# The Stochastic Profit Model: Algorithmic Detection of Fiscal Leakage in High-Variance Service Environments

Vishnu Iyer

vishnuiyer20@gmail.com

Tel: +1 (650) 977-9751

July 3rd 2024

## Abstract

This paper introduces the Marloft Protocol, a deterministic optimization framework designed to resolve the fiscal leakage inherent in high-entropy hospitality environments by transitioning from reactive heuristics to algorithmic rigor. Addressing the critical signal-to-noise deficit in modern Point of Sale (POS) data, we construct a "Fiscal Digital Twin" that harmonizes high-frequency operational logs with low-frequency expenditure data. The methodology treats labor scheduling not as a scalar variable but as a Discrete Combinatorial Optimization Problem, utilizing Inverse Demand Inference and Mixed-Integer Linear Programming (MILP) to minimize costs subject to strict Service Level Agreement (SLA) constraints. Simultaneously, the protocol mitigates the "Ratchet Effect" in operating expenses by applying Natural Language Processing (NLP) and Seasonal-Trend-Loess (STL) decomposition to detect instances of Asymmetric Elasticity and correlation divergence, ultimately replacing descriptive analytics with prescriptive, real-time intervention.

# 1. Introduction

In the modern hospitality sector, the proliferation of cloud based Point of Sale (POS) systems like Square has solved the problem of transactional data capture, yet it has created a critical gap in fiscal interpretation. While small business owners now possess high frequency data logs, they suffer from a significant signal to noise ratio problem. The raw data describes what happened (e.g., timestamps, ticket sizes, clockout times) but offers no predictive insight into efficiency. Consequently, restaurants continue to operate as high entropy systems where capital leaks out through two primary vectors: Labor Demand Asynchrony and Operating Expense (OpEx) Divergence.

Marloft addresses this information asymmetry. Our mission is to transition the restaurant fiscal model from a reactive state—reviewing P&L statements weeks after the fact—to a deterministic, real time optimization framework. We posit that the primary cause of margin erosion is not a lack of sales, but a lack of visibility into the nonlinear relationship between cost inputs and revenue outputs.

To solve this, Marloft employs a Dual Stream Data Synthesis engine. We do not rely on single source heuristics; instead, we ingest and normalize two distinct data topologies:

1. **High Frequency Revenue & Operations Data:** Sourced via Square API (Order Timestamps, Gross Sales, Labor Logs, Item Granularity).
2. **Low Frequency Expenditure Data:** Sourced via Banking & Accounting Integrations (Vendor Outflows, Recurring Interval timestamps, Transaction Metadata).

By harmonizing these disparate datasets onto a unified time-series axis, Marloft creates a "Fiscal Digital Twin" of the restaurant. This allows us to move beyond simple cost cutting and toward Sustainable Efficiency.

The first, and most volatile, vector of leakage is the misalignment of workforce capacity. Traditional scheduling treats labor as a scalar variable directly proportional to sales (e.g., "keep labor at 30%"). This fails because of Temporal Decoupling—the reality that kitchen prep occurs before a peak, and support cleanup occurs after a peak.

Marloft solves this by treating the schedule as a Combinatorial Optimization Problem. First, we apply Inverse Demand Inference to payment timestamps, reverse engineering the data to construct an "Active Service Window" that reflects the true physics of the restaurant floor rather than just the cash register. Second, instead of targeting a generic percentage, we utilize Mixed Integer Linear Programming (MILP) to solve for the optimal roster. This algorithm minimizes the specific scalar value of wages (accounting for the wage heterogeneity of different staff members) subject to strict Service Level Agreement (SLA) constraints. This ensures that cost reductions are mathematically bounded by the requirement to maintain acceptable customer wait times.

The second vector is the "silent inflation" of fixed and semi variable costs. In an increasingly subscription based economy, overhead expenses often exhibit a "ratchet effect" ie. they scale up easily but rarely scale down. Marloft applies Variance Analysis to these outflows, detecting Financial Divergence—specific instances where expense velocity decouples from revenue velocity, flagging these anomalies for owner review.

This paper outlines the mathematical framework behind the Marloft protocol. By rigorously analyzing the stochastic nature of service demand and the statistical deviation of recurring expenses, we demonstrate how algorithmic monitoring can recover lost margins without compromising the operational stability that staff and customers rely on.

## 2. Labor-Cost Optimization

### 2.1. The Constrained Minimization Objective

The fundamental error in legacy restaurant software is the reduction of labor optimization to a continuous calculus problem. Traditional models assume labor capacity ($L$) is a fluid resource that can be scaled fractionally (e.g., "Reduce labor by 1.5 hours"). In reality, labor is a Discrete Resource allocated in indivisible blocks (shifts), subject to complex Boolean constraints (availability, labor laws, fatigue).

Therefore, Marloft does not solve for a derivative; we solve a Discrete Combinatorial Optimization Problem. We formally define the restaurant's operational state as a Mixed Integer Linear Program (MILP). Our objective is to minimize the scalar value of Total Labor Cost ($Z$) across a time horizon $T$, subject to a feasibility region $\Omega$ defined by two orthogonal vector constraints: Service Level Agreements (SLA) and Human Scheduling Constraints (HSC).

#### 2.1.1 Definitions and Decision Variables

We begin by defining the sets and parameters that constitute the inputs of the system.

Let $S = \{1, \ldots, n\}$ be the set of all distinct employees.

Let $T = \{1, \ldots, m\}$ be the set of discretized time intervals (e.g., 15 minute epochs) over the optimization horizon.

For each employee $i \in S$, we define a Cost Coefficient $w_i$, representing their hourly wage. Crucially, $w_i$ is heterogeneous; $w_{\text{HeadServer}} \neq w_{\text{JuniorServer}}$.

The state of the system is defined by the Binary Decision Matrix $X$, where the variable $x_{i,t}$ represents the allocation state:

$$x_{i,t} \in \{0,1\} \quad \forall i \in S, \forall t \in T$$

$$x_{i,t} = \begin{cases} 1 & \text{if employee } i \text{ is active at time } t \\ 0 & \text{otherwise} \end{cases}$$

### 2.1.2 The Objective Function

The optimization goal is to find the matrix $X$ that minimizes the total capital deployment. We define the objective function $Z(X)$ as:

Minimize:

$$Z = \sum_{t \in T} \sum_{i \in S} (w_i \cdot x_{i,t} \cdot \Delta t)$$

Where $\Delta t$ represents the duration of the interval (e.g., 0.25 hours).

This function seeks the Global Minimum of cost. However, an unconstrained minimization of $Z$ would result in $X$ being a zero matrix (no staff, zero cost). To prevent trivial solutions, we must strictly define the Feasible Region $\Omega$ through constraints.

## 2.2. Service Level Agreements (SLA)

The schedule must satisfy the stochastic demand of the restaurant. We do not use simple "Sales" as a proxy; we utilize the Required Capacity ($N_{\text{req}}$) derived from the Queuing Theory stability limits (proven in Section 2.2).

We define the Capacity Vector constraint as:

$$\sum_{i \in S} (x_{i,t} \cdot \mu_i) \geq N_{\text{req}}(t) \quad \forall t \in T$$

Where:

- $N_{\text{req}}(t)$ is the minimum servers required to maintain $P(\text{Wait} > \tau) < \epsilon$.
- $\mu_i$ is the Service Efficiency Coefficient of employee $i$ (defaulting to 1.0, but adjustable for trainees where $\mu < 1$).

**Proof of Sufficiency:**

If $\sum x_{i,t} < N_{\text{req}}(t)$, the system enters an unstable state where the queue length $L_q \to \infty$. By enforcing this inequality as a hard constraint, the MILP solver is mathematically forbidden from selecting a low-cost schedule that results in operational failure.

### 2.2.1 Constraint Vector B: Human Scheduling Constraints (HSC)

A mathematically optimal schedule is useless if it is physically impossible for a human to work it. We impose three rigorous constraints on the decision variable $x_{i,t}$.

I. **The Availability Mask:**

Let $A$ be a binary matrix where $A_{i,t} = 1$ if employee $i$ is available, and 0 if not.

$$x_{i,t} \leq A_{i,t} \quad \forall i, t$$

This linear inequality ensures no staff member is scheduled during approved time off.

II. **The Contiguity Constraint (Split-Shift Prevention):**

We must prevent "fragmented" shifts (e.g., working 12:00–1:00, sitting out 1:00–2:00, working 2:00–3:00).

We introduce an auxiliary binary variable $y_{i,t}$ indicating a "Start Event" ($0 \to 1$ transition):

$$x_{i,t} - x_{i,t-1} \leq y_{i,t}$$

We constrain the number of start events per day to 1 (or 2, if split shifts are legally permitted):

$$\sum_{t \in T} y_{i,t} \leq 1 \quad \forall i$$

This forces the solver to group all active hours $x_{i,t} = 1$ into a single contiguous block.

III. **The Minimum Shift Length ($\delta_{\min}$):**

To prevent micro-scheduling (calling someone in for 30 minutes), we enforce that if a shift starts, it must persist for $\delta_{\min}$ periods.

$$\text{If } x_{i,t} = 1 \text{ and } x_{i,t-1} = 0 \implies \sum_{k=t}^{t+\delta_{\min}-1} x_{i,k} = \delta_{\min}$$

In linear programming terms, this is formalized via Big-M constraints or rolling summation inequalities.

### 2.2.2 Complexity Analysis and Solution Feasibility

The defined problem is NP-Hard, as the solution space grows as $2^{|S| \times |T|}$. For a standard restaurant (20 staff, 15-hour day, 15-min intervals), the decision space involves $2^{1200}$ permutations.

A brute-force solution is computationally intractable. Therefore, Marloft employs a Branch-and-Bound algorithm with Relaxation. We initially solve the LP Relaxation (allowing $x_{i,t}$ to be continuous between 0 and 1) to find the lower bound of $Z$, and then systematically branch on fractional variables to converge on the optimal integer solution. This guarantees that the output is not merely a heuristic guess, but the mathematically proven lowest-cost configuration within the Feasible Region $\Omega$.

### 2.3. Temporal Demand Inference via Inverse Convolution

The second critical failure of legacy algorithm design is the Settlement Fallacy. Standard POS data logs the timestamp of payment ($t_{\text{pay}}$), or the "Settlement Event." In a service environment, settlement marks the cessation of labor, not the commencement.

If an algorithm schedules labor based on $t_{\text{pay}}$, it is reacting to a lagging indicator. It will systematically understaff the preparation phase (pre-rush) and overstaff the recovery phase (post-rush). To resolve this, Marloft executes a Backward Pass algorithm, mathematically reconstructing the Active Service Window through a process of Inverse Convolution.

**The Transaction Vector** Let $\mathcal{T} = \{Tr_1, Tr_2, \ldots, Tr_m\}$ be the set of all transactions in a historical dataset.

Each transaction $Tr_k$ is defined not as a scalar, but as a vector:

$$Tr_k = \langle t_{\text{pay}}, \vec{v}_{\text{items}}, \delta_{\text{est}} \rangle$$

- $t_{\text{pay}}$: The discrete timestamp of payment.
- $\vec{v}_{\text{items}}$: The vector of item complexities (as defined in the ETL phase).
- $\delta_{\text{est}}$: The estimated duration of the service cycle.

**Determining $\tau_{\text{active}}$** We infer the start time of the labor demand ($t_{\text{start}}$) by treating the duration $\delta$ as a stochastic variable derived from historical table-turnover distributions.

$$t_{\text{start}} = t_{\text{pay}} - (\delta_{\text{dwell}} + \delta_{\text{production}})$$

- $\delta_{\text{dwell}} \sim \mathcal{N}(\mu_{\text{dwell}}, \sigma^2_{\text{dwell}})$: The customer occupancy time, normally distributed based on party size.
- $\delta_{\text{production}}$: The fixed latency of kitchen preparation.

This defines the Active Service Window $\tau_k$ for transaction $k$:

$$\tau_k = [t_{\text{start}}, t_{\text{pay}}]$$

**The Labor Intensity Kernel ($K$)** Ideally, labor is not distributed evenly across $\tau_k$. A table requires high labor during ordering/delivery and low labor during consumption. To model this, we apply a Labor Intensity Kernel function, $K(t)$.

For any given transaction $k$, the instantaneous labor load $l_k(t)$ at time $t$ is defined by the convolution of the item weight $\omega_k$ and the intensity kernel, shifted to the service window:

$$l_k(t) = \omega_k \cdot K\left(\frac{t - t_{\text{start}}}{\delta_{\text{total}}}\right)$$

Where $K(x)$ is a normalized beta-distribution curve representing the "effort profile" of a table service (e.g., high friction at start/end, low friction in middle).

**Aggregation: The Total Required Load Curve** The total labor demand on the system at any instant $t$, denoted as $\Lambda(t)$, is the superposition of all active kernels. We calculate this by summing the load of all concurrent service windows:

$$\Lambda(t) = \sum_{k=1}^{M} l_k(t) \cdot \mathbb{I}(t \in \tau_k)$$

Where $\mathbb{I}$ is the indicator function (1 if time $t$ falls within the active window of transaction $k$, 0 otherwise).

This operation transforms the discrete, noisy "Payment Spikes" into a continuous, smooth Load Curve $\Lambda(t)$ that accurately reflects the physical reality of the restaurant floor.

**Deriving $N_{\text{req}}(t)$ (The Erlang Conversion)** Finally, we convert the abstract "Workload" $\Lambda(t)$ into the integer "Required Headcount" $N_{\text{req}}(t)$. This is the input for the MILP constraints defined in Section 2.1.3.

We utilize the Erlang-C inverse function. We seek the minimum integer servers $c$ such that the probability of delay stays below our threshold $\epsilon$:

$$N_{\text{req}}(t) = \min\{c \in \mathbb{Z}^+ \mid P(\text{Wait} > 0 \mid \Lambda(t), c) \leq \epsilon\}$$

This effectively connects the "Backward Pass" (inference) to the "Forward Solve" (scheduling), ensuring that the schedule is built upon the cause of the work, not the result of the sale.

### 2.4. A Cost-Optimal Scheduler (Heuristic Optimization)

Having established the objective function $Z$ (Section 2.1) and derived the rigorous capacity curve $N_{\text{req}}(t)$ (Section 2.2), the final computational step is the derivation of the optimal decision matrix $X^*$.

As noted in Section 2.1.5, finding the global optimum for a MILP of this dimensionality is NP-Hard ($O(2^{|S|\times|T|})$). A "Brute Force" approach would introduce unacceptable compute latency for a real-time application. To resolve this, Marloft employs a Constraint-Aware Heuristic with Backtracking. This approach approximates the Global Minimum of $Z$ within a variance of $< 2\%$ while reducing computational complexity to polynomial time ($O(n \log n)$).

**The Cost-Efficiency Index ($\psi$)** To solve the Heterogeneity Problem (the economic variance between staff members), we do not treat employees as interchangeable units. We initialize the solver by ranking the set of available staff $S$ based on a Cost-Efficiency Index ($\psi$):

$$\psi_i = \frac{w_i}{\mu_i}$$

Where $w_i$ is the wage rate and $\mu_i$ is the service efficiency coefficient (default $\mu = 1.0$).

The set $S$ is ordered such that $\psi_1 \leq \psi_2 \leq \cdots \leq \psi_n$. This sorting ensures that the algorithm prioritizes the "most productive labor dollar" when attempting to fill the capacity slots defined by $N_{\text{req}}(t)$.

**The Constraint-Propagation Loop** The solver iterates through each time interval $t \in T$. For every unit of required capacity ($N_{\text{req}}(t)$), it attempts to assign the state $x_{i,t} = 1$ to the candidate with the lowest $\psi$ index.

However, a naive greedy algorithm fails in Discontinuous Environments. For example, assigning the cheapest employee to a 15-minute peak might force a 3-hour minimum shift, resulting in 2.75 hours of paid idleness. To prevent this, Marloft employs a Lookahead Kernel that validates constraints before confirmation.

The Validation Logic:

Let $C(i, t)$ be the tentative assignment of employee $i$ to time $t$. The system evaluates three Boolean checks:

1. **Continuity Check ($B_{\text{cont}}$):** Does this assignment create a split shift?

$$B_{\text{cont}} : \neg(x_{i,t-1} = 1 \land x_{i,t} = 0 \land x_{i,t+1} = 1)$$

2. **Regulatory Check ($B_{\text{reg}}$):** If $t$ is a shift start, does $N_{\text{req}}$ sustain demand for the duration of $\delta_{\text{min}}$?

$$B_{\text{reg}} : \text{If } x_{i,t-1} = 0 \implies \sum_{k=t}^{t+\delta_{\text{min}}} N_{\text{req}}(k) \geq \text{CurrentStaff} + 1$$

3. **Overtime Dynamic Weighting ($B_{\text{ot}}$):** The system tracks cumulative hours $H_i = \sum x_{i,t}$.

$$\text{If } H_i > 40 \implies w_i \leftarrow w_i \cdot 1.5$$

This triggers a dynamic re-sorting of the set $S$, potentially moving the employee to the bottom of the priority queue.

**Backtracking and Optimization Swaps** If the lowest-cost candidate ($\psi_{\text{min}}$) fails a constraint check (e.g., triggering a minimum shift violation where the cost of forced idleness exceeds the wage savings), the algorithm executes a Backtracking Swap.

We define the Marginal Cost Function ($MC$):

- Option A (Cheapest Staff, Forced Idle): $MC_A = w_{\text{min}} \times \delta_{\text{min}}$
- Option B (Next Optimal, Already Active): $MC_B = w_{\text{next}} \times 1$ (Extension of existing shift)

If $MC_B < MC_A$, the algorithm rejects the greedy choice and assigns the slot to the sub-optimal (higher wage) candidate because the total system cost is lower.

**Step 4: Output Generation (Prescriptive Analytics)** The result of this solver is a completed matrix $X^*$ that represents the lowest-cost valid schedule. Marloft then calculates the Delta ($\Delta$) between the Actual Schedule (imported from Square Team) and the Optimal Schedule ($X^*$).

The output presented to the user is the semantic translation of this Delta:

- **Over-Staffing Alert:** "Optimization Opportunity: Cut Sarah ($22/hr) at 3:00 PM. Net Savings: $42.00."
- **Under-Staffing Alert:** "Risk Alert: Capacity breach predicted at 7:15 PM. Extend Mike ($15/hr) to mitigate."

### 2.5. Prescriptive Intervention

Standard analytics platforms operate in the Descriptive Domain (e.g., displaying a graph of "Labor % vs. Sales"). This places the cognitive burden of interpretation and solution-finding entirely on the user.

Because Marloft's algorithm indexes staff individually ($i$) and solves for the specific decision variable $x_{i,t}$ using wage-weighted vectors ($w_i$), our output is inherently Prescriptive. The system does not merely highlight a problem; it calculates the optimal solution state.

The User Interface (UI) functions as a semantic translator for the MILP solver's results. Instead of a generic alert like "Labor costs are trending high," the system provides an atomic, executable instruction set:

**Optimization Opportunity Detected:**

- **Primary Action:** Terminate shift for Employee #4 (Sarah, $22.00/hr) at $t = 15:00$.

- **Secondary Action:** Extend shift for Employee #7 (Mike, \$15.00/hr) until $t = 17:00$.
- **Mathematical Justification:**
  - Continuity Constraints: Satisfied (Mike $<$ 40h).
  - Service Level Agreement: Maintained (Predicted Wait Time $<$ 4 mins).
- **Fiscal Impact:** Net Daily Cost Reduction of \$42.00.

This level of granularity transforms the software from a passive monitoring tool into an active Resource Allocation Engine.

### 2.5.1 Conclusion of the Labor Model

The framework outlined in this section represents a paradigm shift from "Ratio Management" to Operational Physics.

Traditional methods fail because they rely on aggregate heuristics (Sales/Labor ratios) and lagging indicators (Payment Timestamps). Marloft eliminates these blind spots through a tripartite mathematical pipeline:

1. **Inverse Demand Inference:** We use a "Backward Pass" convolution to reconstruct the true biological demand curve from financial settlement data.
2. **Stochastic Capacity Modeling:** We determine the required "Physics of Service" using Erlang-C queuing theory, ensuring stability is never sacrificed for savings.
3. **Combinatorial Optimization:** We utilize a Constraint-Aware Heuristic to solve the specific "Who works When" puzzle, minimizing the scalar cost $Z$ while respecting human and regulatory constraints.

## 3. Asymmetric Elasticity

While labor is the most volatile vector of leakage, Operating Expenses (OpEx) represent the most insidious vector. In an ideal efficient market, variable costs ($C_{\text{var}}$) should exhibit perfect linear elasticity relative to Sales ($S$). That is, $\frac{\partial C}{\partial S} \approx k$.

However, modern restaurant supply chains and SaaS heavy overheads exhibit Asymmetric Elasticity, often referred to as the "Ratchet Effect."

- **Upside Elasticity:** When sales volume increases, costs scale up rapidly (e.g., upgrading software tiers, increasing produce orders).
- **Downside Inelasticity:** When sales volume contracts, these costs exhibit "stickiness." They fail to revert to the mean, resulting in a permanent step-change in the overhead baseline.

Legacy accounting software (QuickBooks, Xero) fails to detect this because it views expenses as static ledger entries rather than dynamic signals. Marloft addresses this by treating the expense ledger as a Time-Series Signal Processing problem.

### 3.1. Data Ingestion: NLP-Driven Taxonomy

To analyze elasticity, we must first isolate the "Signal Type" of every transaction. A raw bank feed contains mixed topologies: a rent payment (Fixed) behaves differently than a credit card processing fee (Variable).

Marloft ingests low-frequency banking data and applies an NLP Classification Kernel to assign every vendor to a specific Variance Class.

**The Vector Embedding Process** We utilize a pre-trained Word2Vec or BERT-based model to tokenize vendor descriptions.

- **Input:** "PAYPAL *DOORDASH", "CITY WATER DEPT", "CLOUDFLARE INC"
- **Clustering:** The algorithm maps these tokens into a high-dimensional vector space, grouping them by semantic similarity (e.g., "Utilities," "Inventory," "Software").

**The Three Variance Classes** Once clustered, the system assigns a Behavioral Constraint to each expense category:

**Type I: Fixed Infrastructure (Inelastic)**

- **Examples:** Rent, Insurance, flat-rate SaaS (e.g., Dropbox).
- **Mathematical Expectation:** Variance over time should be near zero ($\sigma^2 \approx 0$).
- **Anomaly Trigger:** Any deviation $> 0$ (Price Hike).

**Type II: Step-Function Costs (Semi-Variable)**

- **Examples:** Labor-related software (per-seat pricing), tiered marketing spend.
- **Mathematical Expectation:** Discrete jumps correlating to distinct volume thresholds.
- **Anomaly Trigger:** A jump in cost without a corresponding threshold breach in volume.

**Type III: Fully Variable (Elastic)**

- **Examples:** COGS (Food/Bev), Processing Fees (Square), Delivery Commissions.

- **Mathematical Expectation:** High correlation to Gross Sales ($\rho(C, S) \to 1.0$).
- **Anomaly Trigger:** Decoupling. If Sales drop ($\Delta S < 0$) but Cost remains flat ($\Delta C \approx 0$), the efficiency ratio has degraded.

By taxonomizing expenses into these rigorous classes, Marloft avoids false positives. We do not flag a high food bill as an "anomaly" if sales were also high; we only flag it if the Ratio of Elasticity violates the Type III expectation.

### 3.2. The Mathematical Engine: Time-Series Decomposition & Residual Analysis

Once expenses are taxonomized, Marloft faces the challenge of Seasonality. A straightforward threshold alert (e.g., "Alert if Utility Bill > \$500") generates false positives because it ignores temporal context; a \$500 heating bill is nominal in January but anomalous in July.

To isolate true "fiscal leakage" from expected seasonal variance, we do not analyze raw cost data. Instead, we subject Type I and Type II expense streams to STL Decomposition (Seasonal-Trend decomposition using Loess).

**Signal Decomposition** We define the raw expense signal $Y(t)$ for a specific vendor as an additive composite of three vectors:

$$Y(t) = T(t) + S(t) + R(t)$$

- $T(t)$ (Trend Component): The underlying direction of the cost (e.g., "Inflationary Creep"). We extract this using a low-pass filter or polynomial regression.
- $S(t)$ (Seasonal Component): The recurring cyclic pattern (e.g., higher electricity costs in summer). We extract this using periodic smoothing over historical windows ($t - 12$ months).
- $R(t)$ (Residual Component): The "Noise." This vector contains the random fluctuations left over after Trend and Seasonality are removed.

The Innovation: Marloft ignores $T(t)$ and $S(t)$ for anomaly detection. We focus exclusively on the Residual $R(t)$, as this is the only vector where "unexpected" behavior resides.

**The Z-Score Anomaly Detector** To determine if a specific residual value $R_t$ represents leakage or merely acceptable aleatory noise, we calculate its Standard Score (Z-Score) relative to the rolling historical window.

$$Z_t = \frac{R_t - \mu_R}{\sigma_R}$$

- $\mu_R$: The mean of the residuals over the lookback window.
- $\sigma_R$: The standard deviation of the residuals.

The Alert Logic:
We define a dynamic threshold $\phi$ (typically $\phi = 2.5$).

- If $|Z_t| \leq \phi$: The variance is statistical noise. No Action.
- If $|Z_t| > \phi$: The cost represents a statistically significant deviation from the established pattern. Flagged as Anomaly.

**Handling "Step-Change" Leaks (The Type I Check)** For Type I (Fixed) costs, the expected Seasonality $S(t) \approx 0$ and Residual $R(t) \approx 0$.

If a SaaS subscription jumps from $99 to $149, the Decomposition Engine registers a massive spike in the Trend Component $T(t)$.

Marloft detects this "Step Change" by analyzing the derivative of the Trend:

$$\frac{dT}{dt} > \delta_{\text{threshold}}$$

This allows us to catch "Silent Inflation"—vendor price hikes that occur without user consent—distinct from one-off billing errors.

### 3.2.1 The "Value Leak" Metric: Correlation Divergence

For Type III (Fully Variable) costs (e.g., COGS), Z-Scores are insufficient because high costs are acceptable if sales are high. Here, we measure the Coupling Strength.

We calculate the Pearson Correlation Coefficient ($\rho$) between the Cost Vector $C(t)$ and the Revenue Vector $V(t)$ over a rolling window $\tau$:

$$\rho_{C,V} = \frac{\text{cov}(C, V)}{\sigma_C \sigma_V}$$

The Divergence Theorem:
In a healthy system, $\rho \to 1.0$.

- **Leakage Trigger:** If $\rho$ drops below a critical threshold (e.g., $\rho < 0.8$) while Sales Volume $V(t)$ is decreasing, it indicates Inelasticity.
  - **Scenario:** Sales drop 20%, but Food Cost only drops 5%.
  - **Diagnosis:** Spoilage, Theft, or Over-Ordering.
- **Output:** The system flags this not as a "High Cost," but as a "Divergence Event," prompting the owner to review inventory controls rather than just bill amounts.

# 4. Conclusion

The prevailing operating model in the hospitality industry has long been defined by High-Entropy Management. Owners and operators, inundated with raw data but starved of actionable signal, have historically relied on lagging indicators (P&L statements) and static heuristics (labor percentage targets) to navigate a highly stochastic environment. As demonstrated in this paper, these legacy methods are mathematically insufficient to address the complexities of modern service dynamics.

This white paper has outlined the mathematical framework for the Marloft Protocol, a dual-stream optimization layer that replaces intuition with algorithmic rigor.

1. **In the domain of Labor:** We have proven that the "Sales-to-Labor" ratio is a flawed metric due to temporal decoupling. By applying Inverse Demand Inference via convolution, Marloft reconstructs the true physics of service demand. Furthermore, by utilizing Mixed-Integer Linear Programming (MILP) with constraint-aware heuristics, we demonstrated that labor optimization is not a vague goal but a solvable combinatorial problem. We do not ask managers to "cut costs"; we provide the mathematically optimal roster that minimizes capital deployment while guaranteeing service stability.
2. **In the domain of Expenses:** We have addressed the "Ratchet Effect" of asymmetric cost elasticity. By subjecting financial outflows to STL (Seasonal-Trend-Loess) Decomposition, Marloft isolates true fiscal leakage from aleatory noise. We moved beyond simple budgeting to Correlation Divergence, proving that the health of a business is defined not by the absolute value of its costs, but by the coupling coefficient ($\rho$) between its expenditure and its revenue velocity.

latex

# 5. References

## 5.1. Operations Research & Queuing Theory

- Erlang, A. K. (1909). The Theory of Probabilities and Telephone Conversations. *Nyt Tidsskrift for Matematik B*, *20*, 33–39.

  Core source for the Erlang-C formulas used in Section 2.1.

- Little, J. D. (1961). A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research*, *9*(3), 383–387.

  Foundational proof for the relationship between arrival rate and wait time.

- Green, L. V., Kolesar, P. J., & Soares, J. (2001). Improving the SIPP Approach for Staffing Service Systems That Have Cyclic Demands. *Operations Research*, *49*(4), 549–564.

Academic basis for handling the “Non-Homogeneous Poisson Process” (Time-varying demand).

- Dantzig, G. B. (1963). *Linear Programming and Extensions.* Princeton University Press.

  The seminal text on Linear Programming, supporting the MILP framework in Section 2.3.

## 5.2. Time-Series Analysis & Statistical Methods

- Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, *6*(1), 3–73.

  The primary source for the STL Decomposition method used in Section 3.3.

- Box, G. E. P., & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control.* Holden-Day.

  Standard reference for ARIMA and Z-Score residual analysis.

- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman & Hall.

  Reference for the Kernel Density Estimation (KDE) used in Section 2.1 to smooth the demand curve.

## 5.3. Hospitality Economics & Cost Behavior

- Anderson, M. C., Banker, R. D., & Janakiraman, S. N. (2003). Are Selling, General, and Administrative Costs “Sticky”? *Journal of Accounting Research*, *41*(1), 47–63.

  The foundational paper defining “Cost Stickiness” and Asymmetric Elasticity (The Ratchet Effect) discussed in Section 3.1.

- Kimes, S. E. (1989). Yield Management: A Tool for Capacity-Constrained Service Firms. *Journal of Operations Management*, *8*(4), 348–363.

  Supports the economic theory of optimizing perishable inventory (labor hours).