

Minimizing Total Cost for a Pizza Shop's Perishable Ingredient Inventory: Using Numerical Optimization

Student Name
Postgraduate Diploma in Artificial Intelligence
Optimization for Data Science
Indian Institute of Technology Jodhpur
Jodhpur, India
email address:

Nishant Kumar
Department of Electrical Engineering
Indian Institute of Technology Jodhpur
Jodhpur, India
Email address: drnkiit@gmail.com

Abstract—This

Keywords—component...

I. INTRODUCTION

In the competitive and margin-sensitive food service industry, effective inventory management is a critical determinant of profitability and sustainability [1]. This challenge is particularly acute for pizzerias, whose business model relies entirely on the constant availability of fresh, highly perishable ingredients. The core problem, known as the Perishable Ingredient Inventory Optimization (PIIO), involves determining the optimal order quantities for raw materials to minimize the total cost, a composite of ordering, holding, and spoilage costs, while consistently meeting customer demand [2]. This is a complex balancing act: ordering too little risks costly stockouts and lost sales, while ordering too much leads to significant financial and environmental waste from spoilage [3].

The economic and operational significance of this problem is substantial. For small-to-medium enterprises like independent pizzerias, inefficient ordering directly erodes the bottom line. The Food and Agriculture Organization (FAO) estimates that a significant portion of food waste occurs at the retail and food service levels, representing a pure financial loss [4]. Beyond the immediate economic impact, there is a crucial environmental imperative. Reducing food waste at the source is one of the most effective ways to lessen the carbon footprint associated with production, transportation, and landfill decomposition, aligning business efficiency with broader sustainability goals [5].

The theoretical foundation for this work is rooted in inventory management literature. The seminal Economic Order Quantity (EOQ) model, developed by Harris [6], provides a classic framework for balancing ordering and holding costs. However, its fundamental assumption of non-perishable goods is a major limitation for food inventory. This critical gap was addressed by Ghare and Schrader [7], who introduced an EOQ model with exponential decay, formally incorporating spoilage costs into the objective function. This adaptation opened the door for applying classical optimization techniques to find optimal order quantities for deteriorating

items. Subsequent research has expanded on this, with scholars like Bakker et al. [8] reviewing various deterioration models, and others applying gradient-based methods to solve these non-linear inventory problems efficiently [9]. These techniques are prized for their convergence properties when applied to the differentiable cost functions characteristic of these models. Despite this strong foundation, a gap persists in applying these models to the specific, multi-product context of a real-world pizzeria. Many classical models are single-product [10] or assume constant demand, failing to capture the reality of managing several ingredients with interdependent demand but independent spoilage rates. Furthermore, as noted in case studies like Dutta and Kumar's work [11], there is a need for models that are both mathematically rigorous and practically accessible for small business owners who lack specialized expertise in operations research.

This project aims to bridge these gaps by formulating a multi-product PIIO model for a typical pizzeria, focusing on three core ingredients: pizza dough, cheese, and fresh vegetables. The model is designed with a continuous and differentiable objective function, making it explicitly suitable for solution using the classical optimization techniques that are the focus of this course. The formulation realistically accounts for constraints such as storage capacity and supplier minimum orders. By providing a structured mathematical framework and realistic test cases, this work establishes a practical and actionable approach to a pervasive operational challenge, demonstrating the power of classical optimization to drive both economic and environmental benefits.

II. PROBLEM FORMULATION

A. Objective Function Formulation

The core objective for "Tony's Pizzeria" is to minimize the total daily cost associated with managing the inventory of its three most critical and perishable ingredients: Pizza Dough, Cheese, and Fresh Vegetables. The total cost is a function of the order quantities for these items. The primary decision variables are defined as: Q_d : Order quantity for pizza dough (kg), Q_c : Order quantity for cheese (kg), Q_v : Order quantity for fresh vegetables (kg).

The total cost function, C_{total} , is composed of three distinct cost components for each ingredient: Ordering Cost, Holding Cost, and Spoilage Cost.

1) Ordering Cost

This is the fixed cost incurred every time an order is placed with a supplier, regardless of the order size. It includes administrative expenses, delivery fees, and labor for processing. If demand is constant, the number of orders placed per day is the daily demand divided by the order quantity. The total daily ordering cost is the sum of these costs for all ingredients.

$$C_{order} = \frac{D_d}{Q_d} \cdot S_d + \frac{D_c}{Q_c} \cdot S_c + \frac{D_v}{Q_v} \cdot S_v \quad (1)$$

where: D_d, D_c, D_v : Average daily demand (kg/day) for dough, cheese, and vegetables, respectively. S_d, S_c, S_v : Fixed ordering cost (\$/order) for dough, cheese, and vegetables, respectively.

2) Holding Cost

This cost represents the expense of storing inventory, including refrigeration, electricity, and the opportunity cost of capital tied up in stock. Using the classic Economic Order Quantity (EOQ) assumption, the average inventory level is half the order quantity. The total daily holding cost is:

$$C_{hold} = h_d \cdot \frac{Q_d}{2} + h_c \cdot \frac{Q_c}{2} + h_v \cdot \frac{Q_v}{2} \quad (2)$$

where: h_d, h_c, h_v : Daily holding cost per kg (\$/kg/day) for dough, cheese, and vegetables, respectively.

3) Spoilage Cost

This is the most critical cost for perishables. It is the financial loss from ingredients that spoil before they can be used. We model the *average amount spoiled per day* as a fraction of the average inventory level. This fraction, the spoilage rate (θ), is higher for more perishable items like vegetables.

$$C_{waste} = p_d \cdot \theta_d \cdot \frac{Q_d}{2} + p_c \cdot \theta_c \cdot \frac{Q_c}{2} + p_v \cdot \theta_v \cdot \frac{Q_v}{2} \quad (3)$$

where: p_d, p_c, p_v : Purchase price per kg (\$/kg) for dough, cheese, and vegetables, respectively. $\theta_d, \theta_c, \theta_v$: Daily spoilage rate (a dimensionless fraction) for dough, cheese, and vegetables, respectively.

4) Complete Objective Function

The overall goal is to minimize the sum of all these costs. Therefore, the complete objective function is formulated as follows:

$$\text{Minimize } C_{total}(Q_d, Q_c, Q_v) = C_{order} + C_{hold} + C_{waste}$$

Substituting equations (1), (2), and (3):

$$\text{Minimize } C_{total} = \left(\frac{D_d}{Q_d} S_d + \frac{D_c}{Q_c} S_c + \frac{D_v}{Q_v} S_v \right) +$$

$$\frac{1}{2} (Q_d(h_d + p_d \theta_d) + Q_c(h_c + p_c \theta_c) + Q_v(h_v + p_v \theta_v)) \quad (4)$$

The objective function in (4) is a non-linear, convex function for each $Q_i > 0$. The first term (ordering cost) decreases as order quantities increase, while the second term (combined holding and spoilage cost) increases linearly with order quantities. The optimal solution is the set of Q_d, Q_c, Q_v that finds the minimum point of this aggregate cost curve for each ingredient, effectively balancing the cost of ordering too frequently against the cost of holding and wasting too much inventory.

B. Constraints

The optimization is not unconstrained; the decision variables Q_d, Q_c, Q_v are bounded by practical, real-world limitations of the pizzeria's operations. These constraints ensure the solution is feasible and implementable.

1) Storage Capacity Constraints

The physical space in the kitchen, particularly refrigeration for cheese and vegetables, is limited. The order quantity for any ingredient cannot exceed its designated storage capacity.

$$Q_d \leq C_{cap,d} \quad (5)$$

$$Q_c \leq C_{cap,c} \quad (6)$$

$$Q_v \leq C_{cap,v} \quad (7)$$

where: $C_{cap,d}, C_{cap,c}, C_{cap,v}$: Maximum storage capacity (kg) for dough, cheese, and vegetables, respectively.

Significance: These are *hard constraints* representing the physical space in dry storage and refrigerators. Violating them is impossible, as there is literally no space to store the excess goods.

2) Supplier Minimum Order Constraints

Suppliers often impose a minimum order quantity to make a delivery economically viable. The pizzeria must order at least this amount.

$$Q_d \geq Q_{min,d} \quad (8)$$

$$Q_c \geq Q_{min,c} \quad (9)$$

$$Q_v \geq Q_{min,v} \quad (10)$$

where: $Q_{min,d}, Q_{min,c}, Q_{min,v}$: Minimum order quantity (kg) imposed by the suppliers for dough, cheese, and vegetables, respectively.

Significance: This is a common business-to-business requirement. It prevents the model from suggesting trivial, impractically small orders that a supplier would not fulfill.

3) Demand Fulfillment / Service Level Constraints

To avoid stockouts and ensure that customer demand can be met, the order quantity must be sufficient to cover demand for a reasonable period. A simple and effective constraint is to ensure the order quantity is at least equal to the demand

over the review period (e.g., one day), preventing orders of zero.

$$Q_d \geq D_d \cdot T_{review} \quad (11)$$

$$Q_c \geq D_c \cdot T_{review} \quad (12)$$

$$Q_v \geq D_v \cdot T_{review} \quad (13)$$

where: T_{review} : The review period (in days). For a daily review model, $T_{review} = 1$, meaning $Q_i \geq D_i$.

Significance: This constraint enforces a basic service level. It guarantees that when an order is placed, it is large enough to cover at least the expected demand until the next potential order, thus preventing immediate stockouts.

4) Non-negativity Constraints

The order quantities must be positive values.

$$Q_d > 0, Q_c > 0, Q_v > 0 \quad (14)$$

Significance: This is a fundamental requirement for a physically meaningful solution. A negative order quantity has no practical interpretation.

The constrained optimization problem is thus to minimize the objective function C_{total} in Eq. (4) subject to the set of linear inequality and positivity constraints defined in Eqs. (5) through (14). This formulation creates a well-defined mathematical model that accurately reflects the operational realities of the pizzeria.

III. TEST CONDITIONS / DATA SETS

The following test cases define realistic operational scenarios for different sizes of pizzerias. The data for daily demand, costs, spoilage rates, and constraints are provided to facilitate numerical optimization.

1) Case 1: "Tony's Corner Slice" - Small Shop

This case models a family-owned, small storefront operation, such as a local neighborhood pizzeria. It primarily serves walk-in customers and has a limited delivery radius. The kitchen and storage areas are compact, and daily sales volumes are modest but consistent. The focus is on freshness and managing a tight budget, where even small amounts of waste are noticeable. This case tests the optimization model under significant space and budget constraints. It is expected that the optimal order quantities will be relatively low, and the solution will likely be driven by the need to avoid spoilage, potentially making the minimum order quantity or the daily demand constraint a binding factor for highly perishable items like vegetables.

Table 1: Parameters and Constraints for Case 1 (Small Shop)

Parameter	Description	Pizza Dough	Cheese	Vegetables
D_i	Daily Demand (kg/day)	8	3	2
S_i	Ordering Cost (\$/order)	10	12	8
h_i	Holding Cost (\$/kg/day)	0.08	0.15	0.12
p_i	Price (\$/kg)	2.00	6.50	4.50
θ_i	Spoilage Rate (per day)	0.04	0.08	0.20
$C_{cap,i}$	Storage Capacity (kg)	25	15	8
$Q_{min,i}$	Min. Order Qty. (kg)	5	2	1

2) Case 2: "Pizza Centro" - Medium Shop

This case represents a well-established, medium-sized pizza restaurant. It features a dedicated dine-in area and supports a robust delivery and takeaway service, often utilizing multiple online food delivery platforms. The operation has a larger, more organized kitchen with dedicated refrigeration and dry storage, allowing it to handle a higher and more variable volume of orders typical of a successful urban location. This is the standard, balanced test case. The storage constraints are less restrictive, and demand is high enough to justify larger orders. The model is expected to find optimal quantities where the trade-off between ordering, holding, and spoilage costs is most efficient, likely without being forced to the upper or lower constraint limits, providing a classic illustration of inventory optimization.

Table 2: Parameters and Constraints for Case 2 (Medium Shop)

Parameter	Description	Pizza Dough	Cheese	Vegetables
D_i	Daily Demand (kg/day)	25	10	7
S_i	Ordering Cost (\$/order)	15	15	10
h_i	Holding Cost (\$/kg/day)	0.10	0.20	0.15
p_i	Price (\$/kg)	1.90	6.00	4.00
θ_i	Spoilage Rate (per day)	0.05	0.10	0.25
$C_{cap,i}$	Storage Capacity (kg)	80	40	25
$Q_{min,i}$	Min. Order Qty. (kg)	10	5	3

3) Case 3: "The Mega Pie Factory" - Large Shop / Central Kitchen

This case simulates a large-scale operation, which could be a central kitchen supplying several franchise outlets or a very high-volume standalone restaurant in a prime location. The scale of operation is significant, with bulk purchasing power and extensive, industrial-grade storage facilities. The primary challenge at this scale is managing the immense financial impact of spoilage across large inventories while leveraging economies of scale for ordering. This case tests the model under conditions where the spoilage cost term dominates the objective function due to the high volumes involved. It is highly probable that the optimal unconstrained order quantity for the most perishable item (vegetables) will exceed the physical storage capacity, making the storage constraint a binding factor. This will demonstrate the model's ability to handle active constraints and find the best feasible solution under real-world limitations.

Table 3: Parameters and Constraints for Case 3 (Large Shop)

Parameter	Description	Pizza Dough	Cheese	Vegetables
D_i	Daily Demand (kg/day)	80	35	20
S_i	Ordering Cost (\$/order)	20	25	15
h_i	Holding Cost (\$/kg/day)	0.05	0.10	0.08
p_i	Price (\$/kg)	1.80	5.80	3.80
θ_i	Spoilage Rate (per day)	0.03	0.07	0.15
$C_{cap,i}$	Storage Capacity (kg)	300	150	80
$Q_{min,i}$	Min. Order Qty. (kg)	20	10	5

IV. OPTIMIZATION TECHNIQUES

This section details the core mathematical framework and numerical algorithms used to solve the constrained optimization problems in this paper. The Lagrangian formulation is used for handling constraints and the iterative numerical methods employed to find the solution.

1) Problem Formulation using the Lagrangian

The general nonlinear programming problem is defined as:

$$\begin{aligned} & \min_{\mathbf{x}} f(\mathbf{x}) \\ & \text{subject to } \left. \begin{aligned} c_i(\mathbf{x}) &= 0, i \in \mathcal{E}(\text{Equality Constraints}) \\ c_i(\mathbf{x}) &\geq 0, i \in \mathcal{I}(\text{Inequality Constraints}) \end{aligned} \right\} \quad (1) \end{aligned}$$

To handle these constraints, we formulate the Lagrangian function, which incorporates the objective function and the constraints into a single scalar function:

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \sum_{i \in \mathcal{E}} \lambda_i c_i(\mathbf{x}) - \sum_{j \in \mathcal{I}} \mu_j c_j(\mathbf{x}) \quad (2)$$

Where, $\mathbf{x} \in \mathbb{R}^n$: Vector of primal variables (decision variables). $f(\mathbf{x})$: Objective function to be minimized. $c_i(\mathbf{x})$: Constraint functions. \mathcal{E} : Set of indices for equality constraints. \mathcal{I} : Set of indices for inequality constraints. λ : Vector of Lagrange multipliers associated with equality constraints. μ : Vector of Lagrange multipliers associated with inequality constraints (with $\mu_j \geq 0$).

The resulting problem is then:

$$\min_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}, \lambda; \mu) \quad (3)$$

2) Line Search Method

All descent methods discussed herein operate within a line search framework. Given a current iterate \mathbf{x}_k and a descent direction \mathbf{p}_k , the update is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \quad (4)$$

The step-length $\alpha_k > 0$ is selected to enforce the strong Wolfe conditions, which guarantee sufficient decrease and prevent unacceptably small steps:

$$\left. \begin{aligned} \mathcal{L}_A(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &\leq \mathcal{L}_A(\mathbf{x}_k) + c_1 \alpha_k \nabla \mathcal{L}_A(\mathbf{x}_k)^T \mathbf{p}_k (\text{Sufficient Decrease}) \\ |\nabla \mathcal{L}_A(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| &\leq c_2 |\nabla \mathcal{L}_A(\mathbf{x}_k)^T \mathbf{p}_k| (\text{Curvature Condition}) \end{aligned} \right\} \quad (5)$$

where $0 < c_1 < c_2 < 1$. A practical algorithm for finding such a step length is the backtracking line search with Armijo condition. $\mathbf{p}_k \in \mathbb{R}^n$: Search direction vector. $\alpha_k \in \mathbb{R}_+$: Step length. c_1, c_2 : Wolfe condition constants.

3) Newton's Method

Newton's method leverages second-order derivative information to achieve rapid local convergence. The canonical Newton direction \mathbf{p}_k^N is obtained by solving the linear system:

$$\nabla^2 \mathcal{L}_A(\mathbf{x}_k) \mathbf{p}_k^N = -\nabla \mathcal{L}_A(\mathbf{x}_k) \quad (6)$$

To ensure global convergence and handle non-convexity, a modified Cholesky factorization is often employed. This involves factoring a perturbed Hessian:

$$\nabla^2 \mathcal{L}_A(\mathbf{x}_k) + \mathbf{E} = \mathbf{LDL}^T \quad (7)$$

where \mathbf{E} is a non-negative diagonal matrix chosen such that \mathbf{LDL}^T is sufficiently positive definite. The update is then:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k^N \quad (8)$$

The convergence is quadratic near a strict local minimizer:

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \kappa \|\mathbf{x}_k - \mathbf{x}^*\|^2 \quad (9)$$

It is for some constant $\kappa > 0$.

Where, $\nabla^2 \mathcal{L}_A(\mathbf{x}_k) \in \mathbb{R}^{n \times n}$: Hessian matrix of the Augmented Lagrangian. $\mathbf{p}_k^N \in \mathbb{R}^n$: Newton search direction. \mathbf{E} : Diagonal perturbation matrix. \mathbf{L}, \mathbf{D} : Lower triangular and diagonal factors from the modified Cholesky decomposition.

4) Quasi-Newton Method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method constructs a Hessian approximation $\mathbf{B}_k \approx \nabla^2 \mathcal{L}_A(\mathbf{x}_k)$ iteratively. The search direction is computed by solving:

$$\mathbf{B}_k \mathbf{p}_k^{QN} = -\nabla \mathcal{L}_A(\mathbf{x}_k) \quad (9)$$

The BFGS update formula, which preserves positive definiteness, is given by:

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \quad (10)$$

Where the vectors \mathbf{s}_k and \mathbf{y}_k are defined as:

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k, \mathbf{y}_k = \nabla \mathcal{L}_A(\mathbf{x}_{k+1}) - \nabla \mathcal{L}_A(\mathbf{x}_k) \quad (11)$$

An equivalent and often computationally advantageous update for the inverse Hessian $\mathbf{H}_k = \mathbf{B}_k^{-1}$ is:

$$\mathbf{H}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) \mathbf{H}_k \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \quad (12)$$

This allows the search direction to be computed via a matrix-vector product: $\mathbf{p}_k^{QN} = -\mathbf{H}_k \nabla \mathcal{L}_A(\mathbf{x}_k)$.

Where, $\mathbf{B}_k \in \mathbb{R}^{n \times n}$: Hessian approximation. $\mathbf{H}_k \in \mathbb{R}^{n \times n}$: Inverse Hessian approximation. $\mathbf{s}_k \in \mathbb{R}^n$: Variable displacement vector. $\mathbf{y}_k \in \mathbb{R}^n$: Gradient difference vector.

5) Nonlinear Conjugate Gradient Method

For large-scale problems, the Nonlinear Conjugate Gradient (NCG) method provides a low-memory alternative. The search direction is updated recursively:

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}, \mathbf{p}_0 = -\mathbf{g}_0 \quad (13)$$

where $\mathbf{g}_k = \nabla \mathcal{L}_A(\mathbf{x}_k)$. Different choices for the scalar β_k yield different CG variants. Prominent formulas include:

- Fletcher-Reeves (FR):

$$\beta_k^{FR} = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \quad (14)$$

- Polak-Ribière (PR):

$$\beta_k^{PR} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \quad (15)$$

- Hestenes-Stiefel (HS):

$$\beta_k^{HS} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{p}_{k-1}} \quad (16)$$

To ensure convergence, a restart condition is often applied, typically setting $\beta_k = 0$ (i.e., $\mathbf{p}_k = -\mathbf{g}_k$) if $|\mathbf{g}_k^T \mathbf{g}_{k-1}| > 0.2 \|\mathbf{g}_k\|^2$.

Where, $\mathbf{g}_k \in \mathbb{R}^n$: Gradient vector at \mathbf{x}_k . $\beta_k \in \mathbb{R}$: Conjugate gradient update parameter.

6) Dichotomous Search

The Dichotomous Search method works by repeatedly evaluating the function at two points near the center of the current interval, halving the interval of uncertainty with each pair of evaluations.

Given an initial interval $[a_0, b_0]$ and a small constant $\delta > 0$ (where $\delta \ll (b_0 - a_0)$), the algorithm for iteration k is:

1. Compute two interior points:

$$\alpha_1^k = \frac{a_k + b_k}{2} - \frac{\delta}{2}, \alpha_2^k = \frac{a_k + b_k}{2} + \frac{\delta}{2} \quad (17)$$
2. Evaluate the function at these points: $\phi(\alpha_1^k)$ and $\phi(\alpha_2^k)$.
3. Update the interval based on the function values:
 If $\phi(\alpha_1^k) < \phi(\alpha_2^k)$: $a_{k+1} = a_k, b_{k+1} = \alpha_2^k$ (18)
 If $\phi(\alpha_1^k) \geq \phi(\alpha_2^k)$: $a_{k+1} = \alpha_1^k, b_{k+1} = b_k$ (19)

The length of the interval after N iterations is given by:

$$L_N = \frac{b_0 - a_0}{2^N} + \delta(1 - \frac{1}{2^N}) \quad (20)$$

For small δ , the reduction per iteration is approximately linear, with a ratio of $1/2$.

Where, $[a_k, b_k]$: Interval of uncertainty at iteration k . δ : A small positive constant ensuring distinct evaluation points. α_1^k, α_2^k : Interior evaluation points. L_N : Length of the uncertainty interval after N iterations.

7) Fibonacci Search

The Fibonacci Search is an optimal method for minimizing the number of function evaluations required to reduce the initial interval to a specified length. It uses a sequence of Fibonacci numbers, defined by $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$, to determine the evaluation points.

Given a desired final interval length ϵ (or a total number of iterations N), the initial points are chosen as:

$$\alpha_1^1 = a_0 + (\frac{F_{N-2}}{F_N})(b_0 - a_0), \alpha_2^1 = a_0 + (\frac{F_{N-1}}{F_N})(b_0 - a_0) \quad (21)$$

For iterations $k = 1, 2, \dots, N - 1$:

1. Evaluate $\phi(\alpha_1^k)$ and $\phi(\alpha_2^k)$.
2. If $\phi(\alpha_1^k) < \phi(\alpha_2^k)$, set:
 $a_{k+1} = a_k, b_{k+1} = \alpha_2^k, \alpha_2^{k+1} = \alpha_1^k$ (22)
 $\alpha_1^{k+1} = a_{k+1} + (\frac{F_{N-k-2}}{F_{N-k}})(b_{k+1} - a_{k+1})$ (23)
3. If $\phi(\alpha_1^k) \geq \phi(\alpha_2^k)$, set:
 $a_{k+1} = \alpha_1^k, b_{k+1} = b_k, \alpha_1^{k+1} = \alpha_2^k$ (24)
 $\alpha_2^{k+1} = a_{k+1} + (\frac{F_{N-k-1}}{F_{N-k}})(b_{k+1} - a_{k+1})$ (25)

The final interval length is:

$$L_N = \frac{b_0 - a_0}{F_N} \quad (26)$$

Where, F_n : The n -th Fibonacci number. N : Total number of planned function evaluations. ϵ : Desired final interval accuracy.

8) Golden Section Search

The Golden Section Search is a suboptimal but elegant and computationally efficient variant of the Fibonacci search. It

uses the golden ratio $\tau = \frac{\sqrt{5}-1}{2} \approx 0.618034$ to determine the evaluation points, maintaining a constant reduction ratio per iteration.

The evaluation points are computed relative to the current interval $[a_k, b_k]$ as:

$$\alpha_1^k = a_k + (1 - \tau)(b_k - a_k), \alpha_2^k = a_k + \tau(b_k - a_k) \quad (27)$$

The update rule is:

$$\text{If } \phi(\alpha_1^k) < \phi(\alpha_2^k): a_{k+1} = a_k, b_{k+1} = \alpha_2^k, \alpha_2^{k+1} = \alpha_1^k \quad (28)$$

$$\text{If } \phi(\alpha_1^k) \geq \phi(\alpha_2^k): a_{k+1} = \alpha_1^k, b_{k+1} = b_k, \alpha_1^{k+1} = \alpha_2^k \quad (29)$$

A new point (α_1^{k+1} or α_2^{k+1}) is computed using the golden ratio formula to maintain symmetry.

The interval is reduced by a factor of τ every iteration. After N iterations:

$$L_N = \tau^{N-1}(b_0 - a_0) \quad (30)$$

Where, τ : The golden ratio constant, $\tau = \frac{\sqrt{5}-1}{2}$. α_1^k, α_2^k : Interior points defined by the golden ratio.

9) Bisection Method

The Bisection Method is a root-finding algorithm applied to the derivative of the function, $\phi'(\alpha)$. It is used when the derivative is available and is more efficient than zero-order methods.

Given an initial bracket $[a_0, b_0]$ where $\phi'(a_0) < 0$ and $\phi'(b_0) > 0$, indicating a minimum lies within, the algorithm for iteration k is:

1. Find the midpoint: $\alpha_m^k = \frac{a_k + b_k}{2}$
2. Evaluate the derivative at the midpoint: $\phi'(\alpha_m^k)$
3. Update the interval:
 If $\phi'(\alpha_m^k) > 0$: $a_{k+1} = a_k, b_{k+1} = \alpha_m^k$
 If $\phi'(\alpha_m^k) < 0$: $a_{k+1} = \alpha_m^k, b_{k+1} = b_k$
 If $\phi'(\alpha_m^k) = 0$: Terminate.

The error after N iterations is bounded by:

$$|\alpha^* - \alpha_m^N| \leq \frac{b_0 - a_0}{2^{N+1}} \quad (31)$$

This demonstrates a linear convergence rate with a ratio of $1/2$.

Where, $\phi'(\alpha)$: The derivative of the one-dimensional function. α_m^k : The midpoint of the interval at iteration k .

10) Steepest Descent Method

The Steepest Descent (or Gradient Descent) method is a foundational first-order iterative algorithm for multi-variable minimization. It uses the negative gradient as the search direction, which is the direction of the locally steepest decrease of the function.

The core update formula is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \text{ where } \mathbf{p}_k = -\nabla f(\mathbf{x}_k) \quad (32)$$

The step length α_k is chosen to minimize the function along the search direction, often using the line search methods described in Section 1:

$$\alpha_k = \arg \min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{p}_k) \quad (33)$$

A common, albeit less efficient, alternative is to use a fixed or diminishing step size rule, such as $\alpha_k = \frac{1}{k}$.

The convergence of the Steepest Descent method can be characterized for quadratic functions of the form $f(x) = \frac{1}{2}x^T Qx - b^T x$. The optimal step length for a quadratic function is given by:

$$\alpha_k = \frac{p_k^T p_k}{p_k^T Q p_k} = \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T Q \nabla f_k} \quad (34)$$

The error convergence rate for a strongly convex quadratic function is linear and depends on the condition number $\kappa(Q)$ of the Hessian matrix Q :

$$\|x_{k+1} - x^*\|_Q \leq \left(\frac{\kappa(Q)-1}{\kappa(Q)+1}\right)^2 \|x_k - x^*\|_Q \quad (35)$$

where $\kappa(Q) = \frac{\lambda_{\max}(Q)}{\lambda_{\min}(Q)}$ and $\|y\|_Q^2 = y^T Q y$. This shows that the convergence slows down dramatically as $\kappa(Q)$ increases. Where, $p_k \in \mathbb{R}^n$: Search direction (negative gradient). $\nabla f(x_k)$: Gradient of the objective function at x_k . $Q \in \mathbb{R}^{n \times n}$: Symmetric positive definite Hessian matrix (for quadratic analysis). $\kappa(Q)$: Condition number of matrix Q . $\lambda_{\max}, \lambda_{\min}$: Maximum and minimum eigenvalues of Q .

Note: You are free to choose any suitable method to solve your problem; you are not limited to the techniques listed here. Select any four methods, but make sure each chosen technique can handle problems with three or more variables, as some methods work only for single-variable cases and will not be applicable here.

11) Comparative Analysis

The optimisation methods that are offered form a complementary hierarchy, and choosing a method entails weighing computational cost and convergence rate. These algorithms are frequently combined in hybrid approaches in practice. Global convergence is usually guaranteed by a Line Search framework, and the properties of the particular method are determined by the search direction computation: Quasi-Newton Methods provide superlinear convergence as a workable compromise; Newton's Method offers quadratic convergence with high computational cost; Steepest Descent offers low-cost directions but slow convergence; and Conjugate Gradient permits large-scale optimisation with low memory requirements. The line search subproblem is effectively solved by one-dimensional searches such as Golden Section and Bisection, with the latter being favoured when derivatives are available..

Table I. Comparative Analysis of Optimization Techniques

Technique	Convergence Rate	Computational Cost per Iteration	Memory Footprint	Primary Application Context
Line Search	Linear	$\mathcal{O}(n)$ per evaluation	$\mathcal{O}(n)$	Foundational component for global convergence in descent methods.
Steepest Descent	Linear (dependent on condition number κ ; often slow)	Low ($\mathcal{O}(n)$)	$\mathcal{O}(n)$	Initialization; very large-scale problems where iteration cost is paramount.
Conjugate Gradient	Linear (with superlinear clusters for quadratic functions)	Low ($\mathcal{O}(n)$ per matrix-vector product)	$\mathcal{O}(n)$	Very large-scale problems ($n > 10^4$); inner solver for linear systems.
Quasi-Newton (BFGS)	Superlinear	Medium ($\mathcal{O}(n^2)$)	$\mathcal{O}(n^2)$	Default choice for medium-scale problems; Hessian is unavailable or expensive.
Newton's Method	Quadratic	High ($\mathcal{O}(n^3)$ factorization, $\mathcal{O}(n^2)$ evaluation)	$\mathcal{O}(n^2)$	Medium-scale problems with an analytic/tractable Hessian; requires high accuracy.
Golden Section Search	Linear (reduction ratio $\tau \approx 0.618$)	1 function evaluation	$\mathcal{O}(1)$	Robust, derivative-free minimization within a line search.
Bisection Method	Linear (reduction ratio 1/2)	1 derivative evaluation	$\mathcal{O}(1)$	Most efficient 1D search when derivatives are available; solves $\phi'(\alpha) \approx 0$.

V. RESULTS AND ANALYSIS

This section details the core mathematical

1) Case 1:

This case models.....

Table II. Results of Case-I

	F(x)	Variable-1	Variable-2	Variable-3	Variable-4
Method-1					
Method-2					
Method-3					
Method-4					

Table III. Algorithm performance of Case-I

	No. of iteration required	No. of deferential terms used	% Accuracy	Overall solution complexity
Method-1				
Method-2				
Method-3				
Method-4				

2) Case 2:

This case models

Table IV. Results of Case-II

	F(x)	Variable-1	Variable-2	Variable-3	Variable-4
Method-1					
Method-2					
Method-3					
Method-4					

Table V. Algorithm performance of Case-II

	No. of iteration required	No. of differential terms used	% Accuracy	Overall solution complexity
Method-1				
Method-2				
Method-3				
Method-4				

3) Case 3: ...

This case

Table VI. Results of Case-III

	F(x)	Variable-1	Variable-2	Variable-3	Variable-4
Method-1					
Method-2					
Method-3					
Method-4					

Table VII. Algorithm performance of Case-II

	No. of iteration required	No. of differential terms used	% Accuracy	Overall solution complexity
Method-1				
Method-2				
Method-3				
Method-4				

VI. CONCLUSIONS

This

REFERENCES

- [1] K. Dutt and N. Kumar, "Digital Twin-Based Framework for Dynamic Stability Analysis of Grid-Tied Photovoltaic System," *IEEE Transactions on Industrial Cyber-Physical Systems*, 2025.
- [2] A. Kumar and N. Kumar, "Digital Twin Framework for 1-Phase Grid-Tied PV System: A Frequency Domain Modeling and E2FD-HO-Based Approach for Power Electronic Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2025.
- [3] R. Pandey and N. Kumar, "Digital Twin Formation and Real-Time Parameter Estimation for Consumer-Integrated Grid-Tied PV Systems Using IFO-GDF and Coulomb-EFO," *IEEE Transactions on Consumer Electronics*, 2025.

- [4] N. K. Kumawat and N. Kumar, "Comprehensive Component Health Monitoring of 3-Phase 2-Stage Grid Connected PV System via Digital Twin," *IEEE Transactions on Industrial Electronics*, 2025.
- [5] A. Kumar and N. Kumar, "State-Space Driven Digital Twin for Condition Monitoring and Predictive Health Assessment in Grid-Integrated Power Converter System," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 3, pp. 464-471, 2025.
- [6] R. Pandey and N. Kumar, "Enhanced Power Quality Control in Microgrid-Assisted Electric Vehicle Charging Systems Using ATOGI and CIDPC," *IEEE Transactions on Industry Applications*, vol. 61, no. 1, pp. 676-685, Jan.-Feb. 2025.
- [7] Raghav and N. Kumar, "Consumer-Focused Solutions for Energy Mobility and Security in Off-Grid Farmhouses," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 1, pp. 1784-1791, Feb. 2025.
- [8] N. Kumar, "EV Charging Adapter to Operate With Isolated Pillar Top Solar Panels in Remote Locations," *IEEE Transactions on Energy Conversion*, vol. 39, no. 1, pp. 29-36, March 2024.
- [9] N. Kumar and S. K. Panda, "A Multipurpose and Power Quality Improved Electric Vessels Charging Station for the Seaports," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3254-3261, March 2023.
- [10] N. Kumar and S. K. Panda, "Smart High Power Charging Networks and Optimal Control Mechanism for Electric Ships," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1476-1483, Feb. 2023.
- [11] Sukanya Satapathy, S. and Kumar, N., "Framework of maximum power point tracking for solar PV panel using WSPS technique", *IET Renewable Power Generation*, vol. 14: 1668-1676, 2020.

Note: Page limit is 6 pages.

Similarity excluding references will be counted.

Must add these 11 references in your paper.