Industrial Engineering Journal
ISSN: 0970-2555
Volume : 55, Issue 01 : 2026

COMPUTATIONAL PROGRAMS FOR LINE PERFORMANCE EVALUATION IN A HIGH-VOLUME TWRA MANUFACTURING LINE: IDLE-TIME AND CLASSICAL TIME-BASED APPROACHES

Mrs. Ashwini V., Research Scholar, Department of Mechanical Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, Karnataka, India.
Dr. Paul Vizhian S., Professor, Department of Mechanical Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, Karnataka, India.
Mrs. Rathika M., Assistant Professor, Department of Mechanical Engineering, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka, India.
Mrs. Gomathi R., Assistant Professor, Department of Civil Engineering, Sapthagiri Engineering College, NPS University, Bangalore, Karnataka, India.
Mr. Pawan Kumar S. S., Research Scholar, Department of Mechanical Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, Karnataka, India.

ABSTRACT

High-volume manufacturing lines require fast and repeatable evaluation of line balance, idle loss, and bottleneck-driven imbalance to support improvement decisions on the shop floor. This paper presents Python-based computational programs developed for a Two-Wheeler Rocker Arm (TWRA) manufacturing line. The programs implement two complementary evaluation approaches: (i) an idle-time-driven method that directly quantifies unused time and reports utilization-oriented indicators, and (ii) a classical time-based approach that derives station/cell processing time from production output and evaluates line balance relative to the bottleneck cycle time. Using actual production data from pre-improvement and post-improvement conditions, the programs compute and report key indicators such as Idle Time Efficiency (ITE), Line Balance Efficiency (LBE), Overall Utilization Efficiency (OUE), and (where required) quality-adjusted OEE. Results confirm strong reduction in idle losses after improvement: LBE increased from 97.619% to 99.7487% in the station-level idle-time program. In cell-level analytical evaluation (six manufacturing cells), LBE increased from 97.619% (pre) to 99.9206% (post), with Overall Equipment Effectiveness (OEE) reported for assumed rejection rates to show quality sensitivity. The computational framework provides a transparent, repeatable and industry-ready approach for line performance evaluation aligned with digital decision support needs.

Keywords: Line Performance Evaluation, Line Balancing, Idle Time Method, Bottleneck Analysis, Line Balance Efficiency, Python, TWRA.

1.0 INTRODUCTION

Modern production systems operate under variable demand, multi-workstation constraints, and continuous pressure to improve throughput and utilization. In such environments, routine engineering calculations (line balance checks, idle loss estimation, bottleneck identification, and scenario comparison) are frequently repeated. Manual calculation is slow and error-prone, and repeated recalculation under new operating scenarios becomes impractical.
Digital tools and Enterprise Resource Planning (ERP)-style analytical modules are therefore increasingly used to standardize decision support on the shop floor. Translating basic production parameters (available time, demand, output rates, and idle time) into computational models enables consistent evaluation, reduces human error, and permits rapid comparison between operating

conditions and improvement states. The computational programs presented here were implemented in Python to provide transparent calculations and reproducible outputs for the TWRA manufacturing line. This approach supports Industry 4.0 objectives through data-based evaluation while remaining interpretable and operator-friendly for human-centered decision making (Industry 5.0 perspective) [1–3].

Novelty and Contribution of the Study:

The novelty of this work lies in the use of a simple idle-time based computational framework to evaluate line performance using actual shop-floor data. The study demonstrates a practical workstation-level and manufacturing-cell-level evaluation without reliance on optimization or simulation models. The proposed Python programs provide a transparent and repeatable method for quantifying line balance and utilization in high-volume manufacturing systems.

2.0 OVERVIEW OF THE COMPUTATIONAL FRAMEWORK

The computational framework is organized into two families of evaluation programs:

1. **Idle-Time Method Programs (Utilization-Oriented):** These programs take measured idle time as the primary input and compute ITE, LBE, and OUE for fast before–after assessment in continuous improvement settings. This approach is particularly suitable when the line configuration is known and the focus is on reducing idle loss rather than redesigning precedence allocations [4–6].
2. **Classical Time-Based Programs (Bottleneck-Oriented):** These programs compute time per unit from station or cell output, identify the bottleneck cycle time, and compute classical line balance efficiency relative to the bottleneck. Such time-based logic is widely used for bottleneck identification and balance evaluation in line analysis and assembly-line balancing literature [7–10].

For analytical clarity and practical interpretation, machines can be grouped into manufacturing cells representing major stages (e.g., Soft Machining, Heat Treatment, Hard Machining, Chrome Coating, Finishing, and Inspection & Packing). This grouping supports line-level diagnosis without requiring detailed precedence modelling.

3.0 METHODS AND PERFORMANCE INDICATORS

3.1 Idle-Time Method (Station-Level or Cell-Level)

Let:

- T_{avail} = Available production time per day (s).
- T_{idle} = Total idle time per day (s).
- N = Number of stations (or cells).
- Q_d = Demand per day (units).

Idle time per station, $T_{idle,\ station} = T_{idle}/N$ ….. Equation 1

This is a standard normalization used in work measurement and line efficiency reporting. [4, 5]

TAKT time, $CT_{takt} = T_{avail} / Q_d$ ….. Equation 2

TAKT is a foundational demand-paced production concept in lean systems. [1, 2]

Idle Time Efficiency (ITE), ITE (%) = $(T_{idle} / T_{avail}) \times 100$ ….. Equation 3

This expresses the percentage of available time lost to idle time [4–6].

Line Balance Efficiency (LBE), LBE (%) = 100−ITE (%) ….. Equation 4

LBE is reported here as the complement of idle loss under the idle-time accounting used for this study [4–6].

Overall Utilization Efficiency (OUE), OUE (%) ≈ LBE (%) ….. Equation 5

When losses are not separated into availability, performance, and quality components, OUE is commonly reported as a utilization proxy consistent with LBE under the given assumptions [5, 6].

Quality-adjusted OEE (optional, for interpretation): When a rejection rate r is assumed or measured, a simplified quality factor Q= 1- r can be used:

OEE (%) = OUE (%) × (1- r) ….. Equation 6

This form follows the standard OEE definition (Availability × Performance × Quality) when only a quality adjustment is applied to utilization for reporting sensitivity [11, 12].

3.2 Classical Time-Based Line Balance (bottleneck cycle time method)

Let:

- Qi = output per day at station/cell, i (units/day)

Processing time per unit, $t_i = (T_{avail} / Qi)$ ….. Equation 7

This converts observed output into implied unit processing time [7–9].

Bottleneck cycle time, $CT=\max(t_i)$ ….. Equation 8

The bottleneck governs capacity in a serial flow system; identifying the maximum cycle time is consistent with bottleneck theory and constraints-based reasoning [8, 9].

Classical Line Balance Efficiency, LBE (%) = $(N / (CT\sum ti)) \times 100$ ….. Equation 9

This ratio evaluates how close the line is to the bottleneck-normalized ideal [7, 10].

Idle Time Efficiency (classical form), ITE (%) =100−LBE (%) ….. Equation 10

These Reported as imbalance loss relative to the bottleneck-normalized line [7, 10].

4.0 CASE STUDY INPUTS (TWR, MANUFACTURING LINE)

The programs were executed using actual shop-floor data. Common time and demand inputs used in the reported idle-time computations are:

- T_{avail} = 75,600 s/day.
- Q_d = 7,500 units/day.
- Therefore CT_{takt} = 10.08 s/unit.

Two idle-time operating conditions (station-level program) are:

- **Pre-improvement:** N =20, T_{idle} =1800 s/day.
- **Post-improvement:** N=19, T_{idle} =190 s/day.

In the cell-level analytical evaluation, the line is represented by **six manufacturing cells**, and idle time is evaluated at the cell level with reported cases:

- **Before improvement:** N=6, T_{idle} =1800 s/day.
- **After improvement:** N=6, T_{idle} =60 s/day (reported for both "without OT" and "with OT" in the pasted dataset).

5.0 RESULT AND DISCUSSION

5.1 Idle-Time Method (Station-Level Computational Program)

Using the station-level inputs:

- **Pre-improvement:**
  - ITE = (1800/75600)×100 = 2.381%
  - LBE = 97.619%
  - OUE ≈ 97.619%
- **Post-improvement:**
  - ITE = (190/75600)×100 = 0.2513%
  - LBE = 99.7487%
  - OUE ≈ 99.7487%

5.2 Idle-Time Method (Cell-Level Classical Analytical Evaluation)

With six cells and T_{idle} measured/assumed for the cases:

- **Before improvement:**
  - ITE = 2.381%
  - LBE = 97.619%
  - OUE ≈ 97.619%
- **After improvement:** T_{idle} = 60 s/day
  - ITE = (60/75600)×100 = 0.0794%
  - LBE = 99.9206%
  - OUE ≈ 99.9206%

To show quality sensitivity, OEE was computed for assumed rejection rates:

- At 0% rejection: OEE = 99.9206%
- At 5% rejection: OEE = 9206×0.95 = 94.9246%

Interpretation: This view is useful when the line is discussed at a stage/cell level rather than station-by-station. It also demonstrates that quality losses can be layered onto utilization reporting when required, consistent with standard OEE interpretation [11, 12].

Interpretation: The idle loss reduced by 1610 s/day, and utilization-based balance increased by 2.1296 percentage points, indicating that improvement actions reduced nonproductive waiting and imbalance-related losses. The result is consistent with the expected benefit of stabilizing work content and reducing underutilization in multi-station lines [4–6].

Discussion: The Station-Level Idle-Time Program Is Directly Tied To Workstation Count And Measured Idle Time; It Is Therefore Well Suited For "Before versus After" Improvement Tracking When The Number Of Stations Changes (20 To 19). The Cell-Level Analytical Evaluation, On The Other Hand, Provides A Macro-Level Interpretation When The Line Is Communicated As Major Process Stages. Both Can Coexist Because They Answer Different Reporting Needs: Detailed Operational Diagnostics Versus Stage-Level Explanation.

6.0 CONCLUSION

Python-based computational programs were developed to support repeatable line performance evaluation in a TWRA manufacturing line. Idle-time-based computation quantified substantial reduction in idle loss after improvement, with LBE rising from 97.619% to 99.7487% in station-level reporting, and to 99.9206% in cell-level analytical reporting. The programs convert routine industrial engineering calculations into transparent, reusable modules suitable for continuous improvement studies and decision-support reporting in high-volume manufacturing environments.

7.0 FUTURE WORK

Future extensions can improve the framework in three directions:

1. **Full classical time-based line balancing module integration** using station/cell output datasets to compute t_i, bottleneck CT, and classical LBE in a single standardized report [7–10].
2. **Automated data ingestion** from Manufacturing Execution System/Enterprise Resource Planning (MES/ERP) exports to eliminate manual entry and support periodic performance dashboards [3, 12].
3. **Stochastic analysis** incorporating variability in output rates, downtime, and changeovers to provide confidence bands on LBE/ITE rather than point values [9, 13].

APPENDIX

This section provides supplementary computational details that support the main results of the paper. The appendices include the input parameters, computational procedures, and output metrics of the Python-based programs used for line performance evaluation. These details are provided to ensure transparency, reproducibility, and clarity of the proposed idle-time and classical time-based evaluation approaches without interrupting the flow of the main text.

Appendix A: Idle-Time Method – Station-Level Computational Program, Program Input, Output

A) Program Input

```
# Idle Time Method – Before & After Comparison (Python Program + Output)
# Program 3

# ----------- Common Inputs -----------
available_time_sec = 75600
customer_demand = 7500

# ----------- BEFORE CI -----------
stations_before = 20
total_idle_before = 1800

takt_before = available_time_sec / customer_demand
ite_before = (total_idle_before / available_time_sec) * 100
lbe_before = 100 - ite_before
oue_before = lbe_before

# ----------- AFTER CI -----------
stations_after = 19
total_idle_after = 190

takt_after = available_time_sec / customer_demand
ite_after = (total_idle_after / available_time_sec) * 100
lbe_after = 100 - ite_after
oue_after = lbe_after

# ----------- RESULTS -----------
print("\n-------- Idle Time Method Comparison --------")

print("\n--- BEFORE CI ---")
print("Stations:", stations_before)
print("Total Idle Time (sec):", total_idle_before)
print("Takt Time (sec/unit):", round(takt_before, 2))
print("ITE (%):", round(ite_before, 4))
print("LBE (%):", round(lbe_before, 4))
print("OUE (%):", round(oue_before, 4))

print("\n--- AFTER CI ---")
print("Stations:", stations_after)
print("Total Idle Time (sec):", total_ide_after)
print("Takt Time (sec/unit):", round(takt_after, 2))
print("ITE (%):", round(ite_after, 4))
print("LBE (%):", round(lbe_after, 4))
print("OUE (%):", round(oue_after, 4))

print("\n--- IMPROVEMENT ---")
print("Idle Time Reduction (sec):", total_idle_before - total_idle_after)
print("ITE Reduction (%):", round(ite_before - ite_after, 4))
print("LBE Improvement (%):", round(lbe_after - lbe_before, 4))

# Comments based on LBE and ITE after CI
if lbe_after >= 99:
    print("Comment: Excellent line balance after CI.")
elif lbe_after >= 95:
    print("Comment: Good line balance after CI, but further CI improvement is
possible.")
```

```
else:
    print("Comment: Poor line balance after CI. Bottlenecks or excessive idle
losses exist.")

if ite_after <= 0.5:
    print("Idle Loss: Minimal idle losses after CI.")
elif ite_after <= 2:
    print("Idle Loss: Moderate idle losses after CI.")
else:
    print("Idle Loss: High idle losses — immediate corrective action required.")
```

B) Program Output

```
-------- Idle Time Method Comparison --------

--- BEFORE CI ---
Stations: 20
Total Idle Time (sec): 1800
Takt Time (sec/unit): 10.08
ITE (%): 2.381
LBE (%): 97.619
OUE (%): 97.619

--- AFTER CI ---
Stations: 19
Total Idle Time (sec): 190
Takt Time (sec/unit): 10.08
ITE (%): 0.2513
LBE (%): 99.7487
OUE (%): 99.7487

--- IMPROVEMENT ---
Idle Time Reduction (sec): 1610
ITE Reduction (%): 2.1296
LBE Improvement (%): 2.1296
Comment: Excellent line balance after CI.
Idle Loss: Minimal idle losses after CI.
```

Appendix B: Idle-Time Method- Classical Approach – Cell-Level Computational Program, Program Input, Output

A) Program Input

```
"""
IDLE TIME METHOD (Classical) - LBE, ITE, OUE + OEE
==================================================
Fixed inputs:
- Stations (N) = 6 for ALL cases
- Available time/day = 75600 sec
- Demand/day = 7500 units
- BEFORE Lean: total idle = 1800 sec
- AFTER Lean (WITHOUT OT): total idle = 60 sec
- AFTER Lean (WITH OT): total idle = 60 sec
- Rejection assumptions: 0% and 5% (affects OEE only)
"""

available_time_sec = 75600
demand_per_day = 7500
stations = 6

cases = [
```

```
    ("BEFORE LEAN", 1800),
    ("AFTER LEAN (WITHOUT OT)", 60),
    ("AFTER LEAN (WITH OT)", 60),
]


def comment_lbe(lbe: float) -> str:
    if lbe >= 99:
        return "Comment: Excellent line balance."
    if lbe >= 95:
        return "Comment: Good line balance, but further CI improvement is
possible."
    return "Comment: Poor line balance. Bottlenecks or excessive idle losses
exist."


def comment_ite(ite: float) -> str:
    if ite <= 0.5:
        return "Idle Loss: Minimal idle losses."
    if ite <= 2:
        return "Idle Loss: Moderate idle losses."
    return "Idle Loss: High idle losses - immediate corrective action required."


takt = available_time_sec / demand_per_day

print("========== IDLE TIME METHOD REPORT (N = 6 Stations) ==========")
print("Available Time/day (sec):", available_time_sec)
print("Demand/day (units):", demand_per_day)
print("Takt Time (sec/unit):", round(takt, 2))

for title, total_idle_time_sec in cases:
    idle_per_station = total_idle_time_sec / stations
    ite = (total_idle_time_sec / available_time_sec) * 100
    lbe = 100 - ite
    oue = lbe  # classical simplified assumption (OUE = LBE)

    print("-------------------------------------------------------------")
    print(title)
    print("Stations:", stations)
    print("Total Idle Time (sec):", total_idle_time_sec)
    print("Idle Time per Station (sec):", round(idle_per_station, 2))
    print("ITE (%):", round(ite, 4))
    print("LBE (%):", round(lbe, 4))
    print("OUE (%):", round(oue, 4))
    print(comment_lbe(lbe))
    print(comment_ite(ite))

    for rejection_rate in (0.0, 0.05):
        quality = 1 - rejection_rate
        oee = oue * quality
        print(f"OEE (%) assuming {int(rejection_rate * 100)}% rejection:",
round(oee, 4))

print("-------------------------------------------------------------")
print("NOTE:")
print("1) Rejection rate does NOT change LBE/ITE/OUE (these are line balance /
idle metrics).")
print("2) Rejection rate affects only OEE through the Quality factor.")
```

B) Program Output

```
========== IDLE TIME METHOD REPORT (N = 6 Stations) ========== Available Time/day
(sec): 75600
Demand/day (units): 7500 Takt Time (sec/unit): 10.08
-----------------------------------------------------------------------
BEFORE LEAN
Stations: 6
Total Idle Time (sec): 1800
Idle Time per Station (sec): 300.0
ITE (%): 2.381
LBE (%): 97.619
OUE (%): 97.619
Comment: Good line balance, but further CI improvement is possible.
Idle Loss: High idle losses — immediate corrective action required.
OEE (%) assuming 0% rejection: 97.619
OEE (%) assuming 5% rejection: 92.7381
-----------------------------------------------------------------------
AFTER LEAN (WITHOUT OT)
Stations: 6
Total Idle Time (sec): 60
Idle Time per Station (sec): 10.0
ITE (%): 0.0794
LBE (%): 99.9206
OUE (%): 99.9206
Comment: Excellent line balance.
Idle Loss: Minimal idle losses.
OEE (%) assuming 0% rejection: 99.9206 OEE (%) assuming 5% rejection: 94.9246
-----------------------------------------------------------------------
AFTER LEAN (WITH OT)
Stations: 6
Total Idle Time (sec): 60
Idle Time per Station (sec): 10.0
ITE (%): 0.0794
LBE (%): 99.9206
OUE (%): 99.9206
Comment: Excellent line balance.
Idle Loss: Minimal idle losses.
OEE (%) assuming 0% rejection: 99.9206
OEE (%) assuming 5% rejection: 94.9246
-----------------------------------------------------------------
NOTE:
1) Rejection rate does NOT change LBE/ITE/OUE (these are line balance / idle
metrics).
2) Rejection rate affects only OEE through the Quality factor.
```

Acknowledgment: The author express their sincere gratitude to Mr. Antony Raj, Head of Production, and Mrs. Naveen, Assistant Manager, Production Department, as well as Mr. Senthil Kumaran, Head of Quality Assurance, Mr. Kiran Kumar B. S., Head of SCM and Mr. Subash A., Deputy GM-HR, for their invaluable support, guidance, and encouragement throughout this research work.

Funding Statement: "**No financing / There is no fund received for this article"** or "The authors did not receive financing for the development of this research".

Data Availability: All that support the findings of this study, testing, and trials were conducted within the company's Production, Quality, and SCM departments. The company and component details have been kept confidential in accordance with the confidentiality agreement.

Conflict of interest: "The authors declare that there is **no conflict of interest".**

Ethical Declarations: This study was conducted using process and production data provided by the participating company. No experiments were performed on humans or animals, and no personal data were collected. Therefore, ethical approval and informed consent were not required for this research.

References

[1] Ohno, T., *Toyota Production System: Beyond Large-Scale Production*, Productivity Press, 1988.
[2] Rother, M. and Shook, J., *Learning to See: Value Stream Mapping*, Lean Enterprise Institute, 1999.
[3] Monostori, L., "Cyber-physical production systems: Roots, expectations and R&D challenges," *Procedia CIRP*, 2014.
[4] Barnes, R. M., *Motion and Time Study*, Wiley, 1980/1997.
[5] Zandin, K. B. (Ed.), *Maynard's Industrial Engineering Handbook*, McGraw-Hill, 2001.
[6] Niebel, B. and Freivalds, A., *Methods, Standards, and Work Design*, McGraw-Hill, 2003.
[7] Salveson, M. E., "The assembly line balancing problem," *Journal of Industrial Engineering*, 1955.
[8] Goldratt, E. M. and Cox, J., *The Goal*, North River Press, 1984.
[9] Hopp, W. J. and Spearman, M. L., *Factory Physics*, McGraw-Hill, 2008.
[10] Becker, C. and Scholl, A., "A survey on problems and methods in generalized assembly line balancing," *European Journal of Operational Research*, 2006.
[11] Nakajima, S., *Introduction to TPM: Total Productive Maintenance*, Productivity Press, 1988.
[12] Muchiri, P. and Pintelon, L., "Performance measurement using overall equipment effectiveness (OEE): literature review and practical application discussion," *International Journal of Production Research*, 2008.
[13] Askin, R. G. and Standridge, C. R., *Modeling and Analysis of Manufacturing Systems*, Wiley, 1993.