Optimization of Production Planning with Python's SciPy: A Computational Study

Sana Shabbir[1], Aamir Shahzad[2]*, Tasadduq Niaz[1], Sidra Ashraf[3], Namal[1], Faheem Khan[4]
[1]Faculty of Sciences, Superior University Lahore, Lahore 54000, Pakistan
[2]Department of Mathematics, Baba Guru Nanak University, Nankana Sahib 39100, Pakistan
[3]Department of Mathematics, University of Sargodha, Sargodha 40100, Pakistan
[4]Department of Mathematics, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
*Correspondence: aamir.shahzad@bgnu.edu.pk

Citation | Shabbir. S, Shahzad. A, Niaz. T, Ashraf. S, Namal, Khan. F, "Optimization of Production Planning with Python's SciPy: A Computational Study", IJIST, Vol. 07 Issue. 04 pp 2279-2289, October 2025
Received | August 23, 2025 **Revised |** September 30, 2025 **Accepted |** October 02, 2025 **Published |** October 04, 2025.

Production planning optimization is the act of effectively distributing limited resources, including labor, materials, and equipment, to achieve production targets while optimizing profit and reducing waste. This study analyzes how optimization methods can be applied to production planning models in the cooking oil sector, with a particular emphasis on how linear programming (LP) can be used to handle usable quality limitations to maximize gross profit. The goal of this study is to find the best values for decision variables across a variety of inventory-based production frameworks. It is important in a manufacturing zone where input bound must be weighed against consumer needs, such as the industry of cooking oil. In order to provide a computational method for determining the perfect production levels, the study establishes a linear programming (LP) model and solves it using Python's SciPy package. This optimization method uses objective functions involving dense matrices and numerical equations to solve the production planning problem. In calculating output levels and profit margins, the numerical results show a significant convergence, rating the effectiveness and credibility of the suggested approach in providing the optimal solution for practical industrial planning.

Keywords: Optimization; Linear Programming (LP); Python; Production Planning Problem; SciPy.

Introduction:

The purpose of optimization techniques is to find the optimal solution that satisfies all requirements. Actually, the goal of this research is to employ software and numerical optimization for linear programming to discover the optimal values of various inventory systems' decision-making variables to optimize net profit. Additionally, models of the cooking oil production problem (production planning problem) were fitted with the optimum process using a variety of methods, including objectives and equations, dense matrices. In a chain of closed-loop supplies, the goal of production planning, and more especially lot sizing, is to make the most effective utilization of resources Whitin the planning horizon to meet needs through manufacturing and remanufacturing. Since the early 20th century, scholars have focused on production planning. The first researchers on it were Wagner and Whitin. They used forward dynamic programming to address a production planning problem for a single product, one stage, and several periods without capacity limitations. Among the many issues covered in production planning research, lot sizing is one of the most complicated [1].

Finding the best answer that meets all needs is made easier with the use of optimization techniques [2]. Choosing the best option from a range that satisfies the requirements and does not go against any constraints is known as optimization [2]. Finding the best solution that meets every restriction is the aim of optimization strategies. This method was used with several formats, applying formulas, goals, dense matrices, and sparse matrices to many models of the production planning issue, most notably the difficulty of producing cooking oil [2][3].

Usually, optimization problems are used to formulate and solve production planning difficulties [4]. Businesses use production planning as a strategic method when deciding how to plan their product manufacturing. To ensure efficient and effective production processes, it entails identifying the product to be produced, production amount, capacity planning, material requirements, scheduling timelines, and other relevant details [2][3][4][5].

Maximizing income at fixed, known resource and demand levels is the aim of the production planning problem. Finding the best production plan to maximize income while keeping in mind demand and resource limits is the goal [2].

The following significant checkpoints are part of the production planning process: Analysis of customer demand, Planning production capacity, Control finance, quality, and manufacturing, Evaluate and enhance the production process, and Finish manufacturing the product. From demand research through final production and delivery, these benchmarks stand for the important phases of the production planning process [2].

A factory that produces cooking oil will take the following factors into account when planning its production: Choose the most effective order of operations for obtaining raw ingredients, processing, producing, packing, and distributing cooking oils. To help with production planning, represent this sequence as an objective function in a mathematical model. To account for uncertainty, include interval data in both the objective function and the limitations. This will produce a more robust optimization approach for the production planning problem [2]. Establish and optimize processes, including product concentration levels, for transforming raw materials into finished commodities, raw material mixing ratios, and packaging needs. Represent these processes and actions as constraints in the mathematical model to ensure that the optimization function generates feasible and practical solutions. The purpose of this work is to optimize inventory systems by identifying the optimal values for decision variables that maximize net profit using numerical linear programming techniques. The recommended approach reduces waste, increases profitability, and optimizes production schedules while taking into account real-world constraints for the cooking oil sector. Since the early 1950s, when production planning problems were first defined and treated as optimization problems, a vast body of literature has grown. Extensive use of Enterprise Resource Planning systems, together with advances in scientific computing and information technology, has made

these methodologies more accessible to a broader industry. This chapter examines the fundamental formulations that have dominated academic research and industry practice for the past fifty years, weighing their advantages and disadvantages, and discussing some interesting new avenues that have recently emerged. We emphasize the difficulty in establishing expected lead times and focus on models that aid in decisions on production quantities and order release over time [4].

Objectives of the Study:

The main objective of this study is to develop and demonstrate a computational framework for optimizing production planning in the cooking oil sector using Python's SciPy library. Specifically, the research aims to design and implement a linear programming (LP) model that effectively allocates limited resources such as labor, materials, and equipment to maximize gross profit while meeting demand and quality constraints. By focusing on inventory-based production frameworks, the study seeks to identify optimal decision variable values that balance input limitations against consumer requirements, thereby providing actionable insights for industrial applications.

Novelty of the Statement:

The novelty of this work lies in its integration of Python's SciPy optimization tools into the context of production planning for the cooking oil industry, an application domain that has not been extensively explored in existing literature. Unlike traditional approaches that rely on generic optimization software or theoretical models, this study delivers a practical, open-source, and computationally efficient solution that leverages dense matrix operations and numerical algorithms for real-world industrial planning. The significant convergence of numerical results underscores the reliability of the proposed method, highlighting its potential to serve as a credible and cost-effective alternative for industries seeking to optimize production processes while enhancing profitability. By using Python's SciPy optimization tools, one can obtain more efficient results in the cooking oil industry.

Literature Review:

Historically, the first software to adapt to changes in hardware has been dense linear algebra (DLA). This is because DLA is an effective technique for identifying and putting into practice solutions to problems presented by novel designs, and it is essential to the accuracy and effectiveness of a wide range of applications [6]. A common operation in high-performance and scientific applications is sparse matrix vector multiplication (SpMV), which is often the cause of application performance constraints. An accessible and current introduction to the CasADi framework is provided in this article. Over the past seven years, the CasADi framework has undergone several design advancements [5]. The nested issue formulation allows us to identify solutions that perform well across a wide range of possible outcomes, resulting in a robust and reliable solution [7]. With the help of SciPy, scientists, engineers, and developers may tackle challenging problems in a variety of fields [8]. While the sparse matrix representation greatly affects the speed of the final application, selecting the right representation usually requires expert understanding as well as some experimentation [9].

Numerous ways to solve the integrated facility placement and production planning problems have emerged. The employed methods' numerical results demonstrated a noteworthy convergence in forecasting the production lines and revenue margins, demonstrating their precision and effectiveness in arriving at the best answers [2]. Wu et al. [4] proposed numerous more Dantzig-Wolfe decomposition and column generation (DWCG) strategies for the CFLPP issue. These ideas have led to advancements in technology for solving integrated facility placement and production planning problems. However, their capacity to resolve the CFLPP issue is still restricted. Romeijn et. al. suggested the branch-and-price method and the approximation algorithm [10].

Methodology:

In order to enhance decision-making and computational efficiency, this study investigates numerical optimization techniques for production scheduling in the cooking oil industry, making use of the Python SciPy and NumPy libraries. The core Python package for numerical computing is called NumPy. The NumPy n-dimensional array, which is its main data structure, is comparable to a Python MATLAB matrix. Developed on top of NumPy, SciPy is a set of modules that address a variety of computing challenges in fields such as statistics, linear algebra, linear programming, discrete equations, image processing, interpolation, optimization, sparse matrices, and clustering. Together, NumPy and SciPy encompass elements from different Toolboxes and the fundamental capabilities of the MATLAB product [8]. Python's versatility allows it to optimize model parameters, increase engineering design profitability, and meet quantitative objectives. Pyomo, GEKKO, and SciPy packages are utilized for optimization purposes [3]. The optimization was done in Python using the GEKKO package. GEKKO, a Python library, uses machine learning and optimization capabilities to solve complicated problems like: Mixed-integer and differential algebraic difficulties, Large-scale optimization (LP, QP, NLP, MILP, and MINL). As an object-oriented module, GEKKO supports local execution, making it an effective tool for optimization and machine learning tasks [2].

GEKKO and Pyomo packages were previously utilized to solve production planning challenges in inventory systems. In real-world optimization applications, strategies that ignore uncertainty have little practical relevance since data variations can severely impact performance. Min-max optimization is an effective method for developing robust solutions that account for uncertainty [7]. For this aim, GEKKO has complex programming codes that require a significant amount of time and money to execute.

The purpose of this work is to optimize inventory systems by identifying the optimal values for decision variables that maximize net profit using numerical linear programming techniques. Equations and objective functions are two of the many approaches used in this study to maximize the production of cooking oil, i.e, Dense matrices. Python has a package called SciPy. SciPy is a comprehensive library built on NumPy that provides a wide range of scientific computing modules for a variety of computational tasks, including linear algebra, linear programming, statistics, sparse matrices, clustering, differential equations, image processing, interpolation, and symbolic mathematics. With greater flexibility than that of widely used modeling languages for algebra, like Pyomo, CasADi, GAMS, AMPL, or JuMP, it is a general-purpose tool for modeling and solving optimization issues. Differential equation-constrained problems, or optimum control issues, are especially significant. Despite being created in self-contained C++, CasADi has fully working interfaces with Octave, MATLAB, and Python, providing the most convenient way to utilize it. Since its launch in late 2009, it has been effectively applied in a variety of sectors, such as process control, robotics, and aerospace, in addition to academic instruction.

We apply numerical optimization techniques for production planning in the cooking oil industry using Python's SciPy and NumPy modules to increase computational effectiveness and decision-making. Because supply chain operations are becoming more complicated and because there is a growing need for economical and sustainable production processes, advanced optimization approaches are increasingly necessary. The goal of this project is to create a robust computational framework by integrating SciPy's approaches for dense matrices and linear programming with NumPy's notable capabilities for solving nonlinear issues.

Here's the computational formulation that presents how to formulate a numerical optimization problem:

$$\min_{b \in B} f(b)$$

Where the set $B \subseteq R^n$ represents all feasible solutions, and $f: B \rightarrow R$ is the objective function to be minimized. It is standard practice to express such problems as minimization tasks. A maximization problem can be converted into a minimization problem using the following identity: $\max \{f(b): b \in B\} = -\min \{-f(b): b \in B\}$.

If the feasible set B is empty, the problem is termed *infeasible*, and the objective value is conventionally written as:

$$\text{Min } \{f(b): b \in B\} = +\infty.$$

On the other hand, if the function f is not bounded from below Whitin B, the problem is referred to as *unbounded*, and the objective value is given by:

$$\text{Min } \{f(b): b \in B\} = -\infty.$$

The goal of solving such a problem is to determine an *optimal solution*, that is, a point
$b^* \in B$ satisfying:

$$f(b^*) \leq f(b) \qquad \text{for all } b \in B.$$

An optimal solution is not necessarily unique. A point $\bar{b} \in B$ is considered a *locally optimal solution* if there exists a positive number $\varepsilon > 0$ such that:

$$f(\bar{b}) \leq f(b) \qquad \text{for all } b \in B \text{ such that } \|b - \bar{b}\| \leq \varepsilon.$$

Linear Programming with SciPy Optimization:

The process of selecting the best option from a range of feasible or constraint-compliant alternatives is known as optimization. Actually, Python can be used to increase the profitability of a future engineering design; model parameters should be optimized to better fit the data or accomplish other kinds of goals that use variables and equations to be expressed numerically [2]. For Example, SciPy, with its advanced mathematical features, including integration, optimization, special functions, and solvers for ordinary differential equations, which is based on the NumPy array architecture, raises the bar for scientific programming [9]. The following types of problems are associated with linear programming (LP), commonly referred to as linear optimization:

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n (1)$$

Subject to

$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$ $a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$

$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$ $x_1, x_2, \ldots, x_n \geq 0$

Combining the row vectors into a matrix of size m by n. A produces the condensed representation.

Production Planning Problem:

The planning problem starts with defining the client demand that the production plan must satisfy, while considering the limited production resources that cannot be carried over from one period to the next. The objective of the production scheduling problem is to maximize profit. To solve it, both resource availability and demand are assumed to be fixed and allocated accordingly. Typically, only a portion of future demand is known. Consequently, although forecasts of future demand may be uncertain, they are still relied upon. Revenue decreases when demand goes unfulfilled Whitin a given period. Stephen C. Graves formulates a production planning problem focused on maximizing profits after accounting for manufacturing costs, inventory holding costs, and lost sales. To solve the objective function, it is assumed that the coefficients, as well as the inequality and equality constraints, are known values [2][1][6].

Framework of Linear Programming (LP) for Production Planning Dilemma:

Three essential components of the optimization function must be defined to describe the general formulation of this problem. Since the objective is to determine the production

level for each month, there are twelve decision variables corresponding to the monthly production rates. Examining the first month, we obtain the following:

The cost of manufacture is c_1x_1. Assuming that the ending inventory level, x_1-p_1, is non-negative, the inventory-holding cost is equal to $h_1(x_1$-$p_1)$. The first month's total expense is therefore equal to $c_1x_1 + h_1(x_1 - p_1)$.

For the second month, the cost of manufacture is c_2x_2. The ending inventory is non-negative if it is $(x_1 - p_1 + x_2 - p_2)$. Since this month's beginning inventory level is $x_1 - p_1$, the production level is x_2, and p_2 is the demand for this month, the inventory-holding cost is $h_2(x_1 - p_1 + x_2 - p_2)$. Thus, the entire expense for the second month is equal to $c_2x_2 + h_2(x_1 - p_1 + x_2 - p_2)$.

The following conclusion is reached after further consideration: The full planning horizon's

Production cost, calculated using standard summation notation, is given by

$$\sum_{j=1}^{14}\left[c_jx_j + h_j\left(\sum_{i=1}^{j} x_i - \sum_{i=1}^{j} p_i\right)\right].$$

Since the third assumption states that deficiency is not permitted, we must ensure

$$\sum_{i=1}^{j} x_i - \sum_{i=1}^{j} p_i \geq 0 \quad \text{for j =1,2,3,14}$$

As a result, a set of 28 functional constraints is produced. Naturally, as they are output levels, we also require $x_j \geq 0$ for all $j = 1, 2, 14$.

The Complete Record Formulation:

Finally, we have formed the following formula:

$$\text{Minimize} \sum_{j=1}^{14}\left[c_jx_j + h_j\left(\sum_{i=1}^{j} x_i - \sum_{i=1}^{j} p_i\right)\right]$$

Subject to $\sum_{i=1}^{j} x_i - \sum_{i=1}^{j} p_i \geq 0$ for j =1,2,3,14

$x_j \geq 0$ for j=1,2,14

This linear program includes 14 non-negativity constraints, 14 inventory balance constraints (to avoid shortages), and a total of 14 decision variables. Altogether, it results in 28 functional constraints. The values of h_j, c_j, p_j, and m_j must be replaced with specific numerical values in practical applications [2][11][12].

Algorithm Methodology for Resolving the Issue:

First, define the production planning issue using the constraints and interval numbers in the target function. Determine the interval values in step two by using the required intervals as a guide. Convert the nondeterministic issue into a deterministic problem in step three. To resolve the problem, utilize the SciPy software suite. At the end, offer a conclusion. Utilizing two ingredients, P and Q, to produce the products 1 and 2, the cooking oil sector represents a fundamental production planning challenge. $P = 55$ and $Q = 72$ units are currently in stock. You will need the following to manufacture it: 12 units of Q and 5 units of P are needed to produce Product 1. Six units of Q and seven units of P are required to produce Product 2.

Eight units of Product 2 and a maximum of nine units of Product 1 are available. Product 1 can be sold for 200, while Product 2 can be sold for 250. To maximize profit is the aim of this production problem. To address this difficulty and perhaps develop a viable solution, we must determine the contour plot's limitations and label it with the set of feasible choices. Establish the upper and lower bounds as well as a realistic, impartial response. In certain instances, the upcoming subsection will address this issue.

Case 1: Equations and Objective

For small problems, using equations and an objective function is advantageous, as

they result in clear and easily modifiable optimization models.

Listing 1. Python code for solving the production planning problem using SciPy

```
from scipy. optimize import linprog
#The objective function's coefficients
c = [-200, -250]
# Constraint coefficients
A = [5 ,7], [12, 6]
b = [55,72]
# Bounds for the variables
x0 _bounds = (0, 9)
x1 _bounds = (0, 8)
# Solve the linear programming problem
res = linprog (c, A_ub =A, b_ub =b, bounds = [ x0_bounds, x1 _bounds])
# Print x1 and x2 's ideal values.
print (' Value of x1:', res. x [0])
print (' Value of x2:', res. x [1])
# Print the value of the maximum profit
print (' Profit Value:', - res. fun)
```

By using the previous Python code implemented with the SciPy optimization module, the following outcomes are obtained:

Maximum Profit Z	Value of x_1	Value of x_2
2033.3333333333335	3.2222222222222228	5.5555555555555555

Case 2: Dense Matrices

SciPy also supports dense matrix representations. For SciPy's linprog, the bounds input should be provided as a sequence of tuples, each specifying the lower and upper limits for a particular variable. We used NumPy arrays to define the constraints, limits, and coefficients of the objective function. After solving the linear programming problem with the linprog function, the solution remained unchanged.

Listing 2. Python code using dense matrix bounds with SciPy

```
from scipy. Optimize import linprog import numpy as np
# Coefficients of the objective function c = np. array ([-200, -250])
# Coefficients of the constraints A = np. array ([5, 7], [12, 6])
b = np. array ([55, 72])
# Bounds for the variables x0 _bounds = (0, 9)
x1 _bounds = (0, 8)
#  Define the bounds
bounds = [ x0_bounds, x1 _bounds]
# Solve the linear programming problem
res = linprog (c, A_ub =A, b_ub =b, bounds= bounds)
# Print the optimal values of x1 and x2 print (' Value of x1:', res. x [0])
print (' Value of x2:', res. x [1])
```

The following outcomes were obtained by using the previous code:

Profit Value Z	Value of x_1	Value of x_2
2033.3333333333335	3.2222222222222228	5.5555555555555555

Results:

In this section, some results are given regarding production planning problems.

Mathematical Results Using a Contour Plot:

A contour plot can be used to identify the optimal solution. In this example, the

production of product 1 must be greater than 0 but less than 9, while the production of product 2 must be greater than 0 but less than 8. For products 1 and 2, only 55 units of ingredient P and 72 units of ingredient Q are available. The optimal solution is determined using the contour plot of the previously defined objective function.

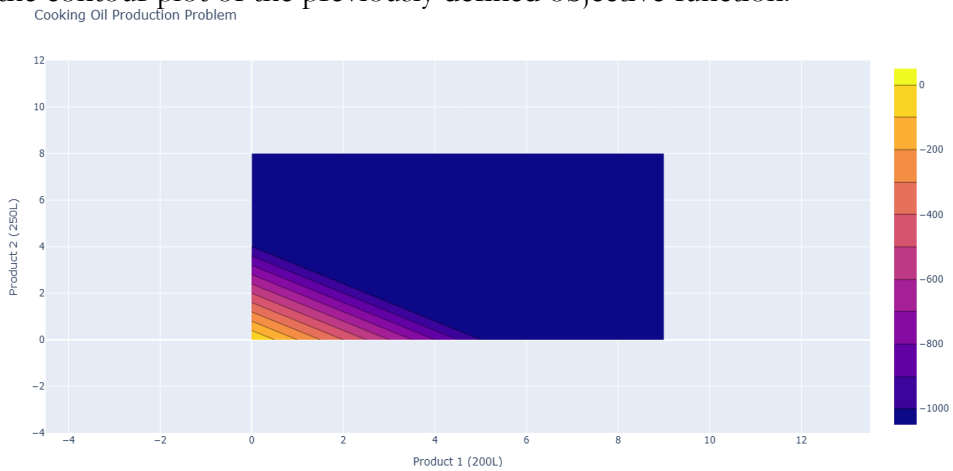

Figure 1. Contour plot

Listing 3. Contour Plot for Cooking Oil Production Problem using SciPy and Plotly

```
import numpy as np
from scipy. Optimize import, minimize import plotly. graph_objects as go
# Define the objective function def objective (x):
return -(200* x [0] + 250* x [1])
# Define the constraints
def constraint1 (x):
return 5* x [0] + 7* x [1] - 55
def constraint2 (x):
return 12* x [0] + 6* x [1] - 72
#  Define the bounds
bounds = [(0, 9), (0, 8)]
x0 = [1, 1]
# Define the constraints dictionary
cons = ({' type':' ineq',' fun': constraint1},'
{' type':' ineq', fun': constraint2})
# Run the optimization
res = minimize (objective, x0, method =' SLSQP', bounds=bounds, constraints=
cons)
# Generate a grid of x1 and x2 values
x1 = np. linspace (0, 9, 100)
x2 = np. linspace (0, 8, 100) X1, X2 = np. meshgrid (x1, x2)
# Calculate the objective function for each point in the grid
Z = -(200* X1 + 250* X2)
# Create a contour plot
fig = go. Figure (data = [ go. Contour (x=x1, y=x2, z=Z, contours= dict(start =
-1000, end =0, size =100))])
fig. update_layout (title =' Cooking Oil Production Problem',
xaxis_title =' x1 (200 L)',
yaxis_title =' x2 (250 L)')
fig. show ()
```

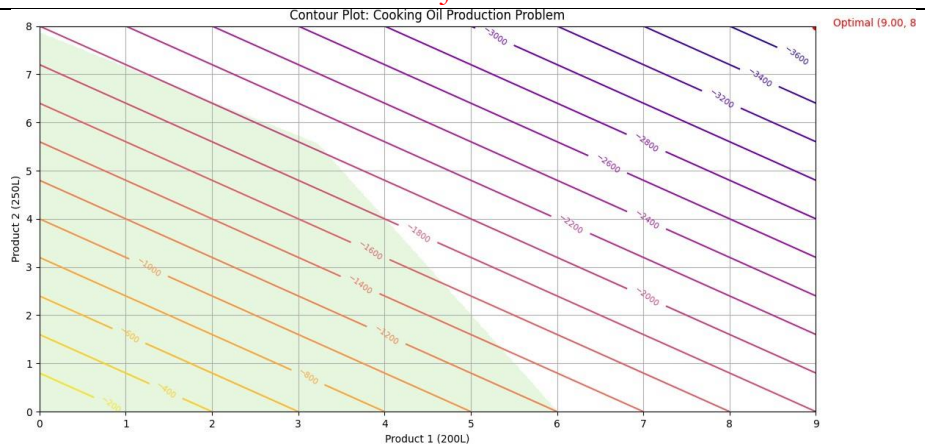

Figure 2. SciPy's Cooking Oil contour plot

This is the graphical representation of a contour plot for SciPy code. It gives us a better understanding as contour lines for the profit functions are shown in it, and the feasible region is shaded in it.

Hence, the optimal result is,

Profit Value	Value of x_1	Value of x_2
2033.3333333333335	3.2222222222222228	5.5555555555555555

Discussion:

The presented contour plots (Figures 1 and 2) illustrate the optimization framework applied to the cooking oil production problem, modeled as a linear programming task in Python. Both visualizations depict the relationship between the two decision variables. x_1 (200L batches) and x_2 (250L batches) subject to production constraints and the objective of maximizing profit.

Figure 1 complements this analysis by providing a heatmap-style contour visualization, where darker regions denote lower objective function values and lighter regions represent higher ones. This figure reinforces the earlier finding that maximum profit occurs toward the upper boundary of the feasible set. The gradient of the contour lines reflects the trade-off between producing x_1 and Product x_2, showing how increases in one product can only be achieved at the expense of the other Whitin the restricted resource framework.

Figure 2, the contour plot provides a clear representation of the feasible region (highlighted in green) formed by the intersection of linear constraints. Each contour line corresponds to a constant objective function value, with lines further from the origin indicating higher profit levels. The optimal solution is marked explicitly at the boundary of the feasible region. This location confirms that the highest achievable profit is obtained when production is pushed to the limit of the constraints. The visualization demonstrates the typical behavior of linear optimization problems, where the optimal solution coincides with a corner point of the feasible region.

Taken together, these figures confirm the robustness of the optimization results. The feasible region and the profit contours are consistent with the theoretical underpinnings of linear programming. Importantly, both plots highlight the sensitivity of the solution to the imposed constraints: relaxing any resource limit would expand the feasible region and potentially shift the optimal point further outward, leading to higher profitability. Conversely, tightening constraints would shrink the feasible area and reduce the maximum achievable profit. From a practical perspective, the results provide decision-makers in production planning with visual and quantitative evidence for allocating resources between competing products. The figures demonstrate that linear programming not only identifies the numerical optimal solution but also provides a graphical means to understand constraint interactions and

trade-offs. Such visualizations can enhance managerial decision-making by offering insight into how changes in production capacity, raw material availability, or demand could affect the optimal production strategy.

Conclusion:

The purpose of this study was to use linear programming (LP) to find the most effective production planning method for the cooking oil business under nominal limitations. In contrast to the earliest studies that used the GEKKO optimization module, this work solved the LP problem using Python's SciPy package. According to the results, SciPy is a more efficacious option for resolving organized linear programming issues than GEKKO since it produced optimum solutions in a slightly less amount of computational time. In addition, contour plots were created to provide a more exhaustive understanding of select variables by graphically representing executable areas and perfect solutions. The solutions' accuracy and consistency, as well as their magnified execution speed, attest to SciPy's relevance for real-world industrial production planning applications, especially those where speedy and trustworthy optimization is vital.

References:

[1] S. Torkaman, S. M. T. F. Ghomi, and B. Karimi, "Hybrid simulated annealing and genetic approach for solving a multi-stage production planning with sequence-dependent setups in a closed-loop supply chain," *Appl. Soft Comput.*, vol. 71, pp. 1085–1104, 2018, doi: https://doi.org/10.1016/j.asoc.2017.10.019.

[2] C. A. Studies, "Numerical Optimization Approach for Solving Production Planning Problem Using Python language," *Cent. ASIAN J. Math. THEORY Comput. Sci.*, Jan. 2022, Accessed: Oct. 01, 2025. [Online]. Available: https://www.academia.edu/86744677/Numerical_Optimization_Approach_for_Solving_Production_Planning_Problem_Using_Python_language

[3] U. M. Diwekar, "Introduction to Applied Optimization," *Springer*, vol. 22, 2020, doi: 10.1007/978-3-030-55404-0.

[4] H. Missbauer and R. Uzsoy, "Optimization Models of Production Planning Problems," *Int. Ser. Oper. Res. Manag. Sci.*, vol. 151, pp. 437–507, 2011, doi: 10.1007/978-1-4419-6485-4_16.

[5] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019, doi: 10.1007/S12532-018-0139-4/METRICS.

[6] C. Brown, A. Abdelfattah, S. Tomov, and J. Dongarra, "Design, Optimization, and Benchmarking of Dense Linear Algebra Algorithms on AMD GPUs," *2020 IEEE High Perform. Extrem. Comput. Conf. HPEC 2020*, Sep. 2020, doi: 10.1109/HPEC43674.2020.9286214.

[7] M. Antoniou and G. Papa, "Evaluation of Parallel Hierarchical Differential Evolution for Min-Max Optimization Problems Using SciPy," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13627 LNCS, pp. 84–98, 2022, doi: 10.1007/978-3-031-21094-5_7.

[8] A. Danial, "Python for MATLAB Development: Extend MATLAB with 300,000+ Modules from the Python Package Index," *Python MATLAB Dev. Extend MATLAB with 300,000+ Modul. from Python Packag. Index*, pp. 1–700, Mar. 2022, doi: 10.1007/978-1-4842-7223-7/COVER.

[9] S. Chen, J. Fang, D. Chen, C. Xu, and Z. Wang, "Adaptive Optimization of Sparse Matrix-Vector Multiplication on Emerging Many-Core Architectures," *Proc. - 20th Int. Conf. High Perform. Comput. Commun. 16th Int. Conf. Smart City 4th Int. Conf. Data Sci.*

Syst. HPCC/SmartCity/DSS 2018, pp. 649–658, Jan. 2019, doi: 10.1109/HPCC/SMARTCITY/DSS.2018.00116.

[10] L. H. Tao Wu, "A supervised learning-driven heuristic for solving the facility location and production planning problem," *Eur. J. Oper. Res.*, vol. 301, no. 2, pp. 785–796, 2022, doi: https://doi.org/10.1016/j.ejor.2021.11.020.

[11] N. S. Gates, D. C. Hill, B. W. Billings, K. M. Powell, and J. D. Hedengren, "Benchmarks for Grid Energy Management with Python Gekko," *Proc. IEEE Conf. Decis. Control*, vol. 2021-December, pp. 4868–4874, 2021, doi: 10.1109/CDC45484.2021.9683406.

[12] E. B. M. Bashier, "Practical Numerical and Scientific Computing with MATLAB® and Python," *Routledge*, 2020, [Online]. Available: https://www.routledge.com/Practical-Numerical-and-Scientific-Computing-with-MATLABr-and-Python/Bashier/p/book/9780367076696?srsltid=AfmBOorc1b0dxC_l-AQc0OQfqkrLz9D18HfYQQEAJPXljb-swQJylGct