オペレーションズ・リサーチ

Operations Research: Models, Algorithms, and Implementations

劉子昂

2025-09-17

目次

Preface

OR の基礎と応用を学ぶための教科書です.

講義

本授業では、オペレーションズ・リサーチ（OR）の中の代表的な手法である PERT、在庫理論、待ち行列理論、動的計画法、階層分析法、及び包絡分析法の数理を理解し、具体的な問題への応用を学ぶ。

講義情報

- 講義名：オペレーションズ・リサーチ B
- 曜日：水曜日
- 時限：2 時限目（10:50〜12:30）
- 教室：西館 W202
- 担当教員：劉 子昂

到達目標

各分野について、下記の事項を目標して講義を行う。

- **PERT** の計算と解析方法を理解し、プロジェクトの評価を行うことができる。
- **在庫モデル**を理解し、自分で式を構築及び解析することができる。
- **待ち行列理論**の重要な式や定理を理論的に導出し、それらを適切に解釈することができる。
- **動的計画法**の基本的な考え方を理解し、簡単な問題への適用ができる。
- **階層分析法**による意思決定の手法を理解し、一対比較行列からウェイトと整合性を計算することができる。
- **包絡分析法**における CCR モデルを理解し、得られた結果を解釈することができる。

さらに、これらの手法を用いて比較的簡単な現象をモデル化し、解析することができる。

OR は重要

経営工学において、最も重要な学問分野の一つ。

日本経営工学会によると、「解決すべき課題の数理モデルを構築し、最適な手法を求めるオペレーションズ・リサーチ（OR）という分野は、経営工学の主要なテーマとなっています」。

海外では、管理科学（Management Science）と OR は同義語として使われることもよくある。

OR の全体像

- 線形計画法
- 整数計画法
- 非線形計画法
- **動的計画法**
- グラフ理論・ネットワーク
- シミュレーション
- **在庫モデル**
- **待ち行列**
- **多基準意思決定分析**
- **プロジェクトマネジメント**
- ...

OR は難しい？

基礎知識が必要

微分積分、線形代数、確率、統計の基礎知識が必要です。これらの基礎が不十分な場合、授業についていけないです。基礎知識が不十分な場合、必ず復習してください。

この講義では、以下の工夫をしています。

- 付録に私が書いた基礎知識のまとめがあります。随時更新しますので、参考にしてください。
- 講義資料には例題、図、演習問題を多く用意しています。
- プログラミングの実装例も示します。

数式が多い

OR は、問題を数理的にモデル化し、解析する学問です。数式をたくさん使い、証明も多いです。数式を読むのが苦手な人は、慣れるまで大変かもしれません。

この講義では、以下の工夫をしています。

- 証明は省略なく丁寧に行います。
- 私が推測したわかりにくいところをコラムで補足します。

何を学ぶ？

モデル

アルゴリズム

実装

授業時間外の学習

本授業の準備・復習等の授業時間外学習は、4 時間を標準とする

必要なもの

- 本講義では, 受講者自身のノート PC を用いて演習を行います. 毎週必ずノート PC を持参してください.

私語

- 講義中の私語は厳禁です.
- 注意してもやめない場合は、減点を行います。

成績評価

- 期末試験（100%）
- 一回の欠席につき、10 点減点。4 回以上の欠席は単位取得不可。

第I部

在庫モデル

第1章　在庫管理とは

商店・工場・倉庫などで、原材料・部品・製品などを適切に管理することを**在庫管理**（Inventory Management）という。一般的に、在庫管理の目的は、顧客の需要を満たしつつ、在庫に関わる費用を最小化することである。

> **ノート**
>
> 豊田自動車が提唱した**ジャストインタイム**（Just In Time, JIT）は、生産方式としてよく知られている。
> JIT とは必要なものを、必要な時に、必要な量だけ生産することである。
> JIT の目的は、在庫を最小限に抑え、効率的な生産を実現することである。
> アメリカの研究者らは、その生産方式を体系化し、**リーン生産方式**（Lean Manufacturing）という概念を提唱した。
>
> - トヨタ生産方式

在庫が多すぎると、保管費用がかかる。逆に、在庫が少なすぎると、欠品が発生し、顧客の需要を満たせなくなる。在庫管理は次の二つの問題を決定する。

1. どのくらいの量を発注するか？（発注量）
2. いつ発注するか？（発注時期）

科学的在庫管理（Scientific Inventory Management）では、これらの問題に答えるために、次の手順で在庫管理を行う。

1. 在庫システムを数学モデルとして定式化する。
2. 最適な発注量と発注時期を決定する。

1.1　在庫モデルの分類

在庫モデルは、次のような要素で分類される。

需要（demand） 需要が決定論的 (Deterministic) か確率的（Stochastic）か。
観測（review） 在庫量を連続観測 (Continuous Review) するか、周期観測 (Periodic Review) するか。連続観測の場合、在庫量が連続的に観測で

き、いつでも発注が可能である。周期観測の場合、一定の期間（例えば 1 週間）ごとに在庫量を観測する。

リードタイム（lead time） 発注から納品までの期間。調達期間とも呼ばれる。リードタイムが決定論的か確率的か。また、リードタイムが 0 かどうか。在庫モデルを単純化するために、リードタイムを 0 とし、発注から納品までの期間を無視することもある。

バックオーダー（backorder） バックオーダーが許容されるかどうか。需要が在庫を上回った場合、バックオーダーが許容されると、欠品が発生しても、後で需要を満たすことができる。バックオーダーが許容されない場合、欠品が発生すると、上回った需要は失われ、機会損失が発生する。

計画期間（planning horizon） 単一期間 (Single Period) か、複数期間 (Multi Period) か、無限 (Infinite) か。

以下の表に、需要と観測に基づく、古典的な在庫モデルを示す。

在庫モデル	需要	観測
EOQ モデル	決定論的	連続観測
Wagner-Whitin	決定論的	周期観測
安全在庫	確率的	連続観測
新聞売り子問題	確率的	周期観測

1.2　在庫の費用

ここでは、在庫に関わる費用を紹介する。

発注費用（ordering cost） 発注量に関わらず、1 回の発注にかかる費用。調達費用、固定費用（fixed cost）などとも呼ばれる。通常、1 回の発注にかかる費用を K とする。

購入費用（purchase cost） 商品を購入するためにかかる費用。通常、単位あたりの購入費用を c とする。

欠品費用（stockout cost） 需要が在庫を上回った場合に発生する費用。通常、単位あたりの欠品費用を p とする。

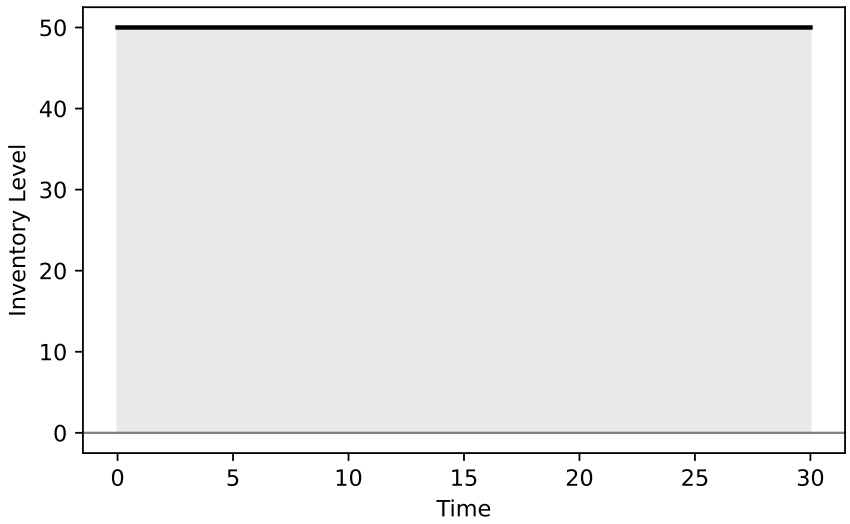
保管費用（holding cost） 在庫を保管するためにかかる倉庫費用、保険費用、税金、機会費用など。通常、単位時間あたりの 1 単位あたりの保管費用を h とする。

例えば、1 日あたり 1 単位の在庫を保管するために、h の費用がかかるとする。30 日間、50 単位の在庫を保管するための総保管費用は $30 \times 50 \times h = 1500h$ となる。下の図を見ると、保管費用は

$$\text{面積} \times h = 1500h$$

となることがわかる。

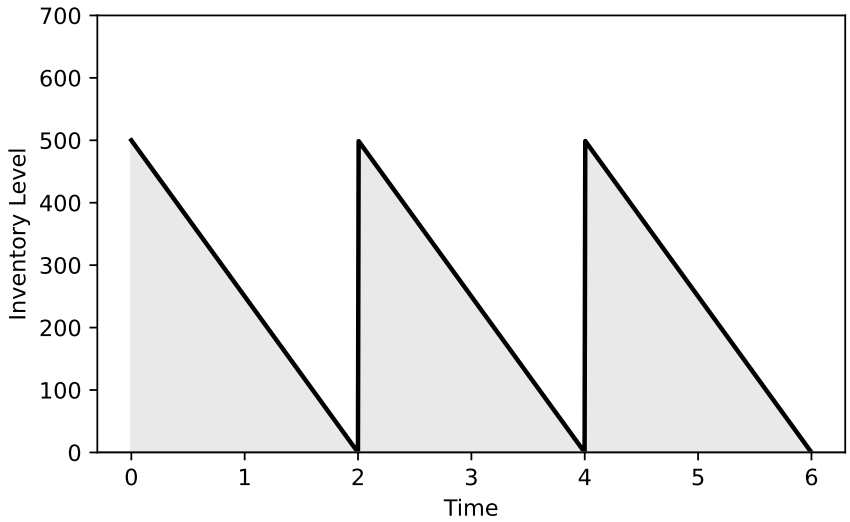
```
import matplotlib.pyplot as plt
import numpy as np

t = np.linspace(0, 30, 1000)
inventory = np.full_like(t, 50)

# Plotting the inventory level
plt.fill_between(t, inventory, color="lightgray", alpha=0.5, label="Inventory Level")
plt.plot(t, inventory, label="Inventory Level", color="black", linewidth=2)
plt.xlabel("Time")
plt.ylabel("Inventory Level")
plt.axhline(0, color="gray", linewidth=1)
plt.tight_layout()
plt.show()
```



一般的に、在庫量が定数ではなく、時間とともに変化する。例えば、発注量を 500 とし、6 日間の在庫量を考える。

```
# Parameters
d = 250  # Demand rate
Q = 500  # Order quantity
T = Q / d  # Cycle length
```

```
t = np.linspace(0, 2.999 * T, 1000)

# Inventory level over time
inventory = np.maximum(0, Q - (d * t) % Q)

# Plotting the inventory level
plt.fill_between(t, inventory, color="lightgray", alpha=0.5, label="Inventory Level
plt.plot(t, inventory, label="Inventory Level", color="black", linewidth=2)
plt.xlabel("Time")
plt.ylabel("Inventory Level")
plt.axhline(0, color="gray", linewidth=1)
plt.ylim(bottom=0, top=Q + 200)
plt.tight_layout()
plt.show()
```



保管費用は次のように計算される。

$$\frac{2 \times 500}{2} \times 3 \times h$$

第2章　経済的発注量

経済的発注量（EOQ: Economic Order Quantity）モデルは、最も基本的な在庫管理モデルの一つである。Harris (1990) このモデルを最初に提案した。

EOQ モデルは、単位時間あたりの需要量は決定論的で、一定であると仮定する。すなわち、需要量は事前に分かっており、時間とともに変化しない。単位時間あたりの需要量は需要率（demand rate）と呼ばれ、記号 d で表される。リードタイムは 0 とし、発注から納品までの時間はないと仮定する。一回の発注量を Q とし、一定であるとする。欠品は許せないとする。全ての需要は満されなければならない。また、EOQ モデルでは、在庫量は連続的に観測され、いつでも発注が可能であるとする。

在庫に関わる費用は、発注費用 K、保管費用 h と、購入費用（購入単価を c と表す）がある。

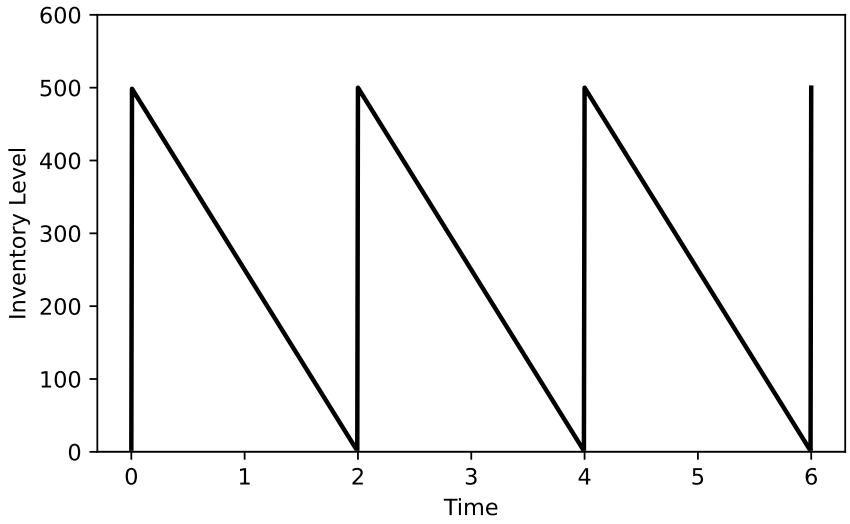
EOQ モデルの最適解は次の二つの性質を持つ (Snyder と Shen 2019)：

1. Zero-inventory ordering (ZIO). 在庫量が 0 のときに発注を行う。リードタイムは 0 であるため、在庫量が 0 でないときに発注すると、保管費用が発生する。
2. Constant order sizes. 発注量は一定である。需要率 d が一定であり、在庫量が 0 のときに発注を行うため、最適発注量も一定である。

以上の性質から、在庫量の時間的変化は下図のようになる。

```
import matplotlib.pyplot as plt
import numpy as np

# Parameters
d = 250  # Demand rate
Q = 500  # Order quantity
T = Q / d  # Cycle length
t = np.linspace(0, 3 * T, 1000)  # Time from 0 to 3 cycles

# Inventory level over time
```

```
inventory = np.maximum(0, Q - (d * t) % Q)
inventory[0] = 0

# Plotting the inventory level
plt.plot(t, inventory, label="Inventory Level", color="black", linewidth=2)
plt.xlabel("Time")
plt.ylabel("Inventory Level")
plt.axhline(0, color="gray", linewidth=1)
plt.ylim(bottom=0, top=Q + 100)
plt.tight_layout()
plt.show()
```



発注の間隔を**サイクル** (cycle) と呼び、サイクル期間は

$$T = \frac{Q}{d}$$

で与えられる。

例 2.1. A 社は、毎月 250 個の需要がある商品を取り扱っている。一回の発注量は 500 個とし、サイクル期間は

$$T = \frac{500}{250} = 2\text{ヶ月}$$

となる。

2.1 コスト関数

ここでは、1 サイクルあたりのコストを考える。

発注費用：発注は 1 回だけ行うため、発注費用は K である。

購入費用：Q 個の商品を単価 c で購入するため、購入費用は cQ である。

保管費用：在庫量はサイクル期間 T の間に Q 個から 0 個まで減少するため、平均在庫量は $\frac{Q}{2}$ である。したがって、平均保管費用は $\frac{hQ}{2}$ である。サイクル期間 T は $\frac{Q}{d}$ であるため、1 サイクルあたりの保管費用は

$$\frac{hQ}{2} \cdot T = \frac{hQ^2}{2d}$$

となる。

以上より、1 サイクルあたりのコストは次のように表される。

$$K + cQ + \frac{hQ^2}{2d}$$

平均コストは、これをサイクル期間 T で割ったものとして定義される。したがって、平均コスト $g(Q)$ は次のように表される。

$$\begin{aligned} g(Q) &= \frac{1}{T}\left(K + cQ + \frac{hQ^2}{2d}\right) \\ &= \frac{d}{Q}\left(K + cQ + \frac{hQ^2}{2d}\right) \\ &= \frac{Kd}{Q} + cd + \frac{hQ}{2} \end{aligned}$$

以上より、平均コストは発注量 Q の関数として次のように表される。

$$g(Q) = \frac{Kd}{Q} + cd + \frac{hQ}{2}$$

2.2 最適発注量

EOQ モデルの目的は、平均コスト $g(Q)$ を最小化する発注量 Q を求めることである。

平均コストの導関数 $g'(Q)$ が 0 となる点を求めることで、最適発注量 Q^* を求めることができる。

$$g'(Q) = -\frac{Kd}{Q^2} + \frac{h}{2} = 0$$

これを解くと、最適発注量

$$Q^* = \sqrt{\frac{2Kd}{h}}$$

を得る。これを EOQ 公式（EOQ formula）と呼ぶ。Q^* を経済的発注量と呼ぶ（経済的は最適という意味である）。

二階導関数 $g''(Q)$ を求めて、最適発注量が最小値を与えることを確認する。

$$g''(Q) = \frac{2Kd}{Q^3} > 0$$

$g''(Q) > 0$ であるため、Q^* は最小値を与える。

最適発注量 Q^* を次の定理にまとめる。

定理 2.1. *EOQ* モデルにおいて、最適発注量 Q^* は

$$Q^* = \sqrt{\frac{2Kd}{h}}$$

で与えられる。

Q^* を用いて、最適なサイクル期間 T^* を求めることができる。

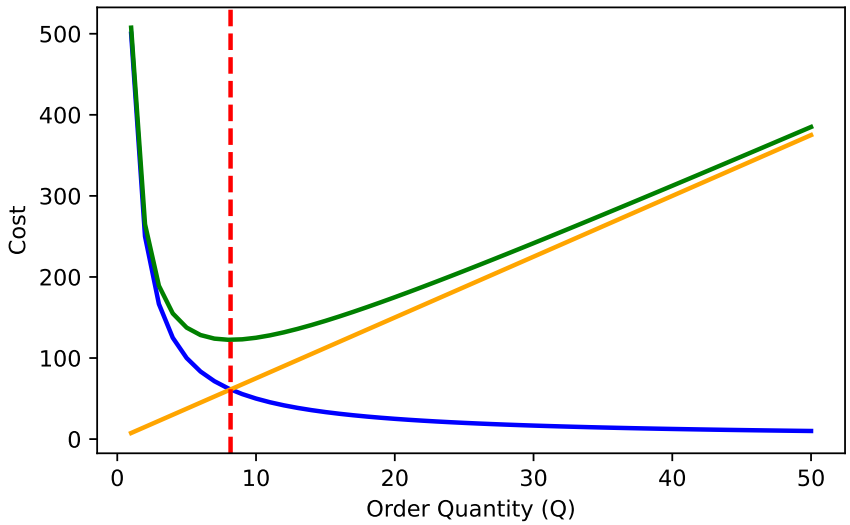
$$T^* = \frac{Q^*}{d} = \sqrt{\frac{2K}{hd}}$$

注釈 2.1. 以下の性質がわかる。

1. Q^* は c には依存しない。
2. h の増加に伴い、Q^* は減少する。保管費用が高い場合は、少量で高い頻度で発注することが望ましい。
3. K の増加に伴い、Q^* は増加する。発注費用が高い場合は、多量で低い頻度で発注することが望ましい。

次の図は、発注コスト、保管コスト、平均コストの関係を示している。購入単価を $c = 0$ とする。

```
# Parameters
K = 500  # Order cost
h = 15  # Holding cost
c = 0  # Purchase cost
Q = np.linspace(1, 50, 50)

# Average cost function
g = (K / Q) + c + (h * Q / 2)

# Plotting the costs vs order quantity
plt.plot(Q, K / Q, label="Order Cost", color="blue", linewidth=2)
plt.plot(Q, c + (h * Q / 2), label="Holding Cost", color="orange", linewidth=2)
plt.plot(Q, g, label="Average Cost", color="green", linewidth=2)
plt.axvline(
    x=np.sqrt(2 * K / h),
    color="red",
    linestyle="--",
    label="Optimal Order Quantity",
    linewidth=2,
)
plt.xlabel("Order Quantity (Q)")
plt.ylabel("Cost")
plt.tight_layout()
plt.show()
```



平均コストが最小となる発注量 Q^* は、発注コストと保管コストの交差点である。すなわち、発注コストと保管コストを等しくする発注量は最適な発注量 Q^* である。この性質は以下の式からわかる。

$$\frac{Kd}{Q^*} = \frac{hQ^*}{2} \Longrightarrow Q^* = \sqrt{\frac{2Kd}{h}}$$

また、この図からもわかるように、Q の増加に伴い、平均発注コストは減少し、平均保管コストは増加する。逆もまた然りである。

例 2.2. ある電気量販店では、毎月 250 台の PC が販売されている。発注費用は 5000 円、保管費用は 1 台あたり月 150 円、購入単価は 10 万円とする。このとき、最適発注量 Q^* は次のように求められる。

$$Q^* = \sqrt{\frac{2 \cdot 5000 \cdot 250}{150}}$$

最適発注量 Q^* を次で計算する。

```python
def eoq(K, d, h):
    """
    Calculate the Economic Order Quantity (EOQ).

    Parameters:
    K (float): Order cost
    d (float): Demand rate
    h (float): Holding cost

    Returns:
    float: Optimal order quantity Q*
    """
    return np.sqrt(2 * K * d / h)


if __name__ == "__main__":
    K = 5000  # Order cost
    d = 250  # Demand rate (units per month)
    h = 150  # Holding cost (per unit per month)

    Q_star = eoq(K, d, h)
    print(f"Optimal Order Quantity (Q*): {Q_star:.2f}")
```

```
Optimal Order Quantity (Q*): 129.10
```

PC の場合は、注文量が整数である必要があるため、$g(129)$ と $g(130)$ を比較して最適発注量を決定する。

2.3 リードタイム

EOQ モデルでは、リードタイムは 0 と仮定している。リードタイムが $L > 0$ の場合も、最適発注量 Q^* も変換せず、L 期間前に Q^* を発注すればよい。

ここでは、r を発注点(reorder point)とする。在庫量が r になったときに発注を行う。リードタイム L の間に需要が dL 個あるため、発注点は次のように表される。

$$r = dL$$

例 2.3. 上の例で、リードタイムが一週間とし、一か月を 4 週間とすると、リードタイムは $L = 1/4$ となる。したがって、発注点は次のように求められる。

$$r = dL = 250 \times \frac{1}{4} = 62.5$$

PC の在庫量が 63 台になったときに発注を行う。

2.4 他の EOQ モデル

- バックオーダーを考慮した EOQ モデル
- 数量割引(quantity discount)を考慮した EOQ モデル
  - 総量割引(all-units discount)
  - 増分割引(incremental discount)

第3章　Wagner-Whitin モデル

記号	意味
$\mathcal{T}$	期間の集合、$\mathcal{T} = \{1, 2, \dots, T\}$
K	1 回あたりの発注費用
h	単位あたりの保管費用
d_t	第 t 期の需要量
q_t	第 t 期の発注量
x_t	第 t 期の在庫量
y_t	第 t 期に発注する場合は 1、しない場合は 0
M	非負の大きな数

Wagner-Whitin モデルは次のように定式化される。

$$\text{minimize} \quad \sum_{t=1}^{T}(Ky_t + hx_t) \tag{3.1}$$

$$\text{subject to} \quad x_t = x_{t-1} + q_t - d_t \qquad \forall t \in \mathcal{T} \tag{3.2}$$

$$q_t \geq 0 \qquad \forall t \in \mathcal{T} \tag{3.3}$$

$$q_t \leq My_t \qquad \forall t \in \mathcal{T} \tag{3.4}$$

$$x_t \geq 0 \qquad \forall t \in \mathcal{T} \tag{3.5}$$

$$y_t \in \{0, 1\} \qquad \forall t \in \mathcal{T} \tag{3.6}$$

第4章　新聞売り子問題

新聞売り子問題(Newsvendor Problem)は、古典的な確率的在庫モデルの一つである。新聞は次の日には売れなくなるため、新聞売り子問題は最も単純な perishable 在庫モデルとして知られている。

新聞売り子が新聞を仕入れ、販売する問題を考える。新聞 1 部の仕入れ価格を c、販売価格を r、残存価額を v、欠品費用を p、保管費用を h とする。ここで、以下の条件を満たすとする。

- $r > c$. 販売価格は仕入れ価格より高い。
- $r > v$. 販売価格は残存価値より高い。

また、初期在庫は 0 であると仮定する。

新聞の需要 D は正規分布 $N(\mu, \sigma^2)$ に従い、μ と σ は既知である。新聞売り子はどれだけの新聞を発注すればよいかという問題である。

4.1　定式化

ある日、新聞売り子が S 部の新聞を仕入れ、需要 D が d であったとする。このとき、新聞売り子の利益は以下のように表される。

$$\pi(S, d) = r\min\{d, S\} - cS + v\max\{0, S-d\} - h\max\{0, S-d\} - p\max\{0, d-S\}$$

第一項は販売利益、第二項は仕入れコスト、第三項は残存価値、第四項は保管コスト、第五項は欠品コストである。

> **ノート**
>
> A は要素の間に順序が定義された集合とし、$\min A$ は、A の全ての要素の中で最小のものを表す。
> 例えば、$\min\{1, 2, 3\} = 1$ である。

以下は、$\max\{0, x\} = x^+$ を用いて書き換えた形である。整理すると、利益は以下のように表される。

$$\pi(S, d) = r \min\{d, S\} - cS + (v - h)(S - d)^+ - p(d - S)^+$$

第一項を以下のように書き換えることができる。

$$r \min\{d, S\} = rd - r(d - S)^+$$

> **ノート**
>
> - $d < S$ の場合、$r \min\{d, S\} = rd$ となる。
> - $d >= S$ の場合、$r \min\{d, S\} = rd - r(d - S) = rS$ となる。

したがって、利益は以下のように書き換えられる。

$$\pi(S, d) = rd - cS + (v - h)(S - d)^+ - (p + r)(d - S)^+$$

利益の最大化は、コストの最小化に帰着される。したがって、コスト関数 $g(S, d) = -\pi(S, d)$ は以下のように表される。

$$g(S, d) = cS - rd + (h - v)(S - d)^+ + (p + r)(d - S)^+$$

D が確率変数であるため、コストの期待値 $g(S)$ は以下のように表される。

$$\begin{aligned}
g(S) &= \mathbb{E}[g(S, D)] && (4.1)\\
&= \int_0^\infty g(S, d) f_D(d) dd && (4.2)\\
&= cS - r\mathbb{E}[D] + (h - v)\mathbb{E}[(S - D)^+] + (p + r)\mathbb{E}[(D - S)^+] && (4.3)\\
&= cS - r\mu + (h - v) \int_0^\infty (S - d)^+ f_D(d) dd + (p + r) \int_0^\infty (d - S)^+ f_D(d) dd && (4.4)\\
&= cS - r\mu + (h - v) \int_0^S (S - d) f_D(d) dd + (p + r) \int_S^\infty (d - S) f_D(d) dd && (4.5)
\end{aligned}$$

4.2 最適化

$g(S)$ の 1 階微分は以下のように求める。

$$\frac{dg(S)}{dS} = c + (h-v)F_D(S) - (p+r)(1-F_D(S)) \tag{4.6}$$

$$\tag{4.7}$$

よって、$dg(S)/dS = 0$ から、

$$c + (h-v)F_D(S) - (p+r)(1-F_D(S)) = 0 \tag{4.8}$$

$$F_D(S) = \frac{p+r-c}{h+p+r-v} \tag{4.9}$$

になる。2 階微分は

$$\frac{d^2g(S)}{dS^2} = (h-v)f_D(S) + (p+r)f_D(S) \tag{4.10}$$

$$= (h-v+p+r)f_D(S) \tag{4.11}$$

である。したがって、コスト関数 $g(S)$ は凸関数であり、1 階微分が 0 になる点は最小値を与える。

コスト関数 $g(S)$ を最小化するための最適発注量 S^* は

$$S^* = F_D^{-1}\left(\frac{p+r-c}{h+p+r-v}\right)$$

となる。ここで、F_D^{-1} は需要 D の累積分布関数の逆関数である。

4.3 臨界率

新聞売り子問題において、より一般的に、在庫超過費用(overage cost)と在庫不足費用(underage cost)を考慮する。

$$C_o = h + c - v, \quad C_u = p + r - c$$

在庫超過費用 C_o は、在庫が余ったときのコストである。1 部の在庫超過に対し、保管コストと仕入れコストが発生するが、残存価額が得られないため、$C_o = h + c - v$ となる。

在庫不足費用 C_u は、在庫が不足したときのコストである。1 部の在庫不足に対し、欠品コスト p と失われた販売機会の利益 $r - c$ が発生するため、$C_u = p + r - c$ となる。

従って、

$$\begin{aligned} S^* &= F_D^{-1}\left(\frac{p + r - c}{h + p + r - v}\right) \\ &= F_D^{-1}\left(\frac{C_u}{C_o + C_u}\right) \end{aligned}$$

が得られる。

定理 4.1. 新聞売り子問題における最適発注量 S^* は、

$$S^* = F_D^{-1}\left(\frac{C_u}{C_o + C_u}\right)$$

で与えられる。

$C_u/(C_o + C_u)$ は**臨界率**（critical ratio）と呼ばれる。臨界率は、在庫不足費用と在庫超過費用の比率を表す。

ここで、$F_D(S) = P(D \leq S)$ は欠品が発生しない確率を表す。この確率のことは**サービスレベル**（service level）と呼ぶ。したがって、最適発注量 S^* を用いることは、サービスレベルを $C_u/(C_o + C_u)$ に等しくすることを意味する。$1 - C_u/(C_o + C_u) = C_o/(C_o + C_u)$ の確率で欠品が発生することが最適である。

注釈 4.1.

- 在庫不足費用 C_u の増加に伴い、サービスレベルと最適発注量 S^* は増加する。
- 在庫超過費用 C_o の増加に伴い、サービスレベルと最適発注量 S^* は減少する。

例 4.1. 需要 D が正規分布 $N(100, 25)$ に従う新聞売り子問題を考える。在庫超過費用が $C_o = 10$、在庫不足費用が $C_u = 40$ のとき、最適発注量 S^* を求める。

$$S^* = F_D^{-1}\left(\frac{C_u}{C_o + C_u}\right) = F_D^{-1}\left(\frac{40}{10+40}\right) = F_D^{-1}(0.8)$$

ここで、F_D^{-1} は需要 D の累積分布関数の逆関数である。Python では、SciPy ライブラリの `ppf()` 関数を用いて、逆関数を求めることができる。

```python
from scipy.stats import norm

# 正規分布のパラメータ
mu = 100
sigma = 5

# 在庫超過費用と在庫不足費用
C_o = 10
C_u = 40

# 臨界率
critical_ratio = C_u / (C_o + C_u)

# 最適発注量
S_star = norm.ppf(critical_ratio, loc=mu, scale=sigma)
print(f"Optimal order quantity S*: {S_star:.2f}")
```

```
Optimal order quantity S*: 104.21
```

4.4 初期在庫を考慮した新聞売り子問題

新聞売り子の初期在庫を I とする。$I \leq S^*$ の場合、最適発注量は $S^* - I$ となる。すなわち、在庫量を S^* にすればよい。

また、$g(S)$ は凸関数であるため、$I > S^*$ の場合、何も発注しないことが最適である。

したがって、最適発注量は

$$Q = \begin{cases} S^* - I, & \text{if } I \leq S^*, \\ 0, & \text{if } I > S^*. \end{cases}$$

となる。

このような発注方式を **Base Stock Policy** (BSP) と呼ぶ。BSP は、各期間の在庫量を観測し、在庫量が S^* に引き上げられるように発注する方式である。新聞売り子問題において、BSP は最適な方策であると知られている。

4.5　発注費用を考慮した新聞売り子問題

Scarf (1959) の論文では、(s, S) 方策が発注費用を考慮した複数期間の新聞売り子問題において最適であることを示している。

第5章　安全在庫

これまで紹介した在庫モデルは、需要が決定論的であると仮定していた。ここからは、需要が確率的であると仮定した在庫モデルを紹介する。

5.1　在庫方策

確率的在庫モデルにおいて、一つ重要な概念は**在庫方策**（inventory policy）である。在庫方策は、在庫の状況に応じて、在庫管理のルールを定めるものである。代表的な在庫方策を以下に示す。

1. (r, Q) 方策：在庫量を連続的に観測し、在庫量が発注点 r 以下になったときに発注量 Q を発注する方式である。**発注点方式**とも呼ばれる。
2. BSP 方策(Base Stock Policy)：在庫量を定期的に観測し、在庫量が基準在庫 S 以下になったときに、在庫量を S まで補充する方式である。**定期発注方式**とも呼ばれる。
3. (s, S) 方策：在庫量を定期的に観測し、在庫量が発注点 s 以下になったときに、在庫量を補充点 S まで補充する方式である。

一部の確率的在庫モデルにに対し、これらの在庫方策は**最適**であることが知られている。その場合、在庫方策が持つパラメータを最適化することで、在庫の期待コストを最小化することができる。

5.2　問題設定

需要 D がある確率分布に従うと仮定する。リードタイムを L とし、既知の定数とする。発注費用を K、単位あたりの保管費用を h とする。在庫量が連続的に観測され、いつでも発注が可能であるとする**連続観測**の場合を考える。

(r, Q) 方策が用いられるとする。在庫量が発注点 r 以下になったときに、発注量 Q を発注する。この場合、発注点 r と発注量 Q を決定変数とし、在庫の**期待コスト**（expected cost）を最小化することを目的とする。

> **ノート**
>
> この問題の定式化および厳密解法は、ここでは説明しない。Snyder と Shen (2019) の「Fundamentals of Supply Chain Theory」などの文献を参照されたい。以下は (r, Q) の近似解法を紹介する。

5.3　近似解法

以下では、単位期間あたりの需要を D とし、D は正規分布 $N(\mu, \sigma^2)$ に従うと仮定する。ここで、μ は平均需要、σ は需要の標準偏差である。

5.3.1　発注量 Q

D の平均 μ を EOQ モデルの需要率とみなすと、発注量 Q は次のように求めることができる (Camm ほか 2022)。

$$Q = \sqrt{\frac{2K\mu}{h}}$$

また、欠品費用も考慮する場合、バックオーダーを考慮した EOQ モデルを用いて、発注量 Q は次のように求めることができる (Hillier と Lieberman 2025)。

$$Q = \sqrt{\frac{2K\mu}{h}} \sqrt{\frac{p+h}{p}}$$

p は単位あたりの欠品費用である。

得られた発注量 Q は、最適解ではなく、近似解であることに注意されたい。

5.3.2　発注点 r

リードタイム期間中に発生する需要は $D_L \sim N(\mu_L, \sigma_L^2)$ とし、正規分布の再生性により、

$$\mu_L = \mu L, \quad \sigma_L^2 = \sigma^2 L$$

になる。すなわち、リードタイム期間中の平均需要は $\mu_L = \mu L$、標準偏差は $\sigma_L = \sigma\sqrt{L}$ である。

発注点 r を決めるためには、**サービスレベル**（service level）を考える。ここでは、サービスレベルを、リードタイム期間中に需要を満たす確率と定義する。サービスレベルを α とし、$0 < \alpha < 1$ とする。

与えられたサービスレベル α に対して、D_L が発注点 r 以下になる確率（欠品が発生しない確率、つまり、サービスレベル）が α になるように発注点 r を決定する。

$$P(D_L \leq r) = \alpha$$

もし、発注点 $r = \mu_L$ とすると、$P(D_L \leq \mu_L) = 0.5$ となる。すなわち、50% の確率で欠品が発生することになる。

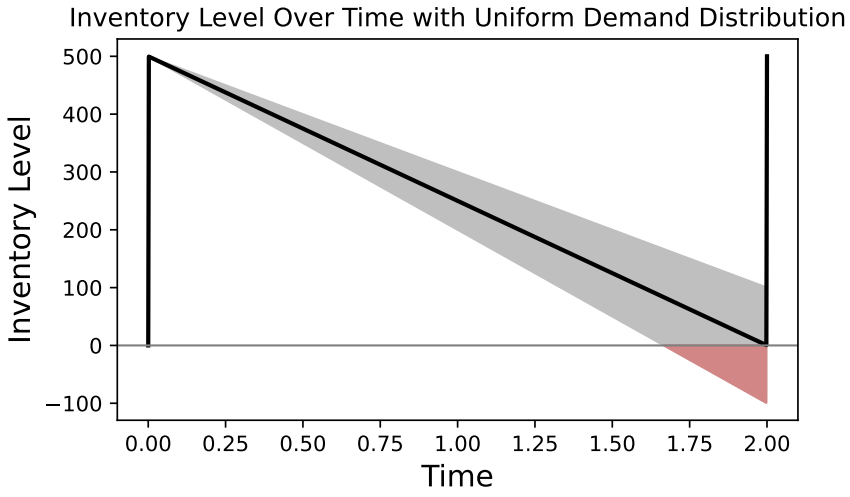
> **ノート**
>
> $$P(D_L \leq \mu_L) = P\left(\frac{D_L - \mu_L}{\sigma_L} \leq 0\right) = \Phi(0) = 0.5$$

したがって、サービスレベル $\alpha > 0.5$ の場合、発注点 r は平均需要 μ_L より大きくなる必要がある。それを**安全在庫**（safety stock）と呼び、s と表す。

例 5.1. 単位期間あたりの需要 D が連続一様分布 $U(200, 300)$ に従うと仮定する。平均需要は $\mu = 250$ である。一回の発注量を $Q = 500$ とする。次の図は、在庫量の時間的変化を示す。灰色の領域は、需要の範囲を示す。赤い領域は、在庫量が 0 以下になったときの欠品を示す。黒い線は平均需要に基づく在庫量の変化を示す。

```
import scipy.stats as stats
import numpy as np
import matplotlib.pyplot as plt

# Parameters
d_mean = 250  # Mean demand rate
d_max = 300   # Max demand rate
d_min = 200   # Min demand rate
Q = 500       # Order quantity
T = Q / d_mean  # Average cycle length

# Simulate over multiple cycles to show repeated pattern
n_cycles = 1
t = np.linspace(0, n_cycles * T, 1000)
```

```
# Inventory levels: linear depletion over time
inventory_mean = Q - d_mean * (t % T)
inventory_mean[0] = 0
inventory_max = Q - d_max * (t % T)
inventory_min = Q - d_min * (t % T)

# Plotting
plt.plot(t, inventory_mean, label="Mean Demand", color="black", linewidth=2)
plt.fill_between(t, inventory_min, inventory_max, color="gray", alpha=0.5, label="D

# Highlight when inventory drops below 0 (shortage)
plt.fill_between(t, inventory_max, 0, where=(inventory_max < 0), color="red", alpha

# Aesthetics
plt.axhline(0, color="gray", linewidth=1)
plt.xlabel("Time", fontsize=14)
plt.ylabel("Inventory Level", fontsize=14)
plt.title("Inventory Level Over Time with Uniform Demand Distribution")
plt.tight_layout()
plt.show()
```



よって、発注点 r は次のように表される。

$$r = \mu_L + s$$

従って、$P(D_L \leq r) = \alpha$ は次のように表される。

$$P(D_L \leq \mu_L + s) = \alpha$$

この式を変形すると、

$$P(D_L - \mu_L \leq s) = \alpha \tag{5.1}$$
$$P\left(\frac{D_L - \mu_L}{\sigma_L} \leq \frac{s}{\sigma_L}\right) = \alpha \tag{5.2}$$
$$\Phi\left(\frac{s}{\sigma_L}\right) = \alpha \tag{5.3}$$
$$\frac{s}{\sigma_L} = \Phi^{-1}(\alpha) \tag{5.4}$$
$$s = \sigma_L \Phi^{-1}(\alpha) \tag{5.5}$$
$$s = \sigma\sqrt{L}\Phi^{-1}(\alpha) \tag{5.6}$$

ここで、$\Phi(\cdot)$ は標準正規分布の累計分布関数であり、$\Phi^{-1}(\alpha)$ はその逆関数である。したがって、発注点 r は次のように表される。

$$r = \mu_L + s = \mu L + \sigma\sqrt{L}\Phi^{-1}(\alpha)$$

$\Phi^{-1}(\alpha)$ は標準正規分布表、Excel、Python などを用いて求めることができる。

例 5.2. リードタイム $L = 4$、平均需要 $\mu = 100$、需要の標準偏差 $\sigma = 20$、サービスレベル $\alpha = 0.95$ のとき、発注点 r と安全在庫 s を求める。

リードタイム期間中の平均需要と標準偏差は次のように計算される。

$$\mu_L = \mu L = 100 \cdot 4 = 400 \tag{5.7}$$
$$\sigma_L = \sigma\sqrt{L} = 20\sqrt{4} = 40 \tag{5.8}$$
$$\tag{5.9}$$

標準正規分布表から $\Phi^{-1}(0.95) \approx 1.64485$ を得る。これを用いて安全在庫 s と発注点 r を求める。

$$s = \sigma_L \Phi^{-1}(0.95) \approx 40 \cdot 1.64485 \approx 65.79 \tag{5.10}$$
$$r = \mu_L + s \approx 400 + 65.79 \approx 465.79 \tag{5.11}$$

したがって、発注点 r は約 465.79、必要な安全在庫 s は約 65.79 となる。

Python では、以下のように計算できる。

```
from scipy.stats import norm

L = 4
mu = 100
sigma = 20
alpha = 0.95

mu_L = mu * L
sigma_L = sigma * (L ** 0.5)

s = sigma_L * norm.ppf(alpha)
r = mu_L + s

print(f"reorder point: {r:.2f},  safety stock: {s:.2f}")
```

```
reorder point: 465.79,  safety stock: 65.79
```

第II部

多基準意思決定分析

第6章　階層分析法

第III部

プロジェクト管理

第7章　PERT/CPM

:::{code-cell} python :tags: [remove-input, remove-output] !pip install matplotlib numpy networkx !pip install pydot graphviz import numpy as np import matplotlib.pyplot as plt import networkx as nx from networkx.drawing.nx_pydot import graphviz_layout :::

- Camm et al. (2022) Chapter 9
- Taha (2016) Chapter 6
- Hillier and Lieberman (2020) Chapter 22
- Eiselt and Sandblom (2019) Chapter 8

PERT（Program Evaluation and Review Technique）**CPM**（Critical Path Method）

ある作業の開始前に完了しなければならない作業を**先行作業**（Immediate Predecessor）と呼ぶ。

プロジェクトは、次の要素で構成される。

- 作業（Activity）：プロジェクトを構成する仕事。活動、アクティビティとも呼ばれる。
- 先行関係（Precedence Relationship）：それぞれの作業の先行作業を定義する関係。
- 作業時間：各作業に必要な時間。通常、確率変数として与えられる。

7.1　プロジェクト・ネットワーク

プロジェクトを表現するネットワークを**プロジェクト・ネットワーク**（Project Network）と呼ぶ。プロジェクト・ネットワークには、**AOA**（Activity on Arrow）や **AON**（Activity on Node）という 2 種類の表現方法がある。

AOA では、作業を辺で表現し、先行関係をノードで表現する。AON では、作業を頂点（node）で表現し、先行関係を辺（edge）で表現する。日本の教科書では AOA が一般的であるが、AON のほうが理解と作成が容易であるため、海外の教科書では AON が一般的で、実務でも AON がよく使われる (Camm

ほか 2022; Hillier と Lieberman 2025; Eiselt と Sandblom 2022)。これ以降の説明では、AON を用いる。

:label: example:pert-1

学生の田中さんと佐藤さんが協力し、ある授業のレポートを作成することになった。このレポートを作成するためには、下の表に示すように、いくつかの作業を行う必要がある。

作業	作業内容	先行作業	時間（日）
A	課題の理解	-	2
B	データ収集	A	3
C	データ分析	B	4
D	文献調査	A	2
E	レポート作成	C, D	5

このプロジェクトを表現するプロジェクト・ネットワークは次の図のようになる。

:::{code-cell} python :tags: [remove-input]

class ProjectNetwork: def **init**(self, tasks): self.tasks = tasks self.G = nx.DiGraph() self._create_graph()

```
def _create_graph(self):
    # Add tasks
    for task_id, task in self.tasks.items():
        self.G.add_node(
            task_id, label=f"{task_id}\n({task['duration']})"
        )

    # Add Start and Finish nodes
    self.G.add_node("Start", label="Start")
    self.G.add_node("Finish", label="Finish")

    # Add edges for precedence
    for task_id, task in self.tasks.items():
        if not task["predecessors"]:  # no predecessors
→ connect from Start
            self.G.add_edge("Start", task_id)
        else:
```

```
            for pred in task["predecessors"]:
                self.G.add_edge(pred, task_id)

    # Add Finish connections (tasks with no successors → to Finish)
    for task_id in self.tasks:
        if self.G.out_degree(task_id) == 0:  # no outgoing edges
            self.G.add_edge(task_id, "Finish")

def draw(self):
    # Graphviz layout (top-to-bottom)
    pos = graphviz_layout(self.G, prog="dot")

    # Draw the graph
    plt.figure(figsize=(10, 10))
    nx.draw(
        self.G,
        pos,
        with_labels=False,
        node_size=1500,
        node_color="lightyellow",
        edgecolors="black",
        arrows=True,
        arrowsize=20,
    )
    nx.draw_networkx_labels(
        self.G, pos, labels=nx.get_node_attributes(self.G, "label"), font_size=10
    )

    plt.title("Project Network Diagram")
    plt.axis("off")
    plt.show()
```

tasks = { “A”: {“name”: “課題の理解”, “predecessors”: [], “duration”: 2}, “B”: {“name”: “データ収集”, “predecessors”: [“A”], “duration”: 3}, “C”: {“name”: “データ分析”, “predecessors”: [“B”], “duration”: 4}, “D”: {“name”: “文献調査”, “predecessors”: [“A”], “duration”: 2}, “E”: {“name”: “レポート作成”, “predecessors”: [“C”, “D”], “duration”: 5}, }

project_network = ProjectNetwork(tasks) project_network.draw()

グラフ理論において、**有向非巡回グラフ**（Directed Acyclic Graph, DAG）と

は、閉路を含まない有向グラフのことを指す。プロジェクト・ネットワークは辺の向きがあるため、有向グラフであり、作業の先行関係は循環しないため、閉路を含まない。したがって、プロジェクト・ネットワークは DAG である。

作業リストに示す作業に加えて、プロジェクト・ネットワークには、**開開始点**（Start Node）と**終了点**（Finish Node）が含まれる。開始点は、先行作業を持たない作業から接続される。終了点は、後続作業を持たない作業から接続される。上の例では、先行作業を持たない作業は A であり、後続作業を持たない作業は E であるため、開始点から A に接続され、E から終了点に接続されている。

:label: example:pert-2

以下の作業リストに基づいて、プロジェクト・ネットワークを作成せよ。

作業	先行作業	時間（日）
A	-	2
B	A	4
C	B	4
D	C	6
E	C	5
F	E	5
G	D	6
H	E, G	9
I	C	8
J	F, I	7
K	J	4
L	J	6
M	H	2
N	K, L	6

先行作業がない作業は A と B であり、後続作業がない作業は M と N であるため、開始点から A と B に接続され、M と N から終了点に接続される。

:::{code-cell} python :tags: [remove-input]

tasks = { “A”: {“name”: “ “,”predecessors”: [], “duration”: 2}, “B”: {“name”: “ “,”predecessors”: [“A”], “duration”: 4}, “C”: {“name”: “ “,”predecessors”: [“B”], “duration”: 4}, “D”: {“name”: “ “,”predecessors”: [“C”], “duration”: 6}, “E”: {“name”: “ “,”predecessors”: [“C”], “duration”: 5}, “F”: {“name”: “ “,”predecessors”: [“E”], “duration”: 5}, “G”: {“name”: “ “,”predecessors”:

[“D”], “duration”: 6}, “H”: {“name”: “ “,”predecessors”: [“E”, “G”], “duration”: 9}, “I”: {“name”: “ “,”predecessors”: [“C”], “duration”: 8}, “J”: {“name”: “ “,”predecessors”: [“F”, “I”], “duration”: 7}, “K”: {“name”: “ “,”predecessors”: [“J”], “duration”: 4}, “L”: {“name”: “ “,”predecessors”: [“J”], “duration”: 6}, “M”: {“name”: “ “,”predecessors”: [“H”], “duration”: 2}, “N”: {“name”: “ “,”predecessors”: [“K”, “L”], “duration”: 6}, }

project_network = ProjectNetwork(tasks) project_network.draw()

7.2 クリティカルパス

プロジェクト・ネットワークにおいて、開始点から終了点までの経路を**パス**（path）と呼ぶ。例えば、には

- Start → A → B → C → D → G → H → M → Finish
- Start → A → B → C → E → H → M → Finish

など、複数のパスが存在する。

Start → A → B → C → E → H → M → Finish では、各作業の所要時間を合計すると、$2 + 4 + 4 + 5 + 9 + 2 = 26$ 日となる。しかし、26 日間でこれらの作業を完了できるかというと、そうではない。なぜなら、例えば、作業 H の先行作業には作業 G と E の両方が含まれており、作業 H の開始時間は最も時間のかかる先行作業である G の完了時間に依存するからである。

したがって、プロジェクトの完了時間は、すべてのパスの中で最も長いパスの長さに依存する。この最も長いパスを**クリティカルパス**（Critical Path）と呼ぶ。クリティカルパス上の作業は、プロジェクト全体の完了時間に直接影響を与えるため、特に重要である。

クリティカルパスを求める問題は、DAG における最長経路問題に帰着される。

参考文献

Camm, Jeffrey, James Cochran, Michael Fry, Jeffrey Ohlmann, David Anderson, Dennis Sweeney, と Thomas Williams. 2022. *An introduction to management science: Quantitative approaches to decision making*. 16th 版. Florence, AL: South-Western College Publishing.

Eiselt, H A, と Carl-Louis Sandblom. 2022. *Operations research: A model-based approach*. Cham: Springer International Publishing.

Harris, Ford W. 1990.「How many parts to make at once」. *Oper. Res.* 38 (6): 947–50.

Hillier, Frederick, と Gerald Lieberman. 2025. *ISE introduction to operations research*. 11th 版. Columbus, OH: McGraw-Hill Education.

Scarf, Herbert. 1959.「The optimality of (S, s) policies in the dynamic inventory problem」.

Snyder, Lawrence V, と Zuo-Jun Max Shen. 2019. *Fundamentals of supply chain theory*. 2nd 版. Nashville, TN: John Wiley & Sons.

付録 A　微分積分学

A.1　1 変数関数の極値

関数 $f(x)$ が点 $x = a$ で極値をとるとき、$f'(a) = 0$ が成り立つ。

$f''(a) > 0$ のとき、$f(a)$ は極小値をとる。

$f''(a) < 0$ のとき、$f(a)$ は極大値をとる。

A.2　凸関数の性質

1 変数 2 階微分可能な関数 $f(x)$ が凸関数であることの必要十分条件は、$f''(x) \geq 0$ が成り立つことである。

$f''(x) > 0$ のとき、$f(x)$ は狭義凸関数である。

凸関数 $f(x)$ の極小値は、最小値である。

付録B　確率

離散型確率変数 X が特定の値 x をとる確率を

$$P(X = x) = p_X(x)$$

と表すとき、$p_X(x)$ を X の**確率質量関数** (PMF) という。

連続型確率変数 X がある区間 $[a, b]$ にある値をとる確率を

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx$$

と表す。$f_X(x)$ を X の**確率密度関数** (PDF) という。

確率変数 X の**累計分布関数**（CDF）は

$$F_X(x) = P(X \leq x) = \begin{cases} \sum_{k \leq x} p_X(k) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{x} f_X(t)dt & \text{if } X \text{ is continuous} \end{cases}$$

と表す。確率密度関数 $f_X(x)$ は累計分布関数 $F_X(x)$ の微分である。

$$f_X(x) = \frac{d}{dx} F_X(x)$$

B.1　正規分布

連続型確率変数 X は**正規分布**（normal distribution）に従うとき、$X \sim N(\mu, \sigma^2)$ と表す。ここで μ は平均、σ^2 は分散である。X の確率密度関数は

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

と表す。平均は $E[X] = \mu$、分散は $\mathrm{Var}(X) = \sigma^2$ である。

X が $N(\mu, \sigma^2)$ に従うとき、$Y = aX + b$ は、$N(a\mu + b, a^2\sigma^2)$ に従う。特に、$Z = \frac{X-\mu}{\sigma}$ は標準正規分布(standard normal distribution)に従う。すなわち、$Z \sim N(0, 1)$ である。

連続型確率変数 Y が標準正規分布に従うとき、Y の累計分布関数は

$$\Phi(y) = P(Y \leq y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-\frac{t^2}{2}} dt$$

と表す。**標準正規分布表**から、y の値に対する $\Phi(y)$ を調べることができる。

Python では、以下のように $\Phi(y)$ を計算できる。

```
from scipy.stats import norm
def phi(y):
    return norm.cdf(y)

phi(0)  # 0.5
```

また、$\Phi(y) = 0.95$ のときの y の値を求めるには、以下のようにする。

```
from scipy.stats import norm
def phi_inverse(p):
    return norm.ppf(p)

phi_inverse(0.95)  # 約 1.64485
```

与えられた $X \sim N(\mu, \sigma^2)$ の累計分布関数 $F_X(x)$ の値を求めるには、以下のように変換する。

$$P(X \leq x) = P\left(\frac{X-\mu}{\sigma} \leq \frac{x-\mu}{\sigma}\right) \tag{B.1}$$

$$\tag{B.2}$$

$$= P\left(Y \leq \frac{x-\mu}{\sigma}\right) \tag{B.3}$$

$$\tag{B.4}$$

$$= \Phi\left(\frac{x-\mu}{\sigma}\right) \tag{B.5}$$

正規分布は**再生性** (reproductive property)を持つ。すなわち、$X_1, X_2, \ldots, X_n$ が独立に $N(\mu_i, \sigma_i^2)$ に従うとき、$Y = \sum_{i=1}^{n} a_i X_i$ は $N\left(\sum_{i=1}^{n} a_i \mu_i, \sum_{i=1}^{n} a_i^2 \sigma_i^2\right)$ に従う。