
1: (Node Removal): Remove a node $i$ from OPEN and for each child $j$ of $i$, do step 2
2: (Node Insertion Test): If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_j + a_{ij}$, and set $i$ to be the parent of $j$. In additiion, if $j \neq t$, place $j$ into the OPEN if $j \notin$ OPEN; otherwise set UPPER to the new value $d_i + a_{it}$ of $d_t$
3: (Termination Test): If OPEN is empty, terminate, else go to step 1.
```

Here we show how to apply the label correcting algorithm into the travelling salesman problem shown below.

| Iteation Number | Node Exiting OPEN | OPEN after Iteration | Upper |
|---|---|---|---|
| 0 | – | 1 | $\infty$ |
| 1 | 1 | $2, 7, 10$ | |
| 2 | 2 | $3, 5, 7, 10$ | $\infty$ |
| 3 | 3 | $4, 5, 7, 10$ | $\infty$ |
| 4 | 4 | $5, 7, 10$ | 43 |
| 5 | 5 | $6, 7, 10$ | 43 |
| 6 | 6 | $7, 10$ | 13 |
| 7 | 7 | $8, 10$ | 13 |
| 8 | 8 | $9, 10$ | 13 |
| 9 | 9 | 10 | 13 |
| 10 | 10 | $\emptyset$ | 13 |

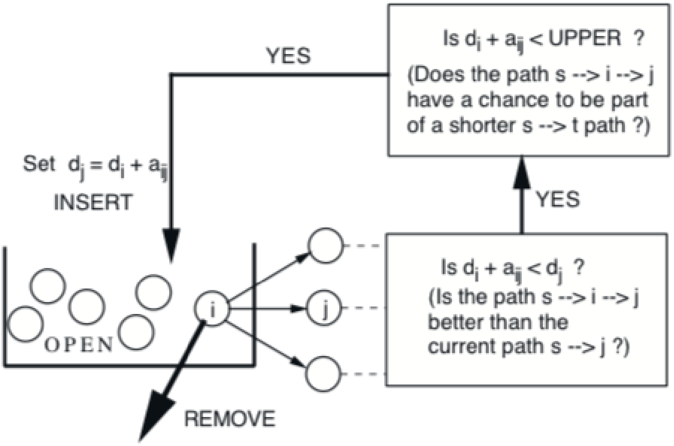
YES
Is $d_i + a_{ij} <$ UPPER ?
(Does the path s --> i --> j have a chance to be part of a shorter s --> t path ?)
Set $d_j = d_i + a_{ij}$
INSERT
YES
OPEN
i
j
Is $d_i + a_{ij} < d_j$ ?
(Is the path s --> i --> j better than the current path s --> j ?)
REMOVE

**Figure 2.3:** Diagram Illustration of Label Correcting Methods

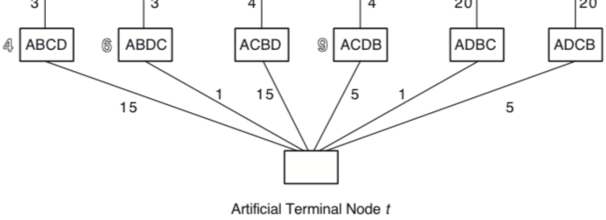

**Figure 2.4:** One example for travelling salesman problem

Note that some nodes never entered OPEN.

Finally, we give a verification for why label correcting methods work.

**Proposition 2.1.** If there exists at least one path from the origin to the destination, the label correcting algorithm terminates with UPPER equal to the shortest distance from the origin to the destination.

*Proof.*

1. Each time a node $j$ enters OPEN, its label is decreased and becomes equal to the length of some path from $i$ to $j$

2. The number of possible distinct path lengths is finite, and therefore the number of timesa node can enter is finite, and the algorithm terminates.

3. Let $\{(s, j_1, \ldots, j_k, t), d^*\}$ be the pair of shortest path and the corresponding shortest distance. If UPPER $> d^*$ at the termination, UPPER will also be larger than the length of all parths $(s, j_1, \ldots, j_m), , = 1, \ldots, k$ throguhout the algorithm, i.e., the node $j_k$ will never enter the OPEN list with $d_{j_k}$ equal to the shortest distance from $s$ to $j_k$.

   Similarly, node $j_{k-1}$ will never enter to the OPEN list with $d_{j_{k-1}}$ equal to the shortest distance from $s$ to $j_{k-1}$. Continue to $j_1$ by induction will obtain a contradiction.

□

# 3

# Dynamic Programming with perfect information

## 3.1 Linear-Quadratic Problems

Now consider the case where the future observations is linear w.r.t. current observation and controller:

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

The goal is to minimize the quadratic cost

$$\mathbb{E}_{\omega_k, \forall k} \left\{ x_N^{\mathrm{T}} Q_N x_N + \sum_{k=0}^{N-1} (x_k^{\mathrm{T}} Q_k x_k + u_k^{\mathrm{T}} R_k u_k) \right\}$$

Assumptions:

1. $Q_k \succeq 0$ and $R_k \succ 0$;

2. the noise $\omega_k$ are *independent* with *zero mean.*

We may apply the DP algorithm to solve it:

$$\begin{aligned} J_N(x_N) &= x_N^{\mathrm{T}} Q_N x_N \\ J_k(x_k) &= \min_{u_k} \{ x_k^{\mathrm{T}} Q_k x_k + u_k^{\mathrm{T}} R_k u_k + J_{k+1}(A_k x_k + B_k u_k + \omega_k) \} \end{aligned}$$

We *claim* several key facts:

**Theorem 3.1.** 1. $J_k(x_k)$ is *quadratic*

2. Optimal policy $\{\mu_0^*,\dots,\mu_{N-1}^*\}$ is *linear*, i.e.,

$$\mu_k^*(x_k)=L_kx_k,\quad k=0,\dots,N-1$$

Therefore, we may apply the similar treatment to solve for $J_k(x_k)$ with $k=N-1,N-2,\dots,1$.

*Proof.* By induction it is easy to verify

$$\mu_k^*(x_k)=L_kx_k,\quad J_k(x_k)=x_k^{\mathrm{T}}K_kx_k+\text{constant}$$

where

$$\begin{aligned}
L_k&=-(B_k^{\mathrm{T}}K_{k+1}B_k+R_k)^{-1}B_k^{\mathrm{T}}K_{k+1}A_k,\ k=0,\dots,N-1\\
K_N&=Q_N\\
K_k&=A_k^{\mathrm{T}}(K_{k+1}-K_{k+1}B_k(B_k^{\mathrm{T}}K_{k+1}B_k+R_k)^{-1}B_k^{\mathrm{T}}K_{k+1})A_k+Q_k
\end{aligned}$$

The equations above are called the *discrete-time Riccatic equation.*

Similar as DP, it starts at the terminal time $N$ and proceeds backwards. □

### Asymptotic Behavior

**Theorem 3.2.** Consider the *Riccatic equation*

$$K_{k+1}=A^{\mathrm{T}}(K_k-K_kB(B^{\mathrm{T}}K_kB+R)^{-1}B^{\mathrm{T}}K_k)A+Q,\quad k=0,1,2,\dots$$

with the initial $K_0$ be arbitrary semidefinite metrix. Suppose that $(A,B)$ is controllable, and $(A,C)$ is observable. As a result:

1. The Riccati equation converges:

$$\lim_{k\to\infty}K_k=K,$$

where $K$ is the unique solution of the *algebraic Riccatic equation*

$$K=A^{\mathrm{T}}(K-KB(B^{\mathrm{T}}KB+R)^{-1}B^{\mathrm{T}}K)A+Q$$

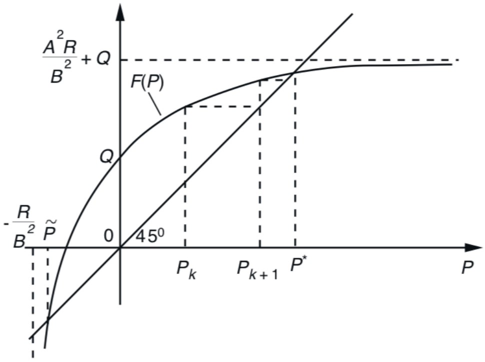

**Figure 3.1:** Graphical Proof For Scalar Systems

2. The corresponding steady-state controller

$$\mu^*(x) = Lx, \quad \text{where } L = -(B^{\mathrm{T}}KB + R)^{-1}B^{\mathrm{T}}KA$$

is stable, i.e., the matrix $(A + BL)$ for the closed loop system

$$x_{k+1} = (A + BL)x_k + \omega_k$$

satisfies $\lim_{k\to\infty}(A + BL)^k = 0$.

*Proof for scalar systems.* Consider the scalar stationary system with $A \neq 0, B \neq 0, Q > 0, R > 0$. The Riccatic equation becomes

$$P_{k+1} = A^2\left(P_k - \frac{B^2P_k^2}{B^2P_k + R}\right) + Q$$

Suppose that $P_{k+1} = F(P_k)$, where

$$F(P) = \frac{A^2RP}{B^2P + R} + Q$$

In Fig (3.1) we find that $F(P)$ has two fixed points $P^*$ and $\tilde{P}$. Note that $F(P)$ is increasing on $(-R/B^2, \infty)$. Therefore, the Riccati iteration coverges to $P^*$ starting everywhere in the inerval $(\tilde{P}, \infty)$ as shown in the figure. □

**Random System Matrices** Suppose that $\{A_0, B_0\}, \dots, \{A_{N-1}, B_{N-1}\}$ are not known but *independent random matrices* that are also independent of the $\omega_k$. The DP algorithm is given by:

$$J_N(x_N) = x_N^{\mathrm{T}} Q_N x_N$$

$$J_k(x_k) = \min_{u_k} \mathbb{E}_{\omega_k, A_k, B_k} \left\{ x_k^{\mathrm{T}} Q_k x_k + u_k^{\mathrm{T}} R_k u_k + J_{k+1}(A_k x_k + B_k u_k + \omega_k) \right\}$$

Following the similar proof in Theorem (3.1), we imply

$\mu^* k(x_k) = L_k x_k$, where $L_k = -(R_k + \mathbb{E}\{B_k^{\mathrm{T}} K_{k+1} B_k\})^{-1} \mathbb{E}\{B_k^{\mathrm{T}} K_{k+1} A_k\}$

and the cost-to-go function $J_k(x_k) = x_k^{\mathrm{T}} K_k x_k +$ constant, with

$$\begin{aligned} K_N &= Q_N \\ K_k &= \mathbb{E}\{A_k^{\mathrm{T}} K_{k+1} A_k\} - \mathbb{E}\{A_k^{\mathrm{T}} K_{k+1} B_k\} \\ &\quad (R_k + \mathbb{E}\{B_k^{\mathrm{T}} K_{k+1} B_k\})^{-1} \mathbb{E}\{B_k^{\mathrm{T}} K_{k+1} A_k\} + Q_k \end{aligned}$$

However, there are several drawbacks for random matrices systems:

1. Certainty Equivalence may not hold (what is certainty equivalence?) Formally speaking, the optimal controller may not be the same as for the corresponding deterministic problem, when the randomness is replaced by its expectation.

2. Riccati equation may not converge to a steady-state.

Consider the scalar stationary system such that for the update for $K_{k+1}$, the random matrices $A_k, B_k$ are replaced by their expectation. Therefore, the update for $K_{k+1}$ can be written as

$$\begin{cases} K_{k+1} = \tilde{F}(K_k) \\ \tilde{F}(K) = \dfrac{\mathbb{E}\{A^2\} B K}{\mathbb{E}\{B^2\} K + R} + Q + \dfrac{T K^2}{\mathbb{E}\{B^2\} K + R} \\ T = \mathbb{E}\{A^2\} \mathbb{E}\{B^2\} - (\mathbb{E}\{A\})^2 (\mathbb{E}\{B\})^2 \end{cases}$$

When $T = 0$, the problem reduces to the case where $A, B$ are not random; and we can obtain the well-studied asymptotic behavior; the convergece also happens as $T$ is small. However, for large enough $T$, as shown in Fig (3.2), the graph of $F$ do not intersect with the 45-degree line on the positive interval, i.e., the Riccati equation diverges to infinity.

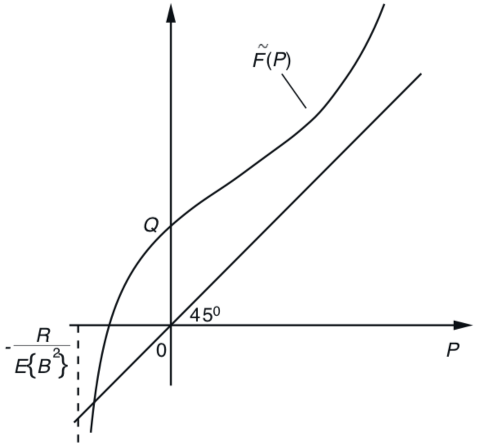

**Figure 3.2:** Graphical Illustration for a scalar stationary system

## 3.2 Inventory Control

- State: inventory level $x_k$
- Control: number of orders $u_k$
- Disturbence: demand $\omega_k$
- System dynamics: $x_{k+1} = x_k + u_k - \omega_k$, $k = 0, 1, \dots, N-1$
- Stage cost:

  $$c \cdot u_k + H(x_k + u_k)$$

  where

  $$H(x_k + u_k) := \mathbb{E}_{\omega_k}\left[p \cdot (x_k + u_k - \omega_k)^{-} + h \cdot (x_k + u_k - \omega_k)^{+}\right]$$

  The goal is to minimize the total cost (question: do we need to add the expectation symbol?)

  $$\mathbb{E}\left\{\sum_{k=0}^{N-1} (cu_k + H(x_k + u_k))\right\}$$

The dynamic programming algorithm applies as follows:

$$\begin{aligned} J_N(x_N) &= 0 \\ J_k(x_k) &= \min_{u_k \geq 0} \left[ c \cdot u_k + H(x_k + u_k) + \mathbb{E}_{\omega_k} J_{k+1}(x_{k+1}) \right] \end{aligned}$$

**Optimal Policy** Regaring $y_k = x_k + u_k$ as the decision variable, it suffices to solve

$$\begin{array}{ll} \min_{y_k \geq x_k} & G_k(y_k) - cx_k, \\ \text{where} & G_k(y) = cy + H(y) + \mathbb{E}_\omega \left[ J_{k+1}(y - \omega) \right] \end{array}$$

**Proposition 3.1.** If $G_k$ is convex, and $\lim_{|x| \to \infty} G_k(x) \to \infty$, we have the optimal policy

$$\mu_k^*(x_k) = \begin{cases} S_k - x_k, & \text{if } x_k < S_k \\ 0, & \text{if } x_k \geq S_k \end{cases}$$

where $S_k$ is the minimizer of $G_k(y)$.

Now we show the assumption for proposition (3.1) mostly holds. We need to add the assumption $c < p$, since otherwise it would never be optimal to order new iterms at the last period.

**Proposition 3.2.** Assume that $c < p$, then $J_k$ is convex for all $k$ and

$$\lim_{|x| \to \infty} J_k(x) \to \infty, \quad k = 0, 1, \dots, N-1$$

*Proof.* 1. It's easy to see that $J_N \equiv 0$ is convex.

2. Since $c < p$ and $H'(y) \to -p$ as $y \to -\infty$, we imply that $G_{N-1} := cy + H(y)$ has a derivative that becomes negative as $y \to -\infty$ and becomes positive when $y \to \infty$ (see Fig (3.3)). Therefore,

$$\lim_{|y| \to \infty} G_{N-1}(y) = \infty.$$

As have shoen,

$$\mu_{N-1}^*(x_{N-1}) = \begin{cases} S_{N-1} - x_{N-1}, & x_{N-1} < S_{N-1} \\ 0, & x_{N-1} \geq S_{N-1} \end{cases}$$

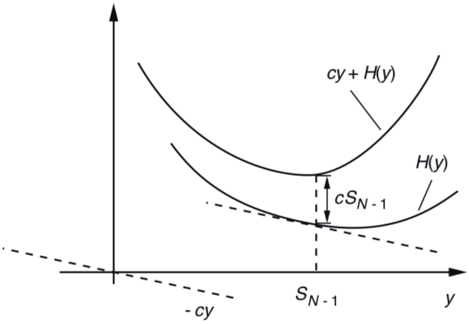

**Figure 3.3:** Graphical Illustration for the structure of cost-to-go function

Or equivalently, $\mu^*_{N-1}(x_{N-1}) = (S_{N-1} - x_{N-1})^+$ (question: can we write in this form?). Substituting $\mu^*_{N-1}$ into $J_{N-1}$, we imply

$$J_{N-1}(x_{N-1}) = \begin{cases} c \cdot (S_{N-1} - x_{N-1}) + H(S_{N-1}), & \text{if } x_{N-1} < S_{N-1} \\ H(x_{N-1}), & \text{if } x_{N-1} \geq S_{N-1} \end{cases}$$

which is clearly a convex function (question: why?)

Therefore, given the convexity of $J_N$, we are able to show the convexity of $J_{N-1}$. Furthermore,

$$\lim_{|y|\to\infty} J_{N-1}(y) = \infty.$$

3. Similarly, given that $J_{k+1}$ is convex and $\lim_{|y|\to\infty} J_{k+1}(y) = \infty$, $\lim_{|y|\to\infty} G_k(y) = \infty$, for $k = N-2, \dots, 0$, we imply

$$J_k(x_k) = \begin{cases} c \cdot (S_k - x_k) + H(S_k) + \mathbb{E}\{J_{k+1}(S_k - \omega_k)\}, & \text{if } x_k < S_k \\ H(x_k) + \mathbb{E}\{J_{k+1}(x_k - \omega_k)\}, & \text{if } x_k \geq S_k \end{cases}$$

where $S_k$ is the unconstrained minimizer of $G_k$. Moreover, we imply $J_k$ is convex, $\lim_{|y|\to\infty} J_k(y) = \infty$ and $\lim_{|y|\to\infty} G_{k-1}(y) = \infty$.

□

Therefore, we reduce the policy-based optimization into the pattern of finding the baseline $S_k$, i.e., a traditional parametric optimization problem.

**Proposition 3.3.** If the disturbance $\omega_k$ is i.i.d., then we imply the stationary optimal policy, i.e., the policy baseline $S_k^*$ becomes constant.

Now suppose further that

$$c(u_k) = \begin{cases} K + c \cdot u_k, & u_k > 0 \\ 0, & u_k = 0 \end{cases}$$

Now the question is that do we obtain the pattern with baseline?

$$\begin{aligned} J_k(x_k) = \min[&\min_{u_k > 0}[K + cu_k + H(x_k + u_k)] + \mathbb{E}_{\omega_k} J_{k+1}(x_{k+1}) \\ &, 0 + H(x_k) + \mathbb{E} J_{k+1}(x_k)] \\ &= \min \left[ \min_{y_k \geq x_j} (K + G_k(y_k)), G_k(x_k) \right] - cx_k \end{aligned}$$

where $f(x_k) := \min_{y_k \geq x_j}(K + G_k(y_k))$ has its shape. Then we need to compare the two curves $f(x_k)$ and $G_k(x_k)$:

$$u_k^* = \begin{cases} S_k - x_k, & \text{if } x_k \leq S_k \\ 0, & \text{if } x_k > S_k \end{cases}$$

- If the inventory is very low (compare the baseline $s_k$), order $u_k^* = S_k - x_k$ to make the inventory into $S_k$
- If not that low, not order anything.

**Remark 3.1.** Unfortunately, $J_{k+1}(x_k)$ is convex does not necessarily imply $J_k(x_k)$ is convex. There is a concept $K$-convexity in literature. Under this $K$-convexity setting, we can show that the control $u_k^*$ still obtain the similar pattern but with two baselines, i.e., the decision can be made by first computing the baseline $(s, S)$, and then obtaining the $(s, S)$ strategy.

The key for reducing computational complexity is to find patterns for decision variable $u_k^*$.

## 3.3 Optimal Stopping Problem

There are two possible controls for a stopping problem:

1. Stop, i.e., incur a one-time stopping cost, and move to a *cost-free*, *absorbing* stop state.
2. Continue, using $x_{k+1} = f_k(x_k, \omega_k)$ and incurring the *cost-per-stage*

Therefore, eacy policy should consist of a partition of a set of states $x_k$ into two regions:

1. Stop region: where we stop?
2. Continue region, where do we continue?

The natural question is that can we obtain an optimal policy that is similar to the pattern of the *baseline* studied in inventory control problem?

**Asset Selling** Suppose an industry wants to sell its iterms. This industry receives a random price $\omega_k$ for $k = 1, \dots, N-1$, which are assumed to be i.i.d. with known distribution. This industry may accept $\omega_k$ and invest the money at a fixed rate of interest $r$; or reject $\omega_k$ and wait for $\omega_{k+1}$. It must accept the last offer $\omega_{N-1}$

We may formulate this problem as follows:

- State: $x_k = \omega_k$
- Decision variable: $u^1$ for sell and $u^2$ for wait.
- Stage reward:

$$g(x_k, u_k, \omega_k) = \begin{cases} x_k(1+r)^{N-1}, & \text{if } u_k = u^1 \\ 0, & \text{if } u_k = u^2 \end{cases}$$

- Therefore, the DP algorithm is by solving the optimization problem below recursively:

$$J_N(x_N) = \begin{cases} x_N & \text{if } x_N \neq T \\ 0, & \text{if } x_N = T \end{cases}$$

$$J_k(x_k) = \begin{cases} \max\left[(1+r)^{N-k} x_k, \mathbb{E}\{J_{k+1}(\omega_k)\}\right] & \text{if } x_k \neq T \\ 0, & \text{if } x_k = T \end{cases}$$

Specifically, note that for $k$-th stage, if $x_k(1+r)^{N-k} \geq \mathbb{E}_{\omega_k}[J_{k+1}(\omega_k)]$, then the industry should decide to sell; otherwise decide to wait.

Define the variable (baseline)

$$\alpha_k = \frac{\mathbb{E}J_{k+1}(\omega)}{(1+r)^{N-k}},$$

then the optimal policy should be:

$$\begin{array}{ll} \text{accept the offer } x_k & \text{if } x_k > \alpha_k \\ \text{reject the offer } x_k & \text{if } x_k < \alpha_k \end{array}$$

In particular, the high baseline corresponds to the high standard for offering the price.

### Future Analysis

**Theorem 3.3.** Based on the assumption of i.i.d. $\omega_k$, the baseline $\alpha_k \geq \alpha_{k+1}$ for all $k$.

*Proof.* First introduce the functions

$$V_k(x_k) = \frac{J_k(x_k)}{(1+r)^{N-k}}, \qquad x_k \neq T$$

The DP algorithm reduces into

$$\begin{aligned} V_N(x_N) &= x_N \\ V_k(x_k) &= \max\left[x_k, (1+r)^{-1}\mathbb{E}_\omega\{V_{k+1}(\omega)\}\right] \end{aligned}$$

Note that $\alpha_k = \mathbb{E}\{V_{k+1}(\omega)\}/(1+r)$. It's clear that

$$V_{N-1}(x) \geq V_N(x), \quad \forall x \geq 0$$

Following the similar procedure, we derive $V_k(x) \geq V_{k+1}(x)$ for all $x$ and $k$. Since $\alpha_k = \mathbb{E}\{V_{k+1}(\omega)\}/(1+r)$, we obtain $\alpha_k \geq \alpha_{k+1}$, as desired. □

**Remark 3.2.** We can also show that $\alpha_k \to \bar{\alpha}$ as $k \to \infty$, where $\bar{\alpha}$ is a constant satisfying

$$(1+r)\bar{\alpha} = P(\bar{\alpha})\bar{\alpha} + \int_{\bar{\alpha}}^{\infty} \omega \, \mathrm{d}P(\omega)$$

Therefore, for an infinite horizon, the optimal policy is stationary.

Now we show that $\{\alpha_k\}$ is monotonically decreasing.

Define

$$V_k(x_k) = \frac{J_k(x_k)}{(1+r)^{N-k}} \implies V_k(x_k) = \max\left(x_k, \frac{\mathbb{E}V_{k+1}(\omega)}{1+r}\right)$$

and $V_N(x_N) = x_N$.

Note that we show $V_{k+1}(x) \leq V_k(x)$ first:

$$\begin{aligned} V_N(x) &= x \\ V_{N-1}(x) &= \max(x, \mathbb{E}V_N(\omega)) \end{aligned}$$

It's clear that $V_N(x) \leq V_{N-1}(x)$.

$$\begin{aligned} V_K(x) &= \max(x, \frac{\mathbb{E}V_{k+1}(\omega)}{1+r}) \\ V_{k-1}(x) &= \max(x, \frac{\mathbb{E}V_k(\omega)}{1+r}) \end{aligned}$$

It's clear that $V_k(x) \leq V_{k-1}(x)$ for any $k$. Therefore, $\{\alpha_k\}$ is monotonically decreasing.

Note that the key for this proof is the i.i.d. of $\omega_k$.

- Pattern
- Trends
- Convergence: If there are inifinte horizons, what is the bottleneck for the policy.

In this case,

$$\begin{aligned} \alpha_k = \frac{\mathbb{E}V_{k+1}(\omega)}{1+r} &= \frac{1}{1+r}\left[\int_0^{\alpha_{k+1}} \alpha_{k+1} dP(\omega) + \int_{\alpha_{k+1}}^{\infty} \omega dP(\omega)\right] \\ &= \frac{P(\alpha_{k+1})}{1+r}\alpha_{k=1} + \frac{1}{1+r}\int_{\alpha_{k+1}}^{\infty} \omega dP(\omega) \end{aligned} \tag{3.1}$$

where the $P(\alpha_{k+1})$ is the cdf, and $\alpha_k$ is bounded.

Start from $\alpha_N = 0$, derive $\alpha_{N-1}, \ldots, \alpha_0$ sequentially.

When $N \to \infty$, note that computing the equation above is based on the derivation of $\alpha_{N+1}$. We can imply the convergence of $\alpha$, and we can compute $\alpha_{N+1}$ based on (3.1).

# 4

# Dynamic Programming with imperfect information

## 4.1 Problem Setting

- State: $x_k$
- Control: $u_k \in \mu_k(x_k)$
- Disturbance: $\omega_k$
- Dynamics: $x_{k+1} = f_k(x_k, u_k, \omega_k)$
- stage cost: $g_N(x_N), g_k(x_k, u_k, \omega_k)$

Imperfect Information Setting

- Observation: $Z_k$
- measurment: $z_k = h_k(x_k, u_{k-1}, v_k)$, where $v_k$ denotes the measurment noise, and $v_k$ is dependent on $x_0, \dots, x_{k-1}, u_0, \dots, u_{k-1}, v_0, \dots, v_{k-1}$
- Information vector: $I_k = (z_0, \dots, z_k, u_0, \dots, u_{N-1})$
- The goal is to derive the optimal policy

$$\pi = (\mu_0(I_0), \dots, \mu_{N-1}(I_{N-1}))$$

where $u_k^* = \mu_k(I_k)$ to minimize the cost function

$$J_\pi = \mathbb{E}_{\omega_0,\ldots,\omega_{N-1},v_0,\ldots,v_{N-1},x_0}\left\{\sum_{k=0}^{N-1} g_k(x_k,\mu_k(I_k),\omega_k) + g_N(x_N)\right\}$$

Therefore, we need the distribution of $x_0$.

## 4.2 Reformulation as a perfect state information system

- State: $I_k = (z_0,\ldots,z_k,u_0,\ldots,u_{k-1})$
- Control: $u_k \in U_k$
- Disturbance: $z_{k+1}$
- System dynamics:

$$I_{k+1} = (I_k, u_{k+1}, z_{k+1})$$

where $z_{k+1}$ is the unknown term, considered as the noise

- Stage cost:

$$\mathbb{E}_{x_k,\omega_k}\left(g_k(x_k,u_k,\omega_k) \mid I_k,u_k\right) = \sum_{x_k} \mathbb{E}_{\omega_k} g_k(x_k,u_k,\omega_k)\cdot P(x_k \mid I_k)$$

Therefore, this stage cost is a function w.r.t. $I_k, u_k$.

Dynamic algorithm applies as follows:

$$\begin{aligned} J_{N-1}(I_{N-1}) = \min_{u_{N-1}\in\mu_{N-1}} & \mathbb{E}_{x_{N-1},\omega_{N-1}}\left[g_{N-1}(x_N,u_{N-1},\omega_{N-1}) \mid I_{N-1},\omega_{N-1}\right] \\ & + \mathbb{E}_{\omega_{N-1},x_{N-1}}[g_N(x_N) \mid I_{N-1},u_{N-1}] \end{aligned}$$

For $k = N-1, N-2, \ldots, 1$,

$$\begin{aligned} J_k(I_k) = \min_{u_k\in U_k} & \mathbb{E}_{x_k,\omega_k}[g_k(x_k,u_k,\omega_k) \mid I_k,u_k] \\ & + \mathbb{E}_{z_{k+1}}\left[J_{k+1}(I_k,u_{k+1},z_{k+1}) \mid I_k,u_k\right] \end{aligned}$$

We need the conditional distribution $P(x_k \mid I_k)$ and $P(z_{k+1} \mid I_k)$.

## 4.3 Machine Repair Example

- State: $x_k$ is $P$ or $\overline{P}$
- Observation: $Z_k = \{G, B\}$
- Control: $u_k = \{C, S\}$, where $C$ denotes continue, $S$ denotes stop to check.
- Disturbance:

  System Dynamics:

  $$P(x_{k+1} = (P, \overline{P}) \mid x_k = P) = (2/3, 1/3)$$

  $$P(x_{k+1} = (P, \overline{P}) \mid x_k = \overline{P}) = (0, 1)$$

- Stage cost:

  $$g(x_k, u_k) = \begin{cases} 0, & (x_k, u_k) = (P, C) \\ 2, & (x_k, u_k) = (\overline{P}, C) \\ 1, & (x_k, u_k) = (P, S) \\ 1, & (x_k, u_k) = (\overline{P}, S) \end{cases}$$

- Measurement:

  $$P(Z_k = G \mid x_k = P) = 3/4, \quad P(Z_k = B \mid x_k = \overline{P}) = 3/4$$

- Cost-to-go Function:

  $$N = 2, \quad J_N(I_N) = 0$$

  $$\begin{aligned} J_1(I_1) &= \min_{u_1 \in \{C,S\}} \mathbb{E}_{x_1}[g(x_1, u_1) \mid I_1, u_1] + \underbrace{\mathbb{E}J_2}_{0} \\ &= \min_{u_1} \{\mathbb{E}_{x_1}[g(x_1, c) \mid I_1, c], \mathbb{E}_{x_1}[g(x, s) \mid I_1, s]\} \end{aligned}$$

  We need the conditional distribution $P(x_k \mid I_k)$, and the initial distribution $P(x_0 = P) = 2/3, P(x_0 = \overline{P}) = 1/3$.

Prior: $P(x_0)$; conditional distribution: $P(z_0 \mid x_0)$; posterior distribution: $P(x_0 \mid z_0)$. Marginal: $P(z_0)$

Bayes's Theorem:

$$P(x_0 \mid z_0) = \frac{P(x_0, z_0)}{P(z_0)} = \frac{P(z_0 \mid x_0)P(x_0)}{\sum_{x_0} P(z_0 \mid x_0)P(x_0)}$$

Therefore,

$$P(x_0 = P \mid z_0 = G) = \frac{P(x_0 = P, z_0 = G)}{p(z_0 = G)} = \frac{2/3 \times 3/4}{2/3 \times 3/4 + 1/3 \times 1/4} = 6/7$$

$$P(x_1 \mid I_1) = P(x_1 \mid z_0, z_1, u_1) = \frac{P(z_1 \mid x_1, z_0, u_1) \cdot P(x_1 \mid z_0, u_0)}{P(z_1 \mid z_0, u_1)}$$

**Notations** We denote $P(x_k \mid I_k)$ as $P_{x_k|I_k}$. We call $P_{x_k|I_k}$ as the *sufficient statistics*, which can be used for decision:

$$\tilde{\mu}_k(I_k) = \tilde{\mu}(P_{x_k|I_k})$$

For $G(P_{x_k|I_k}) = S(I_k)$ and $P_{x_k|I_k} = G'(S'(I_k))$, if $P_{x_k|I_k}$ is sufficient, then $S'(I_k)$ is sufficient.

1. Estimator: the goal is to estimate $x_k$
2. Actuator

$p_k = P(x_k = \text{present} \mid I_k)$, where $P(x_k \mid I_k)$ is a vector, if $x_k$ can only take finite discrete values, e.g., if $x_k = \{1, \ldots, n\}$, what's the dimension for the belief state $P_{x_k|I_k}$? It's $n-1$, independent of the stage $k$.

Realize the data reduction.

When $x_k$ is a continuous rv given $I_k$, dimension of $P_{x_k|I_k}$? In such case, $P_{x_k|I_k}$ follows some probability distributions.

**Finite State System** MDP

1. System dynamics is based on Markov Chain

$$p_k^i = p(x_k = i \mid I_k), \boldsymbol{p}_k = (p_k^1, \ldots, p_k^n)^{\mathrm{T}}$$

2. If no control, then for homogeneous markove chain, $p_{ij} = P(x_{k+1} = j \mid x_k = i)$
3. Otherwise,

$$p_{ij}(u) = P(x_{k+1} = j \mid x_k = i, u_k = u)$$

Dynamic programming: suppose $g(i)$ denotes cost for $x_N = i$.

$$J_N(\boldsymbol{p}_N) = \sum_{i=1}^{n} p_N^i g(i)$$

Stage cost:

$$g(x_k, u_k) = \mathbb{E}_{\omega_k}[g(x_k, u, \omega_k)]$$

Moreover,

$$J_{N-1}(p_{N-1}) = \min_{u_{N-1}} \mathbb{E}\left[g(x_{N-1}, u_{N-1}) \mid I_{N-1}\right] + \mathbb{E}g(\phi(x_{N-1}, u_{N-1})) \mid I_{N-1}, u_{N-1})$$

Moreover,

$$J_k(\boldsymbol{p}_k) = \min_{u_k} \{\mathbb{E}_{x_k}(g_k(x_k, u_k) \mid I_k) + \mathbb{E}_{z_{k+1}}(J_{k+1}(\phi(p_k, u_k, z_{k+1})) \mid I_k, u_k\}$$

**Instruction Problem (Optimal Stopping)** Given a student.

1. State: $x_k = \{L, \overline{L}\}$.
2. Control: $\{T, \overline{T}\}$
3. Observation: $Z_k = \{R, \overline{R}\}$

Each time if choose $T$, pay for $I$; otherwise face certain cost:

$$\text{terminal cost} = \begin{cases} 0, & \text{if } x_k = L \\ C, & \text{if } x_k = \overline{L} \end{cases}$$

# 5

# The Distribution and Installation

## 5.1 Pre-requisites

You will need a working LaTeX installation. We recomend using pdflatex to process the files. You will also need biber.exe installed. This is distributed as part of the latest versions of LiveTex and MikTex. If you have problems, please let us know.

## 5.2 The Distribution

The distribution contains 2 folders: `nowfnt` and `nowfnttexmf`.

### 5.2.1 Folder `nowfnt`

This folder contains the following files using a flat stucture required to compile a FnT issue:

- essence_logo.eps
- essence_logo.pdf
- now_logo.eps
- now_logo.pdf

- nowfnt.cls
- nowfnt-biblatex.sty
- NOWFnT-data.tex

It also contains the following folders:

**journaldata** A set of data files containing the journal-specific data for each journal. There are three files per journal:
<jrnlcode>-editorialboard.tex
<jrnlcode>-journaldata.tex
<jrnlcode>-seriespage.tex
<jrnlcode> is the code given in Appendix A. You will need these three files to compile your article.

**SampleArticle** This folder contains this document as an example of an article typeset in our class file. The document is called `Fn`Tarticle.tex. It also contains this PDF file and the .bib file.

### 5.2.2 Folder `nowfnttexmf`

This folder contains all the files required in a texmf structure for easy installation.

## 5.3 Installation

If your LaTeX installation uses a localtexmf folder, you can copy the nowtexmf folder to the localtexmf folder and make it known to your TeX installation. You can now proceed to use the class file as normal.

If you prefer to use the flat files, you will need to copy all the required files each time into the folder in which you are compiling the article. Do not forget to copy the three data files for the specific journal from the folder `j`ournaldata.

You may need to configure your TeX editor to be able to run the programs. If you have problems installing these files in your own system, please contact us. We use Computer Modern fonts for some of the journals. You will need to make sure that these fonts are installed. Refer to your system documentation on how to do this.

# 6

# Quick Start

The now-journal class file is designed is such a way that you should be able to use any commands you normally would. However, do **not** modify any class or style files included in our distribution. If you do so, we will reject your files.

The preamble contains a number of commands for use when making the final versions of your manuscript once it has been accepted and you have been instructed by our production team.

## 6.1 \documentclass

The options to this command enable you to choose the journal for which you producing content and to indicate the use of biber.

```
\documentclass[<jrnlcode>,biber]{nowfnt}.
```

<jrnlcode> is the pre-defined code identifying each journal. See Appendix A for the appropriate <jrnlcode>.

## 6.2 \issuesetup

These commands are only used in the final published version. Leave these as the default until our production team instructs you to change them.

## 6.3 \maintitleauthorlist

This is the authors list for the cover page. Use the name, affilliation and email address. Separate each line in the address by \\.

Separate authors by \and. Do not use verbatim or problematic symbols. _ (underscore) in email address should be entered as _. Pay attention to long email addresses.

If your author list is too long to fit on a single page you can use double column. In this case, precede the \maintitleauthorlist command with the following:

```
\booltrue{authortwocolumn}
```

## 6.4 \author and \affil

These commands are used to typeset the authors and the afilliations on the abstract page of the article and in the bibliographic data.

**\author** uses an optional number to match the author with the affiliation. The author name is written <surname>, <firstname>.

**\affil** uses an optional number to match the author with the author name. The content is <affililiation>; <email address>.

## 6.5 \addbibresource

Use this to identify the name of the bib file to be used.

# 7

# Style Guidelines and LaTeX Conventions

In this section, we outline guidelines for typesetting and using LaTeX that you should follow when preparing your document

## 7.1 Abstract

Ensure that the abstract is contained within the

```
\begin{abstract}
```

environment.

## 7.2 Acknowledgements

Ensure that the acknowledgements are contained within the

```
\begin{acknowledgements}
```

environment.

## 7.3 References

now publishers uses two main reference styles. One is numeric and the other is author/year. The style for this is pre-defined in the LaTeX distribution and must not be altered. The style used for each journal is given in the table in Appendix A. Consult the sample-now.bib file for an example of different reference types.

The References section is generated by placing the following commands at the end of the file.

```
\backmatter
\printbibliography
```

## 7.4 Citations

Use standard \cite, \citep and \citet commands to generate citations.

Run biber on your file after compiling the article. This will automatically create the correct style and format for the References.

### 7.4.1 Example citations

This section cites some sample references for your convenience. These are in author/year format and the output is shown in the References at the end of this document.

Example output when using `citet`: **arvolumenumber** is a citation of reference 1 and Bertsekas (1995) is a citation of reference 2.

Example output when using `citep`: (**beditorvolumenumber**) is a citation of reference 3 and (**inproceedings**) is a citation of reference 4.

## 7.5 Preface and Other Special Chapters

If you want to include a preface, it should be defined as follows:

```
    \chapter*{Preface}
    \markboth{\sffamily\slshape Preface}
      {\sffamily\slshape Preface}
```

This ensures that the preface appears correctly in the running headings.

You can follow a similar procedure if you want to include additional unnumbered chapters (*e.g.*, a chapter on notation used in the paper), though all such chapters should precede Chapter 1.

Unnumbered chapters should not include numbered sections. If you want to break your preface into sections, use the starred versions of `section`, `subsection`, *etc.*

## 7.6 Long Chapter and Section Names

If you have a very long chapter or section name, it may not appear nicely in the table of contents, running heading, document body, or some subset of these. It is possible to have different text appear in all three places if needed using the following code:

```
\chapter[Table of Contents Name]{Body Text Name}
\chaptermark{Running Heading Name}
```

Sections can be handled similarly using the `sectionmark` command instead of `chaptermark`.

For example, the full name should always appear in the table of contents, but may need a manual line break to look good. For the running heading, an abbreviated version of the title should be provided. The appearance of the long title in the body may look fine with LaTeX's default line breaking method or may need a manual line break somewhere, possibly in a different place from the contents listing.

Long titles for the article itself should be left as is, with no manual line breaks introduced. The article title is used automatically in a number of different places by the class file and manual line breaks will interfere with the output. If you have questions about how the title appears in the front matter, please contact us.

## 7.7 Internet Addresses

The class file includes the `url` package, so you should wrap email and web addresses with `\url{}`. This will also make these links clickable in the PDF.

# 8

# Compiling Your FnT Article

During the first run using the class file, a number of new files will be created that are used to create the book and ebook versions during the final production stage. You can ignore these until preparing the final versions as described in Section 8.3. A complete list of the files produced are given in Appendix B.

## 8.1 Compiling Your Article Prior to Submission

To compile an article prior to submission proceed as follows:

**Step 1:** Compile the LaTeX file using pdflatex.

**Step 2:** Run biber on your file.

**Step 3:** Compile again using pdfLaTeX. Repeat this step.

**Step 4:** Inspect the PDF for bad typesetting. The output PDF should be similar to FnTarticle.pdf. Work from the first page when making adjustments to resolve bad line breaks and bad page breaks. Rerun pdfLaTeX on the file to check the output after each change.

## 8.2 Preparing the Final Versions

If you choose the option to compile the final versions of your PDF for publication, you will receive a set of data from our production team upon final acceptance. With the exception of "lastpage", enter the data into the `\issuesetup` command in the preamble.

**lastpage=** This is the last page number in the sequential numbering of the journal volume. You will need to enter this once you have compiled the article once (see below).

## 8.3 Compiling The Final Versions

The final versions should be created once you received all the bibliographic data from our Production Team and you've entered it into the preamble. You will be creating a final online journal version pdf; a printed book version pdf; and an ebook version pdf.

**Step 1:** Compile the LaTeX file using pdfLaTeX.ArAuthor *et al.*, 2014

**Step 2:** Run biber on your file.

**Step 3:** Compile again using pdfLaTeX. Repeat this step.

**Step 4:** Inspect the PDF for bad typesetting. The output PDF should be similar to FnTarticle.pdf. Work from the first page when making adjustments to resolve bad line breaks and bad page breaks. Rerun pdfLaTeX on the file to check the output after each change.

**Step 5:** When you are happy with the output make a note of the last page number and enter this in `\issuesetup`.

**Step 6:** Compile the article again.

**Step 7:** Open the file <YourFilename>-nowbook.tex. This will generate the printed book version pdf.

**Step 8:** Compile the LaTeX file using pdflatex.

**Step 9:** Run biber on your file.

**Step 10:** Compile again using pdfLaTeX. Repeat this step.

**Step 11:** Repeat steps 7-10 on the file: <YourFilename>-nowebook.tex. This will generate the ebook version pdf.

**Step 12:** Repeat steps 7-10 on the file: <YourFilename>-nowplain.tex. This will generate a plain version pdf of your article. If you intend to post your article in an online repository, please use this version.

## 8.4 Wednesday

**Proposition 8.1.** *$G$ is hamiltionian implies that for any nonempty $S \subseteq V$, $G - S$ has at most $|S|$ components.*

**Theorem 8.1.** *$G$ is Hamiltionian if for any vetices $v, w$ such that $(v, w) \notin E$,*

$$\deg(v) + \deg(w) \geq n$$

### Knapsacle Problem

$$\begin{array}{ll} \max & \sum_{i=1}^{n} v_i u_i \\ \text{such that} & \sum_{i=1}^{n} w_i u_i \leq W \\ & u_i \in \{0,1\},\ i = 1, \dots, n \end{array}$$

This is the binary integer programming problem, which is NP-complete. We cannot find exact solution to this problem. There are $2^n$ possible solutions, which requires exponential time.

We can find some "pseudo"-polynomial algorithm. Assumption:

1. $W$ is an integer.

State: remaining capacity, define as $X_k$.
Stage: iterm $1, \dots, n$.

$$J_N(x_N) = \begin{cases} v_N & \text{if } N \leq X_N \\ 0, & \text{if } w_N > X_N \end{cases}$$

$$J_N(x_{N-1}) = \begin{cases} 0 + J_N(X_N), & \text{if } w_{N-1} > X_{N-1}, \text{Here } X_N = \\ \max\{v_{N-1} + J_N(x_N), 0 + J_N(x_N)\}, & \text{if } w_{N-1} \leq X_{N-1} \end{cases}$$

Here why the DP has “pesduo”-polynomial time, and the $w_i$ should be integer?

Let’s list states $\{x_1, \dots, x_N\}$. For each stage $k$, there would be $W+1$.

Each iteration at most 2 computation, and therefore we face $2(W+1)\cdot N$

### 8.4.1 Label Correcting Methods

Shortest Path. Back up some information.

The label refers to the intermediate information.

$A^*$-algorithm; Bellman-Ford Algorithm

Benchmark: MILP, CPLEX, CVX.

### 8.4.2 DP problems with perfect state information

**Linear Quadratic System** Let $x_k \in \mathbb{R}^1$ be the state; $u_k \in \mathbb{R}^n$ be the control; $\omega_k \in \mathbb{R}^n$ be the disturbance.

System Dynamics.

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

Stage cost:

$$x_k' Q_k x_k + u_k' R_k u_k$$

Here $Q_k$ is required to be positive-semi-definite symmetric matrix; $R_k$ to be positive-definite symmetric matrix.

$$J_{n-1}(x_{n-1}) = \min_{u_{n-1}} \mathbb{E}_\omega \left\{ x_{n-1}' Q_{n-1} x_{n-1} + u_{n-1}' R_{n-1} u_{n-1} + J_n(x_n) \right\}$$

where

$$J_n(x_n) = (A_{n-1}x_{n-1} + Bu_{n-1} + \omega)' Q_n (A_{n-1}x_{n-1} + Bu_{n-1} + \omega)$$

### 8.4.3 Linear Quadratic

Dynamics:

$$x_{k+1} = A_k x_k + B_k u_k + \omega_k$$

Stage cost:

$$x_k' Q_k x_k + u_k' R_k u_k,\ Q_k \succeq 0, R_k \succ 0$$

Cost to go function:

$$J_k(x_k) = \min_{u_k} \mathbb{E}_{\omega_k}[x_k' Q_k x_k + u_k' R_k u_k + J_{k+1}(x_{k+1})]$$

The final cost is

$$J_N(x_N) = x_N' Q_N x_N$$

At $N-1$ stage

$$x_{N-1}' Q_{N-1} x_N + u_{N-1}' R_{N-1} u_{N-1}$$
$$+ \mathbb{E}[(A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + \omega_{N-1})' Q_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + \omega_{N-1})]$$

Optimization model:

$$\frac{\partial}{\partial u}(u'Hu + 2r'u + c) = 2Hu + 2r \implies Hu = -r \implies u^* = -H^{-1}r$$

For $N-1$ stage, we have

$$\begin{aligned} H &= R_{N-1} + B_{N-1}' Q_N B_{N-1} \\ r' &= x_{N-1}' A_{N-1}' Q_N B_N \\ c &= x_{N-1}'(Q_{N-1} + A_{N-1}' Q_N A_{N-1}) x_{N-1} + \mathbb{E}_\omega(\omega_{N-1}' Q_N \omega_{N-1}) \end{aligned}$$

Therefore,

$$u_{N-1}^* = -(R_{N-1} + B_{N-1}' Q_N B_{N-1})^{-1} B_N Q_N A_{N-1} x_{N-1},$$

which is linaer in terms of $x_{N-1}$, which is so called linear controller.

Therefore,

$$J_{N-1}(x_{N-1}) = x_{N-1}' K_{N-1} x_{N-1} + \mathbb{E}(\omega_{N-1}' Q_N \omega_{N-1})$$

The linear controller will be tested during mid-term or final.

$$J_0(x_0) = x_0' K_0 x_0 + \sum_{k=0}^{N-1} \mathbb{E}_\omega\{\omega_k' K_{k+1} \omega_k\}$$

$$K_{k-1} = f(K_k) \implies K_{k-1} = K_k, \quad \text{for large } k.$$

**Riccati Equation** for stationary ststem, i.e., $A_k = A, B_k = B, Q_k = Q, R_k = R$.

$$K = A'(K - KB(B'KB + R)^{-1}B'K)A + Q$$

Therefore as time goes long enough, the cost to go function $J_k(x_k)$ will be a constant plus over stages. such a constant $K$ is called the *optimal stationary controller.*

Stability. For $u^* = Lx$, we imply

$$x_{k+1} = Ax_k + Bu_k + \omega_k = (A + BL)x_k + \omega_k$$

Care about $\lim_{k\to\infty}(A + BL)^k = 0$.

Here

$$L = -(B'KB + R)^{-1}B'KA$$

It suffices to solve

$$P_{k+1} = A^2 \left( P_k - \frac{B^2 P_k^2}{B^2 P_k + R} \right) + Q$$

Then consider the case that $A_k, B_k$ are all random matrices, i.e., independent. $Q_k \succeq 0, R_k \succ 0$.

$$L_k = -\left(R_k + \mathbb{E}(B_k' K_{k+1} B_k)\right)^{-1} \mathbb{E}(B_j' K_{k+1} A_k)$$

Note that

$$P_\infty = \frac{\mathbb{E}A^2 RP}{R + EB^2 P} + \frac{\mathbb{E}A^2 \mathbb{E}B^2 - (\mathbb{E}A)^2(\mathbb{E}B)^2}{R + \mathbb{E}B^2 P}$$

Certainty Equivalence

State: $x_k$, inventory level

Control: $u_k \geq 0$, number of orders placed

Disturbance: $\omega_k$: damand

Dynamics:

$$x_{k+1} = (x_k + u_k - \omega_k)$$

Stage cost:

- Ordering cost: $c \cdot u_k$

- Maintaining cost: full backlog.$\mathbb{E}_{\omega_k}(r(x_k + u_k - \omega_k))$.

  where $r(z) = hz^+ + pz^-$ is a convex function.

  We imply the maintaining cost is $H(x_k + u_k)$, which is convex.

$$J_k(x_k) = \min_{u_k \geq 0} \{cu_k + H(x_k + u_k) + \mathbb{E}[J_{k+1}(x_k + u_k - \omega_k)]\}$$

Define $Y = x_k + u_k$, and therefore

$$G(Y) = cY + H(Y) + \mathbb{E}[J(Y - \omega_k)]$$

Therefore, we can always set $Y = x_k + u_k = S_k$, where $S_k$ minimizes $G(Y)$.

# Acknowledgements

The authors are grateful to Ulrike Fischer, who designed the style files, and Neal Parikh, who laid the groundwork for these style files.

# Appendices

# A

# Journal Codes

The table below shows the journal codes to be used in `\documentclass`. For Example: `\documentclass[ACC,biber]{nowfnt}`

| **J**ournal | <jrnlcode> | **R**ef. Style |
|---|---|---|
| Annals of Corporate Governance | ACG | Author/Year |
| Annals of Science and Technology Policy | ASTP | Author/Year |
| FnT Accounting | ACC | Author/Year |
| FnT Comm. and Information Theory | CIT | Numeric |
| FnT Databases | DBS | Author/Year |
| FnT Econometrics | ECO | Author/Year |
| FnT Electronic Design Automation | EDA | Author/Year |
| FnT Electric Energy Systems | EES | Author/Year |
| FnT Entrepreneurship | ENT | Author/Year |
| FnT Finance | FIN | Author/Year |
| FnT Human-Computer Interaction | HCI | Author/Year |
| FnT Information Retrieval | INR | Author/Year |
| FnT Information Systems | ISY | Author/Year |
| FnT Machine Learning | MAL | Author/Year |
| FnT Management | MGT | Author/Year |
| FnT Marketing | MKT | Author/Year |
| FnT Networking | NET | Numeric |

*Continues*

| **J**ournal | <jrnlcode> | **R**ef. Style |
|---|---|---|
| FnT Optimization | OPT | Numeric |
| FnT Programming Languages | PGL | Author/Year |
| FnT Robotics | ROB | Author/Year |
| FnT Privacy and Security | SEC | Author/Year |
| FnT Signal Processing | SIG | Numeric |
| FnT Systems and Control | SYS | Author/Year |
| FnT Theoretical Computer Science | TCS | Numeric |
| FnT Technology, Information and OM | TOM | Author/Year |
| FnT Web Science | WEB | Author/Year |

# B

# Files Produced During Compilation

The files that are created during compilation are listed below. The additional *.tex files are used during the final production process only. See Section 8.3.

```
<YourFilename>-nowbook.tex
<YourFilename>-nowchapter.tex
<YourFilename>-nowebook.tex
<YourFilename>-nowechapter.tex
<YourFilename>-nowsample.tex
<YourFilename>-nowplain.tex
<YourFilename>.aux
<YourFilename>.bbl
<YourFilename>.bcf
<YourFilename>.blg
<YourFilename>.log
<YourFilename>.out
<YourFilename>.pdf
<YourFilename>.run.xml
<YourFilename>.synctex.gz
<YourFilename>.tex
<YourFilename>.toc
```

# References

ArAuthor, A., B. Author, and C. Author (2014). "An article on something interesting (volume only)". *Journal of interesting Things*. 6: 1–122. I have something special to say about this publication. ISSN: 0899-8256. DOI: 10.5161/202.00000013. URL: http://www.nowpublishers.com/ (accessed on 07/10/2014).

Bertsekas, D. (1995). "Dynamic Programming and Optimal Control". In: vol. 1.

Sutton, R. S. (1988). "Learning to predict by the methods of temporal differences". *Machine Learning*. 3: 9–44.

Watkins, C. J. C. H. and P. Dayan (1992). "Q-learning". In: *Machine Learning*. 279–292.