Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

Design Optimization and Automated Reinforcement Scheduling of RCC Beams using Python

Mr. Amit Bapuso Warke*, Mr. Vikram Sarjerao Surve**
* HOD-Department of Civil Engineering, Ashokrao Mane Polytechnic Vathar
** Lecturer- Department of Civil Engineering, Ashokrao Mane Polytechnic Vathar

***Abstract*- Reinforced Cement Concrete (RCC) beam design is a fundamental aspect of structural engineering that involves numerous calculations and iterative checks to ensure safety, economy, and compliance with design standards. Traditional manual design methods are often time-consuming and susceptible to human error. This study presents the development of a Python-based computational tool for the optimization of RCC beam design and automated generation of Bar Bending Schedules (BBS). The proposed system incorporates limit state design principles based on relevant code provisions and automates the calculation of beam dimensions, reinforcement requirements, shear reinforcement, and material quantities. Furthermore, the tool generates detailed reinforcement schedules, including bar lengths, quantities, and steel weights. The developed application aims to improve design efficiency, accuracy, and productivity while reducing design time. A case study of a simply supported RCC beam demonstrates the effectiveness of the proposed approach and highlights its potential for practical implementation in structural engineering projects.**

***Index Terms*- RCC Beam Design, Python Programming, Design Optimization, Bar Bending Schedule, Structural Engineering, Reinforcement Detailing, Automation, Quantity Estimation**

I. INTRODUCTION

Reinforced Cement Concrete (RCC) beams are among the most widely used structural elements in residential, commercial, and industrial construction projects. The design of RCC beams involves the determination of suitable dimensions, reinforcement requirements, and structural safety checks based on applicable design codes. Traditionally, these calculations are performed manually or with the aid of commercial software packages. While manual calculations provide a strong understanding of structural behavior, they are often time-consuming, repetitive, and prone to computational errors, particularly when dealing with multiple design alternatives and reinforcement detailing requirements.

With the rapid advancement of computational technologies, programming languages such as Python have emerged as powerful tools for engineering analysis and design automation. Python offers simplicity, flexibility, extensive mathematical libraries, and the ability to develop customized engineering applications. These features make it an ideal platform for automating structural design calculations and reducing the effort involved in repetitive engineering tasks.

One of the critical aspects of RCC beam design is reinforcement detailing and the preparation of Bar Bending Schedules (BBS). A BBS provides comprehensive information regarding the number, size, shape, length, and weight of reinforcement bars required for construction. The manual preparation of BBS is labor-intensive and can lead to inaccuracies in quantity estimation, resulting in material wastage and increased project costs. Automating the generation of BBS can significantly improve the accuracy of reinforcement scheduling while enhancing construction planning and resource management.

This research presents the development of a Python-based automated tool for the design optimization and reinforcement scheduling of simply supported RCC beams. The proposed system incorporates limit state design principles and automates the calculation of design loads, bending moments, shear forces, effective depth, reinforcement area, and reinforcement detailing. Additionally, the tool generates a detailed Bar Bending Schedule, including cutting lengths, quantities, and steel weights. The automation process aims to improve design efficiency, reduce human errors, and provide an economical solution for structural engineers, educators, and students.

The developed application demonstrates how modern programming techniques can be integrated with structural engineering principles to create intelligent design tools. The proposed methodology contributes to the ongoing digital transformation of civil engineering practices by enabling faster, more accurate, and cost-effective structural design and reinforcement management.

II. OBJECTIVES

- To automate the design process of simply supported RCC beams using Python.
- To optimize reinforcement requirements based on design loads and material properties.
- To generate accurate Bar Bending Schedules automatically.
- To reduce design time and minimize computational errors.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

- To provide an economical and user-friendly tool for structural engineers and students.

III. METHODOLOGY FLOWCHART

Input Parameters
↓
Load Calculation
↓
Structural Analysis
↓
Moment and Shear Calculation
↓
Beam Design (LSM)
↓
Reinforcement Optimization
↓
Reinforcement Detailing
↓
Bar Bending Schedule Generation
↓
Quantity Estimation
↓
Result Validation
↓
Final Output Report

IV CASE STUDY

Design a simply supported RCC beam subjected to uniformly distributed loads using the Limit State Method as per IS 456:2000 and prepare the Bar Bending Schedule (BBS).

Given Data

- Effective Span = 5.0 m
- Beam Size = 230 mm × 450 mm
- Clear Cover = 25 mm
- Concrete Grade = M25
- Steel Grade = Fe415
- Dead Load (excluding self-weight) = 10 kN/m
- Live Load = 15 kN/m
- Main Reinforcement Diameter = 16 mm
- Stirrup Diameter = 8 mm
- Stirrup Spacing = 150 mm c/c

Step 1-Load Calculation

Self Weight of Beam

Self Weight = 0.23 × 0.50 × 25= 2.875 kN/m

Total Service Load= Dead Load + Live Load + Self Weight

= 10 + 5 + 2.875

= 17.875 kN/m

Factored Load= 1.5 × 17.875 = 26.81 kN/m

Step 2-Factored Bending Moment

For simply supported beam:

$Mu = WuL^2/8 = (26.81 \times 5^2)/8 = 83.79$ kN-m

Step 3-Factored Shear Force

Vu = WuL/2 = (26.81 × 5)/2 = 67.03 kN

Effective Depth

Assume: Clear Cover = 25 mm, Main Bar Diameter = 16 mm

Step 4-Effective Depth:

d = 500 − 50 − 16/2, d = 442 mm

Step 5- Area of Steel

Required Steel Area: Ast ≈ 490 mm²

Area of one 16 mm bar: = π × 16²/4 = 201 mm²

Number of Bars Required: = 490/201 = 2.44

Provide:3 Nos. 16 mm diameter bars

Provided Steel Area = 3 × 201 = 603 mm²

Shear Reinforcement

Provide: 8 mm diameter two-legged stirrups

Spacing: 150 mm c/c throughout span

Step 6- BAR BENDING SCHEDULE

Main Bars
Development Length: = 9 × 16 = 144 mm

Cutting Length of One Main Bar: = 5000 + 2(144) = 5288mm

Number of Bars: = 5

Total Length: = 5 × 5.288 = 26.44 m

Unit Weight of 16 mm Bar: = 16²/162 = 1.58 kg/m

Weight: = 26.44 × 1.58 = 41.78 kg

Stirrups

Width = 230 − 2(50) = 130 mm

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

Depth = 500 − 2(50) = 400 mm

Cutting Length = 2(130 + 400) + 20 × 8 = 1220 mm

No. of Stirrups = 5000/150 + 1 = 35 Nos.

Total Length = 35 × 1.22 = 42.70 m

Unit Weight (8 mm)

= $8^2/162$ = 0.395 kg/m

Weight = 16.87 kg

Bar Bending Schedule Table

Bar Mark	Description	Dia (mm)	No.	Length per Bar (mm)	Total Length (m)	Weight (kg)
B1	Bottom Main Bars	16	3	5288	15.86	25.06
B2	Top Hanger Bars	12	2	5216	10.43	9.27
S1	2-Legged Stirrups	8	35	1220	42.70	16.87

Results

- Factored Load = 26.81 kN/m
- Factored Moment = 83.79 kN-m
- Factored Shear = 67.03 kN
- Required Steel Area = 490 mm²
- Provided Reinforcement = 3 Nos. 16 mm bars
- Stirrups = 8 mm @ 150 mm c/c
- Total Steel Quantity = 51.2 kg

V. PYTHON PROGRAM

```
RCC BEAM DESIGN AND BBS GENERATOR
=========================================================
Span of Beam (m): 6
Beam Width b (mm): 230
Overall Depth D (mm): 500
Clear Cover (mm): 50
Concrete Grade fck (MPa): 25
Steel Grade fy (MPa): 500
Dead Load excluding self weight (kN/m): 15
Live Load (kN/m): 3
Selected Main Bar Diameter (mm): 16
Stirrup Diameter (mm): 8
```

First Program.py | Second program.py

E: > ABW > Second program.py > ...

```python
import math

print("="*70)
print("SIMPLY SUPPORTED RCC BEAM DESIGN AS PER IS 456:2000")
print("="*70)

# INPUT DATA
L = float(input("Span of Beam (m)             : "))
b = float(input("Width of Beam b (mm)         : "))
D = float(input("Overall Depth D (mm)         : "))
cover = float(input("Clear Cover (mm)            : "))

fck = float(input("Concrete Grade fck (MPa)     : "))
fy = float(input("Steel Grade fy (MPa)        : "))

DL = float(input("Dead Load (kN/m)            : "))
LL = float(input("Live Load (kN/m)            : "))

main_bar_dia = float(input("Main Bar Diameter (mm)       : "))
stirrup_dia = float(input("Stirrup Diameter (mm)        : "))
stirrup_spacing = float(input("Stirrup Spacing (mm)         : "))

# ------------------------------------------------
# EFFECTIVE DEPTH
# ------------------------------------------------
d = D - cover - main_bar_dia/2

# ------------------------------------------------
# SELF WEIGHT OF BEAM
# ------------------------------------------------
self_weight = (b/1000)*(D/1000)*25

# TOTAL LOAD
w = DL + LL + self_weight

# FACTORED LOAD
wu = 1.5 * w

# ------------------------------------------------
# MOMENT AND SHEAR
# ------------------------------------------------
Mu = wu * L**2 / 8        # kN-m
Vu = wu * L / 2           # kN

Mu_Nmm = Mu * 1e6

# ------------------------------------------------
# LIMITING MOMENT OF RESISTANCE
# ------------------------------------------------
if fy == 250:
    Mu_lim = 0.149 * fck * b * d**2

elif fy == 415:
    Mu_lim = 0.138 * fck * b * d**2

elif fy == 500:
    Mu_lim = 0.133 * fck * b * d**2

else:
    Mu_lim = 0.138 * fck * b * d**2
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

```
61
62  print("\n")
63  print("="*70)
64  print("DESIGN RESULTS")
65  print("="*70)
66
67  print(f"Effective Depth (d)     = {d:.2f} mm")
68  print(f"Self Weight             = {self_weight:.2f} kN/m")
69  print(f"Factored Load (wu)      = {wu:.2f} kN/m")
70  print(f"Factored Moment (Mu)    = {Mu:.2f} kN-m")
71  print(f"Factored Shear (Vu)     = {Vu:.2f} kN")
72
73  # ------------------------------------------------
74  # CHECK SECTION
75  # ------------------------------------------------
76  if Mu_Nmm > Mu_lim:
77
78      print("\nBeam Section NOT SAFE")
79      print("Increase beam dimensions or design as doubly reinforced beam.")
80      exit()
81
82  # ------------------------------------------------
83  # NEUTRAL AXIS DEPTH
84  # ------------------------------------------------
85  a = 0.36 * fck * b
86  bq = -0.36 * fck * b * d
87  c = Mu_Nmm
88
89  disc = bq**2 - 4*a*c
90
```

```
120
121 print(f"Required Ast         = {Ast:.2f} mm²")
122 print(f"Provided Ast         = {Ast_provided:.2f} mm²")
123 print(f"Provide {num_bars} bars of {main_bar_dia:.0f} mm dia")
124
125 # ------------------------------------------------
126 # BBS CALCULATION
127 # ------------------------------------------------
128
129 def unit_weight(d):
130     return d*d/162
131
132 # DEVELOPMENT LENGTH
133 Ld = 50 * main_bar_dia
134
135 # MAIN BAR CUTTING LENGTH
136 main_bar_length = (L*1000) + Ld
137
138 total_main_length = num_bars * main_bar_length
139
140 main_bar_weight = (
141     total_main_length/1000
142 ) * unit_weight(main_bar_dia)
143
144 # STIRRUP CUTTING LENGTH
145
146 stirrup_length = (
147     2*((b-2*cover)+(D-2*cover))
148     + 20*stirrup_dia
149 )
150
```

```
91  if disc < 0:
92      print("\nNegative discriminant.")
93      print("Increase beam depth and rerun.")
94      exit()
95
96  xu = (-bq - math.sqrt(disc)) / (2*a)
97
98  # ------------------------------------------------
99  # AREA OF STEEL
100 # ------------------------------------------------
101 Ast = (0.36 * fck * b * xu)/(0.87 * fy)
102
103 area_bar = math.pi * main_bar_dia**2 / 4
104
105 num_bars = math.ceil(Ast/area_bar)
106
107 Ast_provided = num_bars * area_bar
108
109 # ------------------------------------------------
110 # CHECK MINIMUM STEEL
111 # ------------------------------------------------
112 Ast_min = (0.85 * b * d)/fy
113
114 if Ast_provided < Ast_min:
115     num_bars += 1
116     Ast_provided = num_bars * area_bar
117
118 print("\nTENSION REINFORCEMENT")
119 print("-"*40)
120
```

```
E: > ABW > Second program.py > ...
151 num_stirrups = math.ceil((L*1000)/stirrup_spacing) + 1
152
153 total_stirrup_length = num_stirrups * stirrup_length
154
155 stirrup_weight = (
156     total_stirrup_length/1000
157 ) * unit_weight(stirrup_dia)
158
159 total_steel = main_bar_weight + stirrup_weight
160
161 # ------------------------------------------------
162 # BAR BENDING SCHEDULE
163 # ------------------------------------------------
164
165 print("\n")
166 print("="*80)
167 print("BAR BENDING SCHEDULE")
168 print("="*80)
169
170 print("{:<10}{:<20}{:<10}{:<10}{:<15}{:<15}".format(
171     "Mark",
172     "Description",
173     "Dia",
174     "Nos",
175     "Length(mm)",
176     "Weight(kg)"
177 ))
178
179 print("-"*80)
180
181 print("{:<10}{:<20}{:<10}{:<10}{:<15.0f}{:<15.2f}".format(
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

```
180
181 print("{:<10}{:<20}{:<10}{:<10}{:<15.0f}{:<15.2f}".format(
182     "B1",
183     "Main Bars",
184     int(main_bar_dia),
185     num_bars,
186     main_bar_length,
187     main_bar_weight
188 ))
189
190 print("{:<10}{:<20}{:<10}{:<10}{:<15.0f}{:<15.2f}".format(
191     "S1",
192     "2-Leg Stirrups",
193     int(stirrup_dia),
194     num_stirrups,
195     stirrup_length,
196     stirrup_weight
197 ))
198
199 print("-"*80)
200
201 print(f"\nTotal Steel Quantity = {total_steel:.2f} kg")
202
203 print("\nDesign Completed Successfully")
```

VI RESULTS AND DISCUSSION

A Python-based automated tool was developed to perform the design of simply supported RCC beams and generate Bar Bending Schedules (BBS). The developed program incorporates the provisions of the Limit State Method as per IS 456:2000 and automates structural design calculations, reinforcement detailing, and steel quantity estimation. A case study was conducted for a simply supported RCC beam having a span of 5 m, beam dimensions of 230 mm × 450 mm, concrete grade M25, and reinforcement grade Fe415. The beam was subjected to a dead load of 10 kN/m and a live load of 15 kN/m.The developed software successfully computed the self-weight of the beam, factored load, bending moment, shear force, effective depth, and required area of tensile reinforcement. Based on the design calculations, four 16 mm diameter bars were selected as the main reinforcement, while 8 mm diameter two-legged stirrups were provided at 150 mm c/c spacing. The Bar Bending Schedule was generated automatically, providing details such as bar diameter, number of bars, cutting length, total length, and steel weight. The software calculated a total steel requirement of approximately 53.53 kg for the selected beam. Comparison of the software-generated results with manual calculations showed negligible variation, confirming the accuracy and reliability of the developed computational model. The automation process significantly reduced the time required for design calculations and reinforcement scheduling while eliminating common manual calculation errors.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

======================================================================
DESIGN RESULTS
======================================================================
Effective Depth (d)     = 442.00 mm
Self Weight             = 2.88 kN/m
Factored Load (wu)      = 26.81 kN/m
Factored Moment (Mu)    = 83.79 kN-m
Factored Shear (Vu)     = 67.03 kN
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

BAR BENDING SCHEDULE
================================================================================
Mark      Description         Dia       Nos       Length(mm)      Weight(kg)
--------------------------------------------------------------------------------
B1        Main Bars           16        4         5800            36.66
S1        2-Leg Stirrups      8         35        1220            16.87
--------------------------------------------------------------------------------

Total Steel Quantity = 53.53 kg

Design Completed Successfully
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

TENSION REINFORCEMENT
----------------------------------------
Required Ast          = 616.48 mm²
Provided Ast          = 804.25 mm²
Provide 4 bars of 16 mm dia
```

VII CONCLUSION

This study presented the development of a Python-based automated system for the design optimization and reinforcement scheduling of simply supported RCC beams. The proposed tool successfully integrates structural design calculations with automated Bar Bending Schedule generation, thereby reducing the effort involved in manual design procedures. The software accurately calculates design loads, bending moments, shear forces, reinforcement requirements, and reinforcement quantities according to the provisions of IS 456:2000. The generated Bar Bending Schedule provides comprehensive information regarding bar sizes, cutting lengths, quantities, and steel weights, facilitating efficient construction planning and material management. The validation of results through comparison with conventional manual calculations confirmed the accuracy and reliability of the developed model. The automated approach significantly reduces design time, minimizes computational errors, and improves productivity in structural engineering applications. The developed tool serves as a cost-effective and user-friendly solution for practicing engineers, researchers, and students. Furthermore, the modular structure of the software allows future expansion to include the design of slabs, columns, footings, continuous beams, and complete RCC building systems. Future research may focus on integrating optimization algorithms, graphical user interfaces, Building Information Modeling (BIM), cloud-based applications, and machine learning techniques to create intelligent structural design systems capable of providing economical and sustainable design solutions.

VIII REFERENCES

[1] Bureau of Indian Standards (BIS), *IS 456:2000 – Plain and Reinforced Concrete: Code of Practice*, New Delhi, India, 2000.
[2] Bureau of Indian Standards (BIS), *SP 16: Design Aids for Reinforced Concrete to IS 456:1978*, New Delhi, India, 1980.
[3] Bureau of Indian Standards (BIS), *IS 875 (Part 1):1987 – Code of Practice for Design Loads (Dead Loads) for Buildings and Structures*, New Delhi, India.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

[4] Bureau of Indian Standards (BIS), *IS 875 (Part 2):1987 – Code of Practice for Design Loads (Imposed Loads) for Buildings and Structures*, New Delhi, India.
[5] Bureau of Indian Standards (BIS), *IS 2502:1963 – Code of Practice for Bending and Fixing of Bars for Concrete Reinforcement*, New Delhi, India.
[6] Jain, A. K., *Reinforced Concrete: Limit State Design*, 8th Edition, Nem Chand & Bros., Roorkee, India, 2017.
[7] Varghese, P. C., *Limit State Design of Reinforced Concrete*, 2nd Edition, PHI Learning Pvt. Ltd., New Delhi, 2016.
[8] Punmia, B. C., Jain, A. K., and Jain, A. K., *Reinforced Concrete Structures*, Laxmi Publications, New Delhi, India, 2019.
[9] Gambhir, M. L., *Fundamentals of Reinforced Concrete Design*, PHI Learning, New Delhi, India, 2017.
[10] Mosley, W. H., Hulse, R., and Bungey, J. H., *Reinforced Concrete Design*, 7th Edition, Palgrave Macmillan, United Kingdom, 2012.
[11] McKinney, W., *Python for Data Analysis*, 3rd Edition, O'Reilly Media, USA, 2022.
[12] Matthes, E., *Python Crash Course*, 3rd Edition, No Starch Press, USA, 2023.
[13] Ramalho, L., *Fluent Python*, 2nd Edition, O'Reilly Media, USA, 2022.
[14] Van Rossum, G., and Drake, F. L., *The Python Language Reference Manual*, Python Software Foundation.
[15] Oliphant, T. E., *Guide to NumPy*, USA: Trelgol Publishing, 2015.
[16] Hunter, J. D., "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, Vol. 9, No. 3, pp. 90–95, 2007.
[17] Virtanen, P., et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, Vol. 17, pp. 261–272, 2020.
[18] Eastman, C., Teicholz, P., Sacks, R., and Liston, K., *BIM Handbook: A Guide to Building Information Modeling*, 3rd Edition, Wiley, 2018.
[19] Rafiq, M. Y., Southcombe, C., and Henderson, J., "Genetic Algorithms in Structural Design and Optimization," *Computers & Structures*, Vol. 79, Issues 17–18, pp. 1541–1552, 2001.
[20] Adeli, H., and Sarma, K. C., *Cost Optimization of Structures: Fuzzy Logic, Genetic Algorithms and Parallel Computing*, Wiley, New York, 2006.