Scientific Report

Numerical Modeling and Analysis of Bag Production Trends for EGIER Warehouse Capacity Planning

Robben Wijanathan
NIM: 2802461681[1]

[1]*Computer Science & Mathematics, Binus University, Jakarta, Indonesia*

Lecturer: Dr. ALFI YUSROTIS ZAKIYYAH, S.Pd., M.Si.

Abstract—This report analyzes EGIER's monthly bag production using mathematical modeling and numerical methods. Nonlinear trend fitting and numerical approximation are applied to forecast warehouse capacity needs. The results predict when storage expansion will be required and provide insights into production growth through differentiation and integration.

Keywords—*scientific computing, numerical methods, polynomial regression, numerical analysis, trend analysis*

1. Introduction

1.1. Background

EGIER, a Bandung-based outdoor equipment manufacturer, produces various types of bags for local and international markets. Monthly production data from January 2018 to December 2023 is analyzed to observe manufacturing trends.

Since the warehouse capacity is limited to 25,000 bags per month, forecasting production helps determine when expansion is needed. A polynomial regression model is used to capture nonlinear patterns, later analyzed through numerical differentiation and integration.

1.2. Objectives

- To model EGIER's monthly bag production using polynomial regression.
- To forecast when warehouse expansion will be required.
- To analyze production rate and total output using numerical methods.

2. Methods

2.1. The Dataset

The EGIER dataset contains monthly bag production records from January 2018 to December 2023, totaling 144 data points (M1–M144). The data, stored in `aol_data.csv`, lists months as labels (M1, M2, …, M144). Before analysis, it was cleaned and reshaped into a tabular format where each row represents a month and its production.

```
data = pd.read_csv('data/aol_data.csv')
data = data.transpose()
data = data.reset_index()
data.columns = ['Month', 'Production']
data['Month'] = data['Month'].str.replace('M', '').
    ↪ astype(int)
x = np.array(data['Month'], dtype=float)
y = np.array(data['Production'], dtype=float)
data
```

Data Reading & Cleansing

```
    Month   Production
0   1       1863
1   2       1614
2   3       2570
... ...     ...
141 142     17033
142 143     16896
143 144     17689
144 rows x 2 columns
```

Output

Scatter Plot as Visualisation

```
plt.figure(figsize=(12,6))
plt.scatter(x, y, c=x, cmap='winter', edgecolors="black",
    ↪ s=20, alpha=1)
plt.title("EGIER Bag Production (January 2018 - December
    ↪ 2023)", fontsize=14, fontweight='bold')
plt.xlabel("Month", fontsize=10)
plt.ylabel("Production", fontsize=10)
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

Scatter Plot for the Dataset

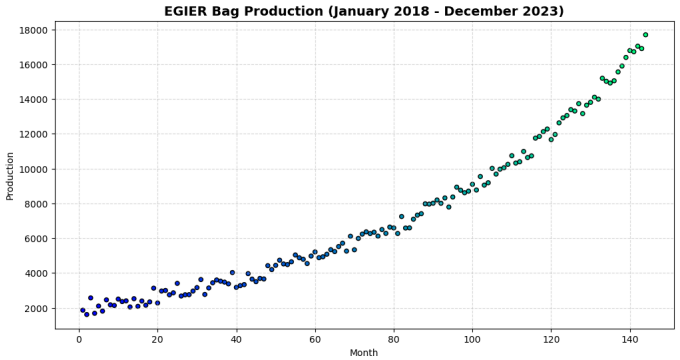

Figure 1. EGIER Bag Production (January 2018 - December 2023)

The data shows a curved pattern, resembling a polynomial relationship. Therefore, polynomial regression is used to model and analyze the production trend.

2.2. Least Squares Polynomial Regression

Given n data points (x_i, y_i), where x_i denotes the **month** and y_i the **production**, a polynomial of degree m is fitted as:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

In **matrix form**:

$$Y = MV + \varepsilon$$

where

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad M = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}, \quad V = \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix}$$

and ε is the residual vector.

The least squares approach minimizes:

$$S = \|Y - MV\|^2 = (Y - MV)^T (Y - MV)$$

Setting $\frac{\partial S}{\partial V} = 0$ yields the **normal equation**:

$$M^T M V = M^T Y$$

Thus, the least-squares estimate of the polynomial coefficients is:

$$\boxed{V = (M^T M)^{-1} M^T Y}$$

which provides the optimal fit minimizing the total squared error.

2.3. Accuracy of the Regression

The polynomial model behaves like a finite **Taylor Series**, where higher-order terms improve local accuracy. Model performance is measured using the coefficient of determination (R^2):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

A higher R^2 indicates a stronger agreement between predicted and actual production values, validating the reliability of the regression model.

2.4. Predicting Warehouse Capacity

To estimate when EGIER's production reaches the warehouse limit of 25,000 bags, the cubic regression model is used to solve for the month x where $f(x) = 25{,}000$. This is formulated as a **root-finding problem**.

Because the equation cannot be solved analytically, the **Newton–Raphson method** is employed to iteratively approximate the root:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where $f'(x)$ is the derivative of the cubic polynomial. The iteration continues until successive estimates differ negligibly, yielding the month when production is projected to reach capacity. This method ensures rapid convergence for smooth, differentiable functions like the cubic model.

2.5. Numerical Differentiation for Rate of Change

Numerical differentiation approximates a function's derivative from discrete data points. In this study, it is used to estimate the rate of change of EGIER's monthly production over time. Given evenly spaced monthly data, the derivative $\frac{dy}{dx}$ is calculated using the **finite difference method** with $h = 1$ month:

$$f'(x_i) \approx \begin{cases} \dfrac{f(x_{i+1}) - f(x_i)}{h}, & i = 0 \quad \text{(forward difference)} \\ \dfrac{f(x_{i+1}) - f(x_{i-1})}{2h}, & 1 \le i \le n-1 \quad \text{(central difference)} \\ \dfrac{f(x_i) - f(x_{i-1})}{h}, & i = n \quad \text{(backward difference)} \end{cases}$$

The **central difference** offers higher accuracy by averaging forward and backward slopes, while the boundary points use forward or backward differences. This method effectively estimates production rate changes without requiring an explicit analytical model.

2.6. Numerical Integration for Total Production

To estimate the total bag production from January 2018 to December 2023, the **Numerical Integration** method is applied to approximate the area under the production curve, representing cumulative output over time.

The **Trapezoidal Rule** is employed for its balance between accuracy and simplicity:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

where $f(x_i)$ is the production at month i and $h = 1$ month.

This provides an estimate of total production, later compared with the actual dataset sum to assess accuracy, reflecting the **Riemann integral** concept of integration as accumulated change over time.

3. Results & Discussion

3.1. Least Square Fit Polynomial Regression

We model production with a degree-d polynomial

$$\hat{y}(x) = a_0 + a_1 x + \cdots + a_d x^d,$$

```python
# Polynomial Regression

for i in range(1,4):

    plt.figure(figsize=(24,12))
    plt.subplot(2, 2, i)
    plt.scatter(x, y, c=x, cmap='winter', edgecolors="
    ↪ black", s=20, alpha=0.4)

    y_est = np.polyfit(x, y, i)

    eq = "y = " + " + ".join([f"{coef:.6f}x^{len(y_est)-
    ↪ j-1}" for j, coef in enumerate(y_est)])
    print(eq)

    plt.plot(x, np.polyval(y_est,x), color='red',
    ↪ linewidth=2)
    plt.xticks(
        ticks=np.linspace(1, 144, 24),
        labels=[f"Q{(i % 4) + 1} Y{2018 + (i // 4)}" for
    ↪  i in range(24)],
        rotation=45,
        fontsize=8
    )

    plt.title(f"Polynomial Order {i}", fontsize=14,
    ↪ fontweight='bold')
    plt.xlabel("Month", fontsize=10)
    plt.ylabel("Production", fontsize=10)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.show()
```

Polynomial Regression

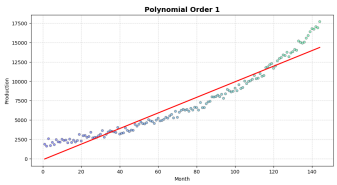

Figure 2. Polynomial Order 1

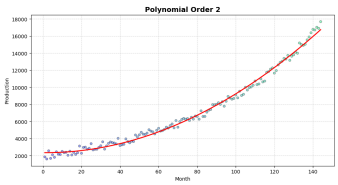

Figure 3. Polynomial Order 2

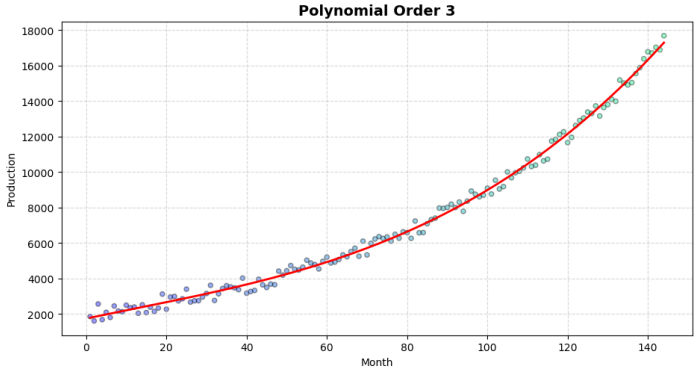

Figure 4. Polynomial Order 3

3.2. Polynomial Regression Model & Accuracy

From the cubic polynomial regression, the estimated coefficients are:

$$[a_3, a_2, a_1, a_0] = [0.003863, -0.134357, 47.223553, 1748.506723]$$

Thus, the regression model can be expressed as:

$$\boxed{y = 0.003863x^3 - 0.134357x^2 + 47.223553x + 1748.506723}$$

Scatter Plot as Visualisation

```python
# Polynomial coefficients (highest degree first)
coeffs = [0.003863, -0.134357, 47.223553, 1748.506723]

y_pred = np.polyval(coeffs, x)

plt.figure(figsize=(12,6))
plt.scatter(x, y, c=x, cmap='winter', edgecolors="black", s=20, alpha=0.6, label='Actual Data')
plt.plot(x, y_pred, color='red', linewidth=4, label='3th-Degree Polynomial Fit')
plt.xticks(
    ticks=np.linspace(1, 144, 24),
    labels=[f"Q{(i % 4) + 1} Y{2018 + (i // 4)}" for i in range(24)],
    rotation=45,
    fontsize=8
)
plt.title("EGIER Bag Production (January 2018 - December 2023) w/ Polynomial Regression", fontsize=14, fontweight='bold')
plt.xlabel("Month", fontsize=10)
plt.ylabel("Production", fontsize=10)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()
```

Scatter Plot of the Regression

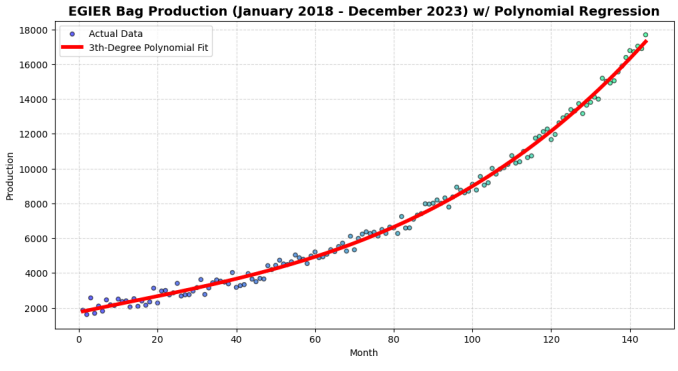

Figure 5. EGIER Bag Production (January 2018 - December 2023) Polynomial Regression

Evaluating the accuracy of the regression, the Mean Squared Error (MSE) and the coefficient of determination (R^2):

```python
# Actual data
y_true = np.array(data['Production'])

y_pred = np.polyval(coeffs, x)

mse = np.mean((y_true - y_pred)**2)
r2 = 1 - np.sum((y_true - y_pred)**2) / np.sum((y_true - np.mean(y_true))**2)

print("Mean Squared Error (MSE):", mse)
print("R^2 =", r2)
```

Accuracy of the Cubic Regression Model

```
Mean Squared Error (MSE): 83195.22187393636
R^2 = 0.9955827569999465
```

Output

The regression achieved a Mean Squared Error (MSE) of about 83,195, showing minimal deviation between predicted and actual values. The coefficient of determination $R^2 = 0.9956$ indicates that 99.56% of the data's variance is explained by the model.

The 3rd-order polynomial provides an excellent fit, accurately capturing both the trend and minor fluctuations in production, with its higher-order terms enhancing approximation accuracy.

3.3. Predicting Warehouse Capacity Using Newton–Raphson Method

To estimate when EGIER's production reaches the warehouse limit of 25,000 bags, the cubic regression model is reformulated as a root-finding problem:

$$f(x) = 0.003863x^3 - 0.134357x^2 + 47.223553x + 1748.506723 - 25000 = 0$$

The **Newton–Raphson method** iteratively updates the estimate of x as:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where $f'(x)$ is the derivative of the regression function. The iterations continue until the change between successive estimates is below the tolerance (10^{-6}), yielding the month when production first equals the warehouse capacity.

```python
def f(x):
    return np.polyval(coeffs, x) - 25000

def f_prime(x):
    dcoeffs = np.polyder(coeffs)
    return np.polyval(dcoeffs, x)

x = 140
tol = 1e-6

# Newton-Raphson iteration
for i in range(20):
    x_new = x - f(x) / f_prime(x)
    if abs(x_new - x) < tol:
        break
    x = x_new

print(f"Predicted month when production reaches 25,000: {x:.2f}")
print(f"Recommended start for new warehouse construction: {x - 13:.2f}")
```

Newthon-Raphson Method for Root-Finding

```
Predicted month when production reaches 25,000: 170.38
Recommended start for new warehouse construction: 157.38
```

Output

From the computation, the predicted month when EGIER's production reaches 25,000 bags is approximately:

$$\boxed{x = 170.38}$$

Since constructing a new warehouse requires at least 13 months of preparation, the company should begin the expansion around:

$$\boxed{x = 157.38}$$

3.4. Numerical Differentiation for Rate of Change

Numerical differentiation was applied to the production data to estimate the rate of change over time. The derivative at each point was approximated using the finite difference method, with the **central difference** used for interior points and **forward/backward differences** at the boundaries for accuracy.

```
x = np.array(data['Month'])
y = np.array(data['Production'])

# Step size
h = x[1] - x[0]

dy_dx = np.zeros_like(y, dtype=float)

# Forward difference for first point
dy_dx[0] = (y[1] - y[0]) / h

# Central difference for interior points
for i in range(1, len(y)-1):
    dy_dx[i] = (y[i+1] - y[i-1]) / (2 * h)

# Backward difference for last point
dy_dx[-1] = (y[-1] - y[-2]) / h

max_month = x[np.argmax(dy_dx)]
min_month = x[np.argmin(dy_dx)]

print(f"Steepest increase at month {max_month}: {dy_dx.max():.2f} bags/month")
print(f"Steepest decrease at month {min_month}: {dy_dx.min():.2f} bags/month")
```

Numerical Differentiation for Rate of Change

```
Steepest increase at month 144: 793.00 bags/month
Steepest decrease at month 40: -376.50 bags/month
```

Output

The derivative values show how rapidly production changes each month. The steepest increase occurred at **Month 144** with about **793 bags/month**, while the sharpest decline occurred at **Month 40** with around **-376.5 bags/month**, marking the periods of greatest change in EGIER's production rate.

3.5. Numerical Integration for Total Production

The Trapezoidal Rule was used to approximate the total cumulative production between January 2018 and December 2023. This numerical integration method divides the production curve into trapezoids of equal width ($h = 1$) and sums their areas according to:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

```
# Step size
h = x[1] - x[0]

# Trapezoidal Rule Formula
total_area = (h / 2) * (y[0] + 2 * np.sum(y[1:-1]) + y[-1])
print(f"Estimated total production (2018 - 2023): {total_area:.2f} bags")
```

Numerical Integration for Total Production

```
Estimated total production (2018 - 2023): 1020969.00 bags
```

Output

Trapezoidal Rule Integration Method Plot

```
plt.figure(figsize=(14,6))
plt.plot(x, y, 'b-', linewidth=2, label='Production Curve')
plt.title("Riemann - Trapezoidal Approximation of Total Production", fontsize=14, fontweight='bold')
plt.xlabel("Month")
plt.ylabel("Production (bags)")
plt.grid(True, linestyle='--', alpha=0.4)

for i in range(len(x)-1):
    plt.fill_between([x[i], x[i+1]], [y[i], y[i+1]], color='skyblue', alpha=0.4)

plt.legend()
plt.show()
```

Trapezoidal Rule Integration Method Plot

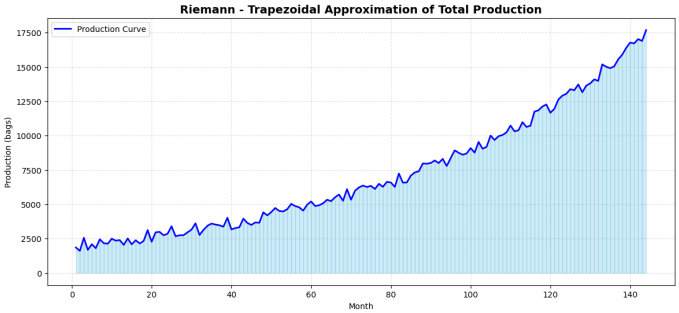

Figure 6. Trapezoidal Rule Integration Method

The shaded regions in the figure represent the estimated area under the curve, corresponding to the total production.

Comparing to the Actual Total Production

```
# Actual total from dataset
total_production = np.sum(y)
print("Actual Total Production =", total_production)

# Compare trapezoidal estimate vs actual sum
error = (total_production - total_area) / total_production * 100

print(f"Error = {error:.2f} %")
print(f"Accuracy = {100 - error:.2f} %")
```

Trapezoidal Rule Integration Method Plot

```
Actual Total Production = 1030745.0
Error = 0.95 %
Accuracy = 99.05 %
```

Output

Compared to the actual total of 1,030,745 bags, the trapezoidal estimate showed an error of only **0.95%**, corresponding to an accuracy of about **99.05%**. This demonstrates that the Riemann–Trapezoidal method provides a highly reliable approximation of EGIER's total production.

References

[1] Jaan Kiusalaas, *Numerical Methods in Engineering with Python 3*, Cambridge: Cambridge University Press, 2013.

[2] Kong, Q., Siauw, T., and Bayen, A. M., *Python Programming and Numerical Methods: A Guide for Engineers and Scientists*, Amsterdam: Academic Press, Elsevier, 2021.