buildings

MDPI

Article

A Python-Based Framework for Automated Reinforced Concrete Flat Slab Design with Integrated Eurocode Compliance Checks

Anna Dorka Triber, Flórián Iker, János Szép and Dániel Gosztola *

Department of Structural and Geotechnical Engineering, Széchenyi István University, Egyetem tér 1, H-9026 Győr, Hungary; triber.anna.dorka@hallgato.sze.hu (A.D.T.); iker.florian@hallgato.sze.hu (F.I.); szepj@sze.hu (J.S.)
* Correspondence: gosztola.daniel@sze.hu

Abstract

Reinforced concrete flat slab design commonly relies on a fragmented workflow: finite element models are built through graphical interfaces, reinforcement demands are manually interpreted, and code-compliance checks are performed separately. This fragmentation is time-consuming, error-prone, and difficult to reproduce, particularly for slabs with non-convex geometries such as re-entrant corners and openings, where manual column classification and reinforcement zoning become especially unreliable. This study presents a Python-based parametric framework that integrates geometry definition, finite element model generation, Eurocode load-combination assembly, reinforcement design, and code-compliance verification into a single traceable Jupyter Notebook workflow. The framework connects to a commercial finite element environment through its Component Object Model Application Programming Interface (COM API). A percentile-based zoning algorithm converts continuous element-wise reinforcement demands into discrete, constructable layouts, and a geometry-aware classification procedure extends automated punching-shear verification to non-convex slab geometries. The framework is validated against a published equivalent-frame reference solution and three additional benchmark problems. Bottom reinforcement agrees within 10%, and punching-shear resistance matches exactly ($v_{Rd,c}$ = 0.399 MPa). The automated workflow reduces modelling and design time from 30–90 min to approximately 1–2 min. The principal scientific contributions are (1) a reproducible methodology for transforming continuous FEM demand fields into discrete, constructable reinforcement layouts, and (2) a geometry-aware column classification that enables automated Eurocode punching-shear checks for non-convex flat slabs.

Keywords: reinforced concrete flat slab; parametric design; finite element automation; Component Object Model Application Programming Interface (COM API); Eurocode 2; punching shear; Python; Jupyter Notebook

Check for updates

Academic Editor: Muxuan Tao

Received: 4 September 2026
Revised: 22 September 2026
Accepted: 23 September 2026
Published: 25 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Reinforced concrete flat slabs, two-way floor systems supported directly on columns without beams, rank among the most widely used floor types in contemporary construction. Their economic advantages, including reduced story height, simpler formwork, and unobstructed routing of building services below the soffit, make them popular in office buildings, car parks, residential buildings, and industrial facilities [1,2]. Their structural behaviour is, however, complex: flexural reinforcement must be designed for positive and negative moments in two orthogonal directions; punching shear at column–slab connections represents a critical and brittle failure mode that can occur with little warning, as

demonstrated experimentally by Moe [3] for isolated slab–column specimens and, more recently, by Sharaky et al. [4] for slabs with beams framing into one or both sides of the column; and the rules governing edge reinforcement and anchorage are frequently not applied with sufficient rigour in design practice [5].

Several interrelated factors influence the structural behaviour of flat slabs. The distribution of moments, which concentrates differently in column strips and middle strips, depends strongly on the slab stiffness and support configuration. The theoretical basis for punching shear verification is the Critical Shear Crack Theory (CSCT) [6], which underpins the punching provisions of both EN 1992-1-1 [7] and EN 1992-1-1:2023 [8]. Internal openings, such as stairwells and lift shafts, not only modify the load distribution but also affect the punching control perimeter at nearby columns; de Paula Randi et al. [9] showed experimentally that asymmetric openings near columns can reduce punching capacity by amounts not captured by the standard EC2 perimeter-reduction rule. Punching shear at re-entrant corner columns exhibits particularly complex behaviour: Siqueira et al. [10] tested flat slabs supported on re-entrant corner columns and reported failure loads and crack patterns that diverge from the idealised corner-column model assumed by EC2, while Ebeido et al. [11] demonstrated that unbalanced moment transfer at edge columns further amplifies this sensitivity. These findings motivate the geometry-aware column classification implemented in Section 2.7.1, since a design tool that misclassifies such columns as interior would non-conservatively underestimate the design shear stress. Edge details, namely edge reinforcement and anchorage length, are likewise mandatory code requirements [7] but are frequently omitted in practice. Taken together, a comprehensive flat slab design is a multi-step process whose manual execution is time-consuming and prone to error.

Current engineering practice typically proceeds in two sequential stages. First, the engineer constructs a model in a commercial finite element program, defines the load cases, and runs the structural analysis. The post-processor then determines the required reinforcement areas, which the engineer manually converts into a practical steel arrangement and separately verifies against the code requirements. This largely manual workflow carries well-known drawbacks [12–15]: it is time-consuming, difficult to reproduce, susceptible to operator errors, and particularly inconvenient for parametric studies in which slab geometry or loading must be examined in several variants. Manual modelling can introduce errors such as incorrect combination factors, missing load cases, or incorrectly applied boundary conditions. These often remain hidden until the analysis is complete and require considerable rework to identify.

The automation of repetitive structural design tasks through programming interfaces has attracted growing research attention over the past decade. The EC2-based domestic practice for reinforced concrete design in Hungary was summarised by Deák et al. [16]. Most publications deal with steel-framed structures—e.g., harmony-search-based optimisation of semi-rigid steel frames by Degertekin et al. [17,18] and genetic-algorithm-based cost optimisation of steel I-girders by Martins et al. [19]—bridge structures (parametric BIM-driven bridge design by Girardet and Boton [20]), or shear wall systems (GAN-based automated design by Liao et al. [21]), while work on the automated design of reinforced concrete flat slabs remains relatively scarce. Existing solutions are typically limited to beam or column sizing [22–24] or automate BIM-based pre-processing steps such as geometry import or load case generation, without linking these to a running finite element model and subsequent code checks within a single reproducible workflow. Python–SAP2000 COM API-based automated model-updating procedures have been demonstrated for structural health monitoring [25], as have IfcOpenShell-based BIM solutions for pre-processing reinforced concrete slabs [13,26]; however, to the authors' knowledge, no notebook-based workflow spanning the full chain from finite element model creation through Eurocode

load combination generation, reinforcement envelope determination, and code checks has previously been published. In the Hungarian context, Roszevák and Haris [27] conducted FEM-based investigations of cast-in-situ reinforced concrete flat slabs using an equivalent frame model. However, that approach does not extend to automated sizing and code-checking workflows. Automated slab design approaches with objectives similar to the present framework have been described in other contexts by Farid et al. [28], and the scope for parametric optimisation of flat slabs has been studied in detail by Jayasinghe et al. [29].

Recent advances in Building Information Modelling (BIM) and scripting have further expanded the scope of automated slab design. Xu et al. [30] proposed a BIM-based algorithm for automatic slab splitting and reinforcement detailing, while Izhar et al. [31] developed StrucPy.RCFA, an open-source Python package for the analysis and design of reinforced concrete frames. In parallel, machine learning (ML) methods have been increasingly applied to punching shear prediction, which represents a critical verification step in flat slab design. Shen et al. [32] and Alkhatib and Deifalla [33] employed ML techniques to predict the punching shear capacity of FRP-reinforced slabs, whereas Nasab et al. [34] used artificial neural networks and Abood et al. [35] applied gradient-boosted regression trees to conventional RC slabs. Abdallah et al. [36] compared M5P model trees with nonlinear regression approaches, and Derogar et al. [37] benchmarked several ML and deep learning algorithms against Eurocode 2 provisions. Wu and Zhou [38] combined ensemble ML models with SHAP-based interpretability analysis to identify the governing parameters of punching failure. Beyond analysis, Wefki et al. [39] demonstrated a generative design optimisation workflow that integrates structural analysis with multi-objective optimisation for reinforced concrete elements. Despite these advances, to the authors' knowledge, no publicly available tool has been identified that provides a unified, script-based workflow covering geometry definition, finite-element model generation, load-combination generation, code-compliant design checks, and reinforcement detailing for flat slabs within a single reproducible environment. In particular, the automated transformation of element-wise finite-element reinforcement demands into discrete, constructable reinforcement zones, together with geometry-aware punching shear classification for non-convex slab geometries, remains insufficiently addressed in the existing literature.

The Component Object Model (COM) technology developed by Microsoft allows applications written in different programming languages to communicate through a unified binary interface. Many commercial finite element programs, including AxisVM [40], expose their full modelling workflow through a COM API. Python's comtypes library provides a connection to these interfaces, enabling the structural model to be created, analysed, and post-processed entirely from Python without using the graphical user interface. COM API-based automation is not unique to AxisVM: analogous approaches have been applied to other commercial software such as SAP2000 and ETABS [25], indicating that the methodological framework presented here is applicable more broadly.

The Jupyter Notebook environment has become a widely adopted tool for reproducible scientific and engineering computation over the past decade [41,42]. Its interactive, documented, and version-controllable format allows every step of the design process, from geometry input to result visualisation, to be captured in a single transparent document. This property is particularly valuable in engineering applications, where the auditability and reproducibility of calculations are fundamental requirements.

The present study aims to fill this gap by presenting a parametric reinforced concrete flat slab design framework implemented in a Jupyter Notebook environment. The specific objectives of this work are: (i) to eliminate manual, GUI-based model construction for flat slabs of arbitrary (including non-convex) geometry; (ii) to automate Eurocode-compliant load combination generation and envelope-based reinforcement sizing without sacrificing

traceability; and (iii) to integrate the principal EN 1992-1-1 serviceability and punching-shear checks into the same reproducible workflow. The development rests on several key design decisions. The geometry is not imported from a BIM model but defined on an interactive tkinter drawing canvas, so the framework can be used standalone without additional software licences. Load groups and Eurocode combination factors need be entered only once; the program propagates them automatically to all load combinations, reducing the volume of user input and the risk of transcription errors. The required reinforcement areas are derived from element-wise envelope values over all ULS load combinations, ensuring that no critical slab region is overlooked. Punching shear, edge reinforcement, deflection (EN 1992-1-1 Clause 7.4), and crack width (EN 1992-1-1 Clause 7.3) checks are integrated into the same workflow and draw directly on the computed reinforcement data, support reactions, and SLS combination results.

2. Materials and Methods

The framework was developed and validated using the following materials and tools. The commercial finite element software AxisVM 20 [40] served as the structural analysis engine, accessed via its Python-exposed COM API. The Eurocode provisions EN 1992-1-1 [7] and EN 1992-1-1:2023 [8] provided the normative basis for all reinforcement and serviceability checks; EN 1990 governed load combination generation. Validation data were drawn from a published equivalent-frame reference solution (Structville, Section 3) and from the framework's own internally consistent results on three additional benchmark geometries (Section 4). All software components (Python 3.10 comtypes, tkinter, Jupyter Notebook 7.4.5. [41,42]) are freely available or included in standard scientific Python distributions.

The presented framework is implemented as a 22-cell Jupyter Notebook [41,42] executed in strictly sequential order. The complete workflow is divided into eight logically distinct stages, each occupying one or more notebook cells.

In the first stage, an interactive tkinter drawing canvas is used to define the slab outline and any internal openings. Nodes, line elements, and surface domains are then created automatically in AxisVM through the COM API. In the third stage, loads are assigned to Eurocode load groups, after which ULS, SLS, and accidental load combinations are generated automatically. In the fourth stage, a uniform finite element mesh is generated, and material and reinforcement parameters are assigned to the surface domain. In the fifth stage, an interactive selector is used to specify line supports and column supports, each with individually defined stiffness values. A linear static analysis is then run, and the program assembles the envelope reinforcement values over all ULS combinations. In the seventh stage, punching shear, edge reinforcement, deflection, and crack width checks are executed automatically, together with the percentile-based zoning of the envelope reinforcement and the automatic selection of an economical base reinforcement–supplementary reinforcement pair. In the final eighth stage, results are presented as contour maps and schematic reinforcement plans.

All intermediate results generated during the calculation are stored in Python variables that remain available throughout the Jupyter kernel session. No intermediate files are produced, avoiding repeated read/write operations. All computations were performed on a standard engineering laptop computer (Intel Core i7, 16 GB RAM), Figure 1.

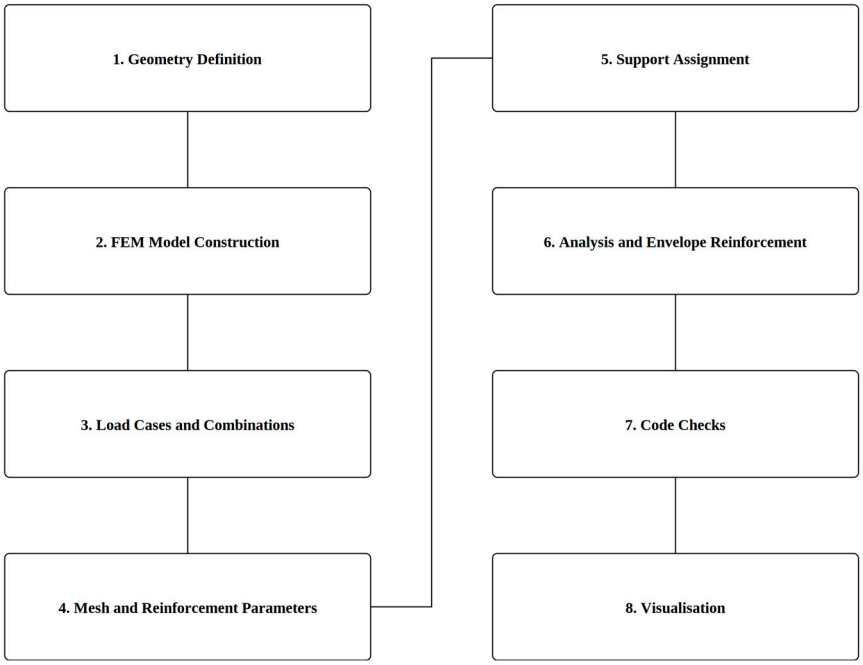


Figure 1. Eight-step data flow diagram of the parametric flat slab design process. The diagram illustrates the sequential, eight-stage architecture of the framework, from geometry definition through finite element modelling and code checks to final visualisation. Each box corresponds to one or more Jupyter Notebook cells executed in strict order.

2.1. Geometry Definition

The slab floor plan geometry is entered on an interactive drawing canvas developed specifically for this purpose using Python's tkinter library (notebook cell 2). The canvas displays a 10 cm × 10 cm raster grid rendered at 8 pixels per 10 cm. The workspace is scrollable and effectively unlimited in size. The scroll range is fixed at 5000 × 5000 pixels, accommodating large floor plans. The default window size is 900 × 700 pixels and can be extended in any direction using scroll bars.

The user places polygon vertices by mouse clicks; each vertex snaps automatically to the nearest grid point. The polygon is closed using a dedicated control button.

The application handles two polygon types: the outer contour, which describes the slab boundary, including slabs with slanted sides and irregular floor plans, and one or more internal openings marking the locations of, for example, stairwells, lift shafts, or other slab penetrations. Because the contour may be any convex or concave polygon, the tool is equally suitable for L-shaped slabs and slabs with re-entrant corners.

Both geometric elements are stored as ordered lists of (x, y) coordinate pairs in metres, with the origin at the lower-left corner of the slab.

The 10 cm snap resolution provides adequate precision for the practical size range of reinforced concrete flat slabs. If finer geometric resolution is required, a single configuration parameter suffices.

Two undo options assist in correcting user errors: deletion of the most recently placed vertex, and removal of the most recently closed polygon. This allows the geometry to be corrected without restarting the entire drawing process.

These data structures serve directly as input for the finite element model constructed in the subsequent workflow stage, Figure 2.

An important design decision in the drawing canvas was to avoid reliance on BIM formats such as IFC files. Although BIM-based geometry import represents a natural extension, its adoption would introduce additional software dependencies. It would

presuppose the availability of a detailed architectural model at the outset of the design process. The present solution is, by contrast, self-contained. The structural engineer need only open the Jupyter Notebook and can immediately begin creating a new flat slab model without any external files or additional software. This simplifies the use of the framework in practical engineering environments.

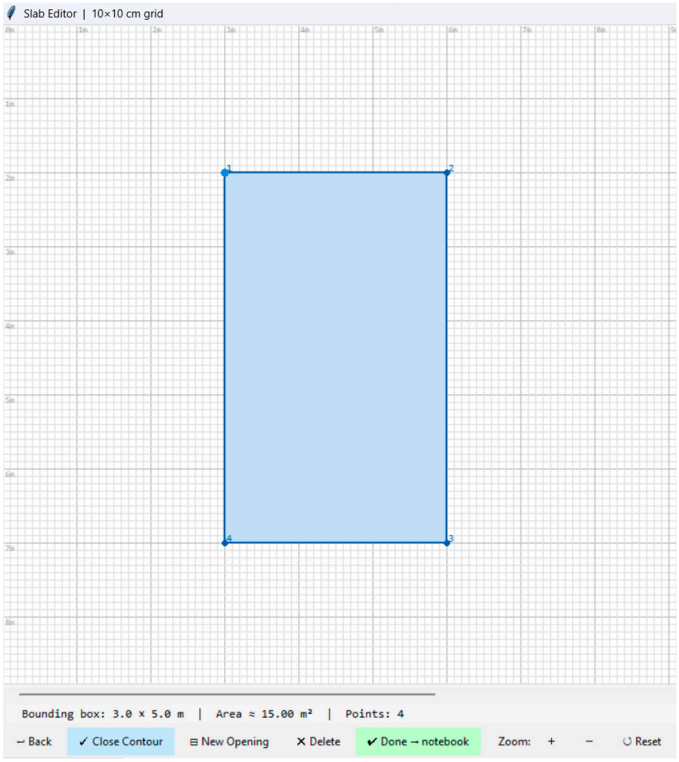


Figure 2. Interactive slab geometry editor: definition of the outer contour and an internal opening on a raster grid, illustrative example. The figure shows the tkinter-based drawing canvas used to define the slab outer contour and an internal opening by placing grid-snapped vertices. The resulting polygons are stored as coordinate lists that feed directly into the finite element model.

2.2. Finite Element Model Creation

This modelling sequence mirrors standard finite element practice—establishing a connection to the solver, defining the geometric primitives, and assembling them into a structural domain—but here every step is fully scripted rather than performed through the graphical interface, eliminating the manual, click-by-click model construction that is the principal source of transcription error in conventional workflows.

AxisVM (Inter-CAD Ltd., Budapest, Hungary) is a commercial finite element design software widely used in Central Europe [40], employed here as the computational back-end. It is referred to simply as "the design software" in what follows.

During installation, a COM server is registered that exposes the program's full modelling and analysis functionality to external applications. In the present framework, the axisvm Python package [43] provides the high-level connection interface. Python's comtypes library loads the appropriate type library at runtime and makes the classes, methods, and enumerations defined therein available as native Python objects.

Establishing the connection requires three steps: starting the design software or attaching to an already running instance, creating a new empty model, and retrieving the model object through which all subsequent operations are performed. This brief initialisation sequence is sufficient to give the program full control over the finite element model.

The same COM-based connection principle has previously been applied successfully to the automation of other commercial finite element programs [25], demonstrating that the approach presented here is not tied to a single software product.

In practice, this means that the complete model lifecycle is controlled entirely from Python, without using the graphical user interface: selecting the national standard and entering material and reinforcement parameters; creating nodes, line elements, and the slab domain; applying loads and meshing; running the linear static analysis; and reading the results.

This offers a significant practical advantage. A rectangular flat slab model with four load cases, automatic load combinations, and a finite element mesh is estimated to require approximately 30–90 min to build manually by an engineer familiar with the software's graphical interface, depending on model complexity and user experience. In comparison, the same model was generated using the presented framework in approximately 1–2 min on a standard laptop computer (Intel Core i7, 16 GB RAM). The 1–2 min value was measured for the automated workflow, whereas the 30–90 min manual modelling time is provided as an indicative estimate rather than as the result of a controlled timing study.

The use of AxisVM is not a fundamental requirement of the framework; rather, it is the chosen finite element back-end. The modelling primitives employed, including nodes, line elements, surfaces, load cases, and result arrays, have direct counterparts in other programs such as SAP2000, ETABS, or Robot Structural Analysis [25]. Porting to a different finite element program would therefore primarily require modification of the API connection layer, while the geometry handler, load combination generator, and code-checking modules would remain unchanged.

Every vertex of the outer contour is programmatically added to the finite element model, where each node receives a unique identifier. The contour is closed by connecting successive nodes with straight line elements whose identifiers are stored in a list for later use. For internal openings, the same procedure is repeated; a separate list of line elements is created for each opening and used in the following step to define the holes in the surface domain.

The flat slab surface domain is created from the line elements forming the outer contour. When creating the domain, the slab thickness, surface type, material identifier, and parameters for elastic foundation and non-linearity are specified. Internal openings are then added to the domain one by one, resulting in a surface object that represents the entire slab, including all openings, as a single structural element.

To prevent in-plane rigid-body motion of the slab, the program assigns nominal in-plane elastic supports to the surface. The applied nx and ny stiffnesses are 1000 kN/m/m, which eliminates the rigid-body motion warnings issued by the design software while having a negligible effect on the flexural behaviour that is the subject of the investigation. This simplification is an established approach for suppressing in-plane rigid-body motion; the several orders of magnitude difference between the flexural stiffness and the nominal in-plane stiffness ensures that the latter does not materially affect the results.

Taken together, these three steps replace what would otherwise be a sequence of manual mouse-driven operations in the design software's graphical interface with a deterministic, version-controllable script, so that an identical model can be regenerated at will from the same notebook cell.

2.3. Materials and Reinforcement Parameters

The concrete and reinforcing steel grades are selected from the design software's Eurocode material catalogue using standard designations such as C25/30 or B500B.

All parameters required for reinforcement design are assembled in a single data structure. This includes the top and bottom concrete cover, the axis distance of the top and bottom bars in the X and Y directions, the maximum aggregate size, the structural class, and the exposure class. The assembled parameter set is passed to the design software, and the program automatically verifies that the assignment was successful. If the assignment fails, the finite element analysis will run but all reinforcement results will be zero regardless of the applied loading. Correct assignment of the reinforcement parameters is therefore a prerequisite for the entire design workflow.

2.4. Significance of the Execution Order

An important characteristic of the framework is that a strict execution order must be maintained between setting the reinforcement parameters and invoking the structural analysis. The setting that activates the reinforcement design calculation must be applied before the linear static analysis is launched; only then may the analysis itself be run, and only after its completion may the additional parameters required for result retrieval, such as the analysis type, the load combination identifier, and the load state under investigation, be set.

It was empirically established during development that these steps cannot be reordered. If the reinforcement design setting is applied after the analysis, the design software raises no error, yet the reinforcement results take zero values for every element. This behaviour is not documented in the publicly available API documentation and was identified solely through experimentation during development. The framework therefore strictly adheres to the required execution order, ensuring the correct determination of reinforcement results.

2.5. Automatic Generation of Load Cases and Eurocode Load Combinations

Load case and combination generation is the step most prone to manual error in conventional design practice, since it requires the engineer to enumerate every permutation of permanent, variable, and accidental actions by hand; the framework instead automates this process end to end, from individual load case definition through to the final Eurocode-compliant combination set.

Following model creation, the next step is the definition of load cases and load groups. The framework is designed so that each load need be entered only once; the generation of all combinations is handled entirely by the program. Load cases are created by calling the appropriate COM API functions of the design software. For each load, the following are specified: the load case name; the load type; the associated load group; and its code category.

Surface loads, line loads, and concentrated loads are connected to the finite element model uniformly, so that subsequent load combinations automatically include all defined loads.

Properly defined load groups form the basis for generating Eurocode-compliant combinations. The framework automatically creates groups for permanent loads, imposed loads, snow loads, wind loads, and, where required, accidental actions, following the load classification of EN 1991-1-1 [44]. The Eurocode combination factors (ψ values) are assigned to each load group, so the user does not need to specify them individually for every combination.

Once the load groups have been created, the framework invokes the design software's built-in combination generation algorithm. This automatically produces the ULS persistent and transient combinations, the SLS characteristic, frequent, and quasi-permanent combinations, and the accidental load combinations. According to EN 1990 [45] Expression 6.10, the ULS design load is:

$$qEd = 1.35 \cdot Gk + 1.50 \cdot Qk \quad (1)$$

where Gk is the characteristic value of permanent loads, and Qk is the characteristic value of variable loads. This expression applies for a single variable load. When multiple variable loads are present, the ψ_0 combination factors appear for accompanying variable loads. The framework accounts for ψ_0 factors in all combinations automatically through the combination generation algorithm.

The combinations are generated in full conformity with EN 1990 [45], eliminating the need for manual assembly. For a typical flat slab problem with self-weight, superimposed dead load, imposed load, and snow load, the framework automatically produces 16 load combinations: 10 ULS, 2 SLS, and 4 accidental combinations.

In conventional design practice, the manual assembly of load combinations requires considerable effort, particularly when a structure carries several variable loads, since the number of possible combinations grows exponentially with the number of load cases. The automated approach offers several advantages. It eliminates errors arising from transcription of combination factors. It ensures consistent application of the Eurocode to all limit states, since the ψ factors and partial safety factors are entered once and applied automatically to every combination. It facilitates the straightforward addition of new load cases: if a new load must be considered, assigning it to a load group is sufficient, and the combinations are regenerated automatically. It is particularly advantageous for parametric studies, where changing the geometry or loading causes the entire combination set to be rebuilt automatically on every run, without human intervention.

As a result, the entire load combination process becomes reproducible and auditable, and the need for user intervention is substantially reduced. The generated ULS load combinations serve directly as input to the reinforcement design algorithm, where the program sweeps all ULS combinations, determines the required flexural reinforcement area for every finite element, and forms element-wise envelope values. The reinforcement design is thus based not on a single selected combination but on the full ULS combination set, ensuring that the final reinforcement plan satisfies every load state considered.

The practical benefit of this automation is cumulative: because load cases, groups, and combinations are all generated from a single set of user-defined inputs, any subsequent change to the loading-for example, adding a new variable action-propagates automatically through the entire combination set without requiring the engineer to re-derive it by hand.

2.6. Determination of Flexural Reinforcement

Converting raw finite element output into a constructable reinforcement layout involves three conceptually distinct steps-establishing the governing design envelope, partitioning it into a practical zoning scheme, and selecting an actual bar arrangement-each of which is conventionally performed manually by the engineer and is automated here in sequence.

After the linear static analysis, the required flexural reinforcement areas are read from the result handler object of the design software. The algorithm sweeps all ULS persistent and transient load combinations and, for each finite element, determines the maximum required reinforcement area across all combinations.

Mathematically this is expressed as:

$$As,bx,max(j) = max\ \{As,bx(j, k): k \in ULS\} \quad (2)$$

$$As,by,max(j) = max\ \{As,by(j, k): k \in ULS\} \quad (3)$$

where j is the element index, and k is the ULS combination index.

The program assembles four result arrays: Asbx (bottom reinforcement in the X direction), Asby (bottom reinforcement in the Y direction), Astx (top reinforcement in the

X direction), and Asty (top reinforcement in the Y direction). The design software provides these in m^2/m. The framework multiplies the values by 10^6 to convert to mm^2/m and sets all negative values to zero, as these indicate compression zones where flexural reinforcement is not required. This envelope procedure ensures that every finite element is governed by its own critical ULS combination, rather than by a single globally critical combination for the whole slab.

Practical reinforcement plans for flat slabs generally comprise two reinforcement regions: a base reinforcement covering the majority of the slab, and supplementary reinforcement zones located in column strips, at re-entrant corners, and in other high-demand areas. The framework implements this engineering practice through a percentile-based algorithm. For each of the four reinforcement result arrays, it computes the p-th percentile specified by the user; the default value is $p = 70\%$.

This becomes the base reinforcement demand for the direction in question. Every finite element whose required reinforcement exceeds this value is automatically assigned to the supplementary reinforcement zone. The use of the percentile is motivated by the distribution of FEM reinforcement results. In flat slabs, the required reinforcement area in column strips is frequently two to five times greater than in the central field region. If the base reinforcement were selected on the basis of the maximum demand over the entire slab, an unjustifiably large amount of steel would be incorporated over most of the slab area. Using the p-th percentile instead means that the base reinforcement is self-sufficient over $p\%$ of the slab area, with supplementary reinforcement added over the remaining $(100 - p)\%$. At the default $p = 70\%$, the base mesh suffices over 70% of the slab, and additional reinforcement is placed over the remaining 30%. The governing reinforcement demand in the supplementary zone is taken as the 75th percentile of values within that zone, thereby excluding extreme local peaks from the design. The use of the 75th percentile as the governing design value for the supplementary zone is a deliberate simplification analogous to standard structural design practice, where representative rather than absolute-maximum values are commonly adopted to avoid excessive local conservatism. Consequently, individual finite elements within the supplementary zone whose demand exceeds the 75th-percentile value are not explicitly guaranteed to be covered by the resulting bar arrangement, although the upward rounding to the nearest available bar diameter–spacing combination provides some margin in practice. Users requiring an absolute guarantee against local under-reinforcement may set the percentile parameter p to a higher value (up to $p = 100\%$, corresponding to the full envelope maximum), at the cost of a more conservative and less economical design. This trade-off, and its practical implications, is discussed further in Section 5.1.

It is important to emphasise that the 75th-percentile rule is a practical engineering simplification for producing constructable reinforcement layouts, not a code-prescribed design rule. The mesh-convergence study (Section 5.1) demonstrates that envelope maxima are mesh-sensitive and therefore unsuitable as direct design values, but it does not by itself validate the adequacy of any particular percentile threshold. The responsibility for verifying that the adopted percentile provides sufficient coverage for the specific design at hand remains with the engineer, who should adjust the parameter based on the geometry, loading, and criticality of the structure.

The required reinforcement cross-sections are not in themselves sufficient for producing a construction drawing; the framework therefore employs a reinforcing steel database. The database contains the following nominal bar diameters: 8, 10, 12, 14, 16, 20, 25, and 32 mm, with four standard spacings: 100, 150, 200, and 300 mm. The possible combinations cover reinforcement areas ranging from 94 to 8042 mm^2/m.

The program automatically selects the base reinforcement–supplementary reinforcement pair that satisfies the required area, minimises total steel consumption, and enforces a

minimum two-catalogue-step difference between the two bar diameters. This constraint prevents the combined use of, for example, Ø8 and Ø10 bars, which are difficult to distinguish on site. Instead, the system favours combinations such as Ø10 + Ø14 or Ø12 + Ø16, which are unambiguously identifiable during erection. When sizing the supplementary mesh, the program works not from the total required area but from the difference between the area actually provided by the base mesh (As,base,prov) and the governing demand in the supplementary zone (As,supp). The governing supplementary value is taken as the 75th percentile of values within the supplementary zone, excluding extreme local peaks. This approach results in more economical steel consumption, since the contribution of the base mesh is not disregarded.

This three-step procedure—envelope determination, percentile-based zoning, and catalogue-constrained bar selection—is what allows the framework to output a reinforcement drawing directly usable in construction documentation, rather than merely a continuous field of required steel areas that would still require manual interpretation by the engineer.

2.7. Integrated Eurocode 2 Checks

2.7.1. Punching Shear Verification (EN 1992-1-1 Clause 6.4)

For flat slabs without beams, punching shear at column–slab connections is the most critical ultimate limit state. The punching mechanism is described by the Critical Shear Crack Theory (CSCT) [6], which underpins the punching provisions of both EN 1992-1-1 [7] and EN 1992-1-1:2023 [8]. The framework implements the EN 1992-1-1 [7] verification procedure for interior, edge, and corner columns. A comparative review of punching design methods used in Hungarian practice and the EC2 provisions may be found in Jobbágy and Völgyi [46].

The effective depth is:

$$d = h - ct - (\varphi x + \varphi y)/2 \tag{4}$$

where h is the slab thickness, ct is the top cover, and φx, φy are the top bar diameters in the X and Y directions. The effective depth for punching is consistently referenced to the top reinforcement layer, which governs the hogging (negative) moment region above columns.

Flexural reinforcement ratio:

$$\rho l = \sqrt{(\rho lx \cdot \rho ly)} \leq 0.02 \tag{5}$$

where ρlx = As,tx,mid/(d · 1000) and ρly = As,ty,mid/(d · 1000). The mean reinforcement values are computed from the FEM envelope results.

Punching shear resistance without shear reinforcement [6,47]:

$$vRd,c = CRd,c \cdot k \cdot (100 \cdot \rho l \cdot fck)^{(1/3)} \tag{6}$$

where CRd,c = 0.18/γc = 0.12 and k = min(1 + $\sqrt{}$(200/d), 2.0).

The maximum punching shear resistance, which limits the slab thickness adequacy:

$$vRd,max = 0.5 \cdot \nu \cdot fcd \tag{7}$$

where ν = 0.6 · (1 − fck/250). If vEd > vRd, max, the slab thickness is insufficient and must be increased.

The framework automatically identifies the column position relative to the slab contour (interior, edge, or corner). It determines the basic control perimeter and the eccentricity factor β in accordance with EN 1992-1-1 Clause 6.4.3 [7]. The edge and corner perimeter expressions given in EN 1992-1-1 Figure 6.4.2(2) yield a reduced perimeter u_1, to which the

eccentricity factor β, per EC2 Clause 6.4.3(3), is applied as an approximate allowance for load eccentricity; the two are not double-counted. The 2023 edition of EC2 [8] revised the punching provisions based on the CSCT mechanical model [6]. Experimental support for eccentric punching and the application of the β factor is provided by Vargas et al. [48] and Abu-Salma et al. [49].

Interior column:

$$u_1 = 2(c_1 + c_2) + 4\pi d,\ \beta = 1.15 \quad (8)$$

Edge column:

$$u_1 = c_1 + 2c_2 + 2\pi d,\ \beta = 1.40 \quad (9)$$

Corner column:

$$u_1 = c_1/2 + c_2/2 + \pi d,\ \beta = 1.50 \quad (10)$$

A comparative evaluation of the EC2 and fib Model Code 2010 [50] punching provisions has been performed by M. Ricker, C. Siburg [51]. The design shear stress on the control perimeter, including the β eccentricity factor:

$$vEd = \beta \cdot VEd/(u_1 \cdot d) \quad (11)$$

where VEd is the maximum ULS support reaction at the column node (the maximum over all ULS combinations) and β is the eccentricity factor depending on column position (interior: 1.15; edge: 1.40; corner: 1.50). This is a deliberate conservative simplification: EC2 Clause 6.4.2(2) permits the deduction of the load applied within the control perimeter (ΔVEd), which the framework conservatively disregards.

If vEd > vRd,c, the required shear reinforcement is determined by the program:

$$Asw/(sr \cdot u_1) = (vEd - 0.75 \cdot vRd,c)/(1.5 \cdot fywd,ef) \quad (12)$$

where fywd,ef = min(250 + 0.25d, fywd) [MPa] is the effective yield strength per EN 1992-1-1 Expression (6.52).

The radial spacing of reinforcement rings is sr = 0.75d; the first ring is placed 0.5d from the column face. For each column, the framework determines the required number of reinforcement rings and, for bar diameters from Ø8 to Ø16, the required number of bars per ring, in tabular form.

The control perimeter outside which shear reinforcement is no longer required:

$$uout = \beta \cdot VEd/(vRd,c \cdot d) \quad (13)$$

Re-entrant corners are also subject to punching shear checks. The framework automatically identifies concave (re-entrant) vertices in the contour using a cross-product-based geometric analysis: a vertex is classified as re-entrant if the cross product of the two edges meeting at that vertex is negative (for clockwise traversal). When a re-entrant corner is detected, the conservative approximation is applied (β = 1.50, $c_1 = c_2 = 0$) and the governing shear force is estimated from the approximate ULS load on the tributary area. Since no concentrated column reaction is associated with a re-entrant corner, a more accurate check should use the FEM shear force envelopes.

All checks are performed automatically for every column support entered by the user. Column coordinates, dimensions, and constraint types are stored in a data structure. The program automatically classifies each column as interior, edge, or corner based on its minimum distance to each boundary edge segment of the slab contour polygon, rather than to the overall bounding box. A column within tolerance of a single boundary edge is classified as an edge column, while a column within tolerance of two intersecting

edges—whether at a convex (outward) or concave (re-entrant) corner—is classified as a corner column. For columns at concave (re-entrant) corners, where the slab extends on three sides, the standard corner perimeter and $\beta = 1.50$ would significantly underestimate the available control perimeter; such columns are therefore reclassified as edge columns ($\beta = 1.40$), consistent with the experimental findings of Siqueira et al. [10]. This distance-to-contour approach correctly handles non-convex slab outlines, such as the L-shaped geometry examined in Section 4.2, where a bounding-box-based classification would incorrectly identify columns located on interior free edges or at re-entrant corners as interior columns. The classification tolerance is set to 5 cm, corresponding to half the 10 cm grid resolution used for geometry and support placement (Section 2.1); this ensures that a column snapped to a contour-defining grid point is always classified correctly despite floating-point rounding, while remaining well below the spacing between distinct grid points. Where a re-entrant corner coincides with a column position, the column-based punching check (using the actual finite element reaction force) takes precedence over the geometry-only re-entrant corner estimate, avoiding duplicate or conflicting evaluations of the same location. The results for all columns and re-entrant corners are collected in a summary table that lists the element identifier, type, design shear stress (vEd), punching resistance (vRd,c), check result, required reinforcement (Asw, mm^2/ring), and number of reinforcement rings. Machine learning-based punching shear prediction models [52–54] offer a future opportunity for extending and validating the framework.

2.7.2. Edge Reinforcement and Anchorage Length (EN 1992-1-1 Clauses 9.3.1.4 and 8.4.4)

Eurocode 2 [7] requires that U-shaped edge reinforcement, or an equivalent solution, be provided along all free slab edges. The framework automatically classifies each slab edge as predominantly X- or Y-directed and performs the check for every free edge.

Design yield strength:

$$f_yd = f_yk/\gamma_s = 500/1.15 = 435 \quad (14)$$

where f_yd is the design yield strength of the reinforcing steel [MPa], f_yk = 500 MPa is the characteristic yield strength (B500B), and γ_s = 1.15 is the partial safety factor for reinforcing steel per EN 1992-1-1 [7].

The design bond stress per EN 1992-1-1 Clause 8.2(2):

$$f_bd = 2.25 \cdot \eta_1 \cdot \eta_2 \cdot f_ctd \quad (15)$$

where $\eta_1 = 1.0$ for good bond conditions per EN 1992-1-1 Figure 8.2: this applies when the bar is in the lower 250 mm of the pour or at least 300 mm from the top surface during casting. For slabs with $h \leq 250$ mm, all reinforcement is in the good bond zone ($\eta_1 = 1.0$). For thicker slabs ($h > 250$ mm), top reinforcement may fall in the poor bond zone, requiring $\eta_1 = 0.7$, $\eta_2 = 1.0$ (bar diameter ≤ 32 mm), and f_ctd = α_ct · f_ctk,0.05/γ_c = 0.7 · f_ctm/1.5 is the design tensile strength of concrete, with α_ct = 1.0.

The minimum required edge reinforcement area:

$$A_s,edge = max(0.25 \cdot \bar{A}_s,main; \rho_min \cdot 1000 \cdot d) \quad (16)$$

where Ā_s, main is the mean value of the main reinforcement perpendicular to the edge in question [mm^2/m], ρ_min = max(0.26 · f_ctm/f_yk; 0.0013) per EN 1992-1-1 [7] Clause 9.2.1.1, and d is the effective depth [mm].

The design anchorage length per EN 1992-1-1 [7] Clause 8.4.4:

$$l_bd = \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \cdot \alpha_4 \cdot \alpha_5 \cdot l_b,rqd \cdot (A_s,req/A_s,prov) \quad (17)$$

where l_b,rqd = (φ/4) · (f_yd/f_bd) is the basic anchorage length, and the factors α_1–α_5 are taken as 1.0 in this conservative approximation. The minimum condition for the design anchorage length is: l_bd ≥ max(0.3 · l_b,rqd; 10φ; 100 mm).

For each free edge, the framework determines, for bar diameters from Ø8 to Ø16, the required bar spacing, the provided reinforcement area, the basic anchorage length (l_b,rqd), and the design anchorage length (l_bd), presenting the results in tabular form. It also automatically identifies free corners where corner reinforcement is required in both directions and provides the required lap length.

2.7.3. Deflection Check (EN 1992-1-1 Clause 7.4)

EN 1992-1-1 Clause 7.4 requires that the deflection of a reinforced concrete flat slab not exceed span/250, where the span is the largest dimension between adjacent supports. The framework performs the deflection check based on the SLS quasi-permanent combination results. The governing span is taken as the largest centre-to-centre distance between adjacent supports. For point-supported slabs this is determined automatically from the column coordinates; for line-supported slabs the framework conservatively uses the largest slab dimension as the governing span.

$$w_max = L_eff/250 \tag{18}$$

where Leff is the effective span; the program reads the z-direction displacements at all nodes from the linear static analysis results under the SLS combinations and determines the maximum slab deflection. This is compared with the allowable wmax value, and the check result is reported clearly. If the deflection exceeds the allowable value, the framework displays a warning and provides an estimate of the required slab thickness. The deflection check uses the same linear elastic stiffness as the global structural analysis; unlike the crack width check, it does not apply a reduced (cracked-section) stiffness. This is a non-conservative simplification, since Eurocode 2 deflection assessment normally requires cracked-section or effective stiffness (Clause 7.4.3), and the use of uncracked stiffness may therefore underestimate deflections for heavily cracked slabs. This limitation is discussed further in Section 5.1.

Consequently, the deflection results reported in Sections 3 and 4 should be interpreted as a screening-level indicator rather than a definitive serviceability verification.

2.7.4. Crack Width Check (EN 1992-1-1 Clause 7.3)

EN 1992-1-1 Clause 7.3 limits crack widths to protect the durability and serviceability of the reinforced concrete structure. The framework applies an allowable crack width of w_k,max = 0.30 mm, corresponding to exposure classes XC2–XS3 under EN 1992-1-1 Table 7.1N (all of which share this limit); the exposure class, and the corresponding w_k,max, is a user-configurable input, and the specific value used here was selected to represent a moderately aggressive environment rather than the least severe (XC1, 0.40 mm) case. The calculation is based on the following material properties.

Mean tensile strength of concrete:

$$f_ctm = 0.30 \cdot f_ck^{(2/3)} \tag{19}$$

where f_ctm is the mean tensile strength of concrete and f_ck is the characteristic compressive cylinder strength of concrete.

Modulus of elasticity of concrete:

$$E_cm = 22 \cdot ((f_ck + 8)/10)^{0.3} \tag{20}$$

where E_cm is the secant modulus of elasticity of concrete.

Modulus of elasticity of reinforcing steel: E_S = 200,000 MPa.

Modular ratio:

$$\alpha_e = E_s/E_cm \quad (21)$$

where α_e is the modular ratio (ratio of the moduli of elasticity of reinforcing steel and concrete).

The crack width is calculated using the direct calculation method of EN 1992-1-1 Clause 7.3.4. The crack width is computed for the governing SLS quasi-permanent moment in each direction using the cracked section properties.

2.8. Interactive Support Definition

Supports are defined on a second interactive drawing canvas that uses the same coordinate system, scale, and margin settings as the geometry canvas. The previously drawn slab contour therefore appears at the same position, facilitating support placement since the user does not need to re-interpret the geometry.

The program provides two modes: edge mode and column mode. In edge mode, the user clicks on a slab edge, whereupon a dialogue box appears in which six stiffness parameters may be entered: translational stiffnesses in x, y, and z and three rotational stiffnesses. In column mode, the user places the column support by clicking; its position snaps automatically to the nearest grid point, after which a dialogue box allows the constraint type at the bottom and top node and the column cross-sectional dimensions to be specified, Figure 3.

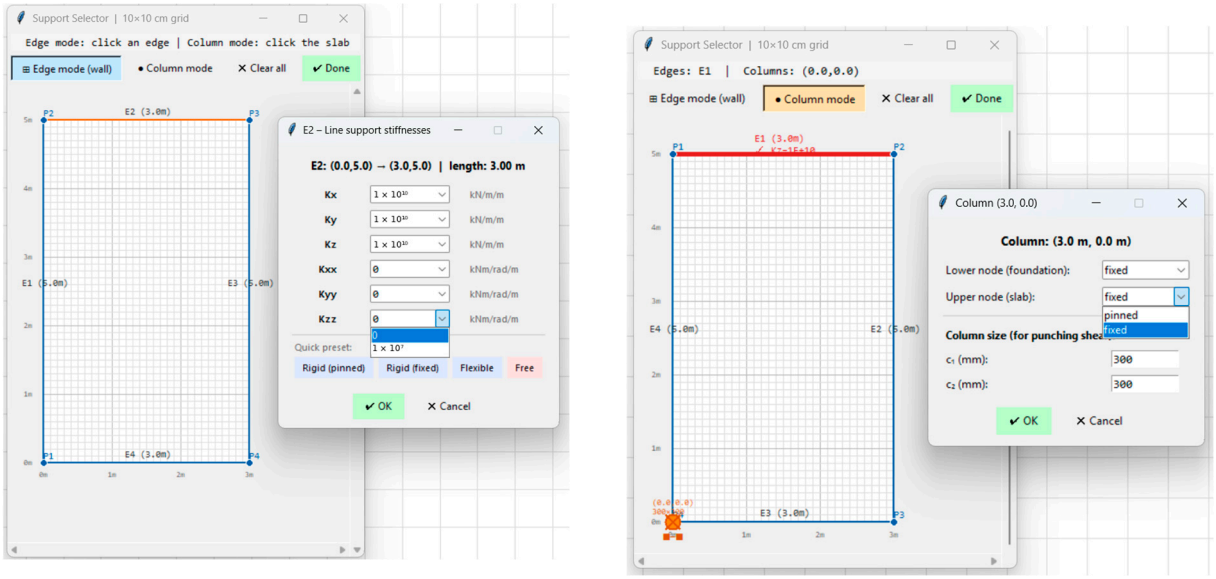


Figure 3. Interactive support selector: assignment of edge line supports and column supports, illustrative example. The two panels show the support-definition canvas: the left panel illustrates the stiffness input dialogue for a selected line (edge) support, and the right panel shows the equivalent dialogue for a column support, including its cross-sectional dimensions. Supports are placed on the same coordinate grid as the geometry, so their position matches the previously drawn slab outline exactly.

After the finite element mesh has been generated, the design software no longer guarantees that line element identifiers remain in their original order. The framework therefore does not locate the selected slab edge by index but geometrically: it examines the midpoint of the edge supports and identifies the edge selected by the user using a

pre-defined distance tolerance. This approach remains reliable even if the design software assigns internal line element identifiers in a different order.

2.9. Result Visualisation

The framework automatically produces two types of result figure. The first type presents contour maps for the four reinforcement result arrays, namely bottom and top reinforcement in the X and Y directions, obtained by projecting element centroid values onto a regular grid using linear interpolation. Each contour map includes a colour bar, a contour line marking the base reinforcement threshold, and text annotations showing the proposed bar diameters and spacings.

The second type is a schematic reinforcement plan that separates areas covered by the base reinforcement from supplementary reinforcement zones and indicates their bounding dimensions. The supplementary mesh is sized from the difference: it provides only the reinforcement demand not yet covered by the base mesh, so that the composite cross-section of the two meshes together satisfies the required reinforcement. This presentation format can be used directly as part of a design document or calculation report, Figure 4.

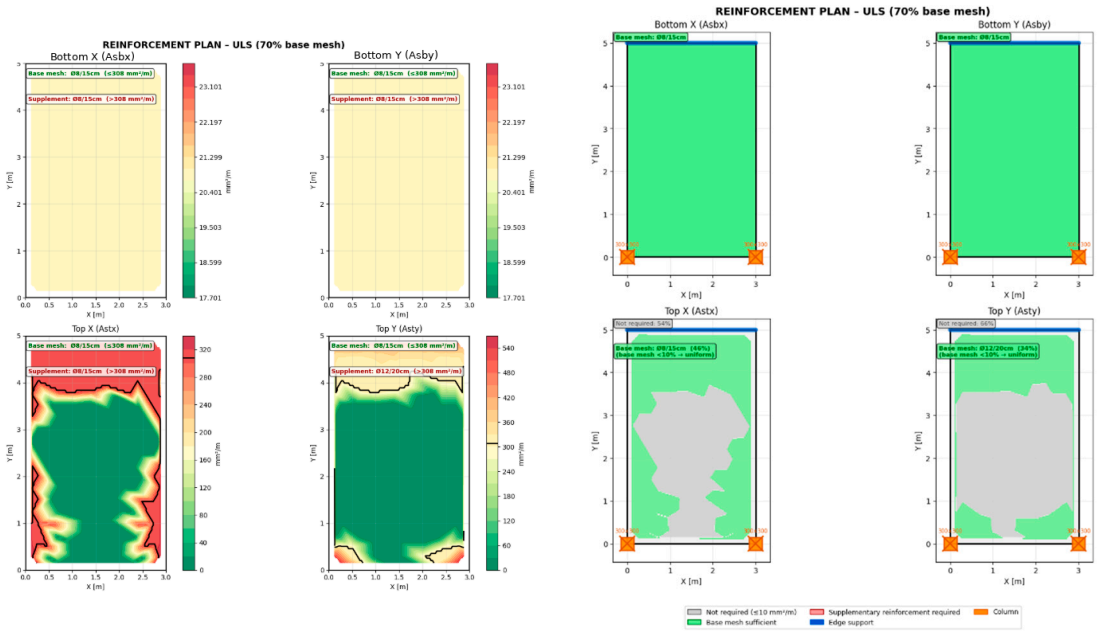


Figure 4. Framework result figures: contour map of flexural reinforcement areas (**left**) and schematic reinforcement zone plan with bounding dimensions (**right**), illustrative example. The (**left**) panel shows a contour map of the flexural reinforcement demand with a colour scale and annotated bar diameters and spacings, while the (**right**) panel shows the corresponding schematic plan separating the base mesh from the supplementary reinforcement zones. Together, the two outputs translate the raw FEM results into a directly usable reinforcement layout.

3. Validation: Structville Reference Problem

3.1. Geometry and Loading

To verify the framework results, a comparison is carried out using a published, fully documented design example. The reference problem is an EC2-compliant manual design of a reinforced concrete point-supported flat slab [55], solved by the equivalent frame method. This method models the slab as an equivalent frame of rigid beams and determines the moments in column strips and middle strips analytically. Comparing the manual calculation and the linear elastic FEM results allows verification of the framework implementation, since both methods rest on the same code provisions [7,45] but start from different modelling assumptions.

The slab examined is a regular 3 × 3 bay point-supported reinforced concrete flat slab whose floor plan is shown in Figure 5. The bay size is 6.5 m × 6.5 m; the total slab extent is 19.5 m × 19.5 m. The slab thickness is 230 mm and the column dimensions are 300 × 300 mm. The concrete grade is C30/37 and the reinforcing steel is B500B. Top and bottom cover are both 30 mm.

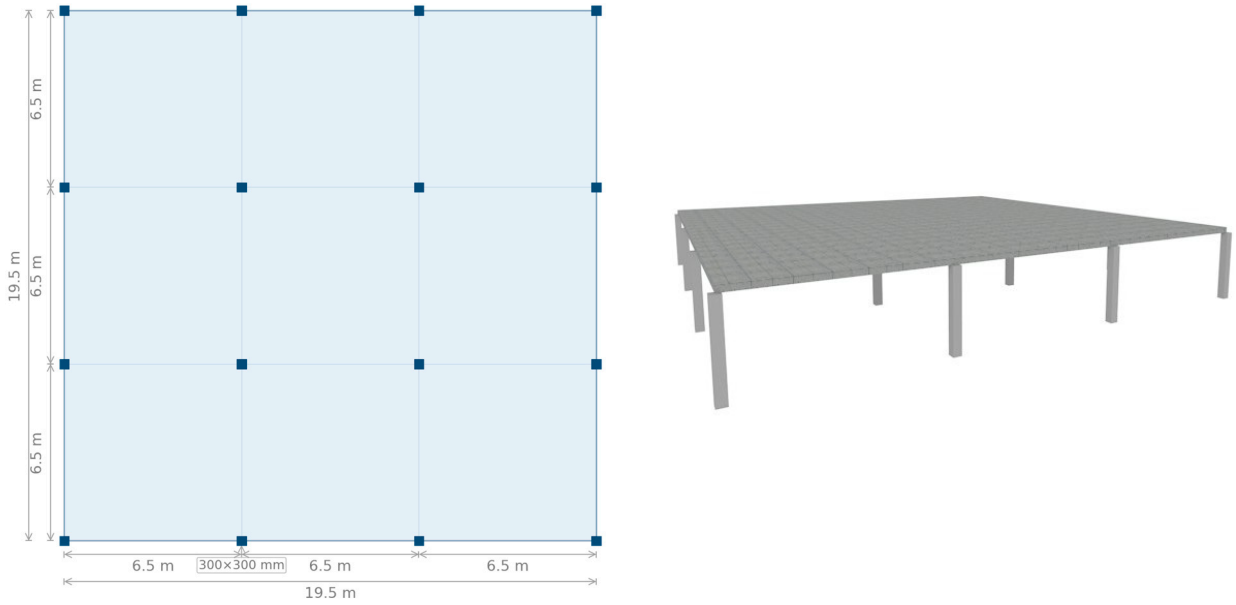


Figure 5. Floor plan of the point-supported reinforced concrete flat slab. The figure shows the 3 × 3 bay, 19.5 m × 19.5 m point-supported flat slab used as the validation benchmark, with a 6.5 m × 6.5 m regular column grid. This geometry corresponds to the published Structville reference example against which the framework's results are compared.

The applied loads are self-weight 5.75 kN/m^2 (computed automatically from the slab thickness and a concrete unit weight of 25 kN/m^3), superimposed dead load 2.40 kN/m^2, and imposed load 3.00 kN/m^2. No snow load is included (q_snow = 0 kN/m^2). The ULS design load per EN 1990 [45] Expression 6.10: qEd = 1.35 × (5.75 + 2.40) + 1.5 × 3.0 = 15.50 kN/m^2. All 16 column supports are rigidly connected to the slab; columns are located at grid points at 6.5 m × 6.5 m centres.

3.2. Framework Execution

The problem is run in the framework as follows. On the interactive drawing canvas, the slab outer contour is entered as a 19.5 m × 19.5 m rectangular polygon without internal openings. The input parameters are: concrete C30/37, reinforcing steel B500B, slab thickness h = 230 mm, top and bottom cover ct = cb = 30 mm. The finite element model is then built automatically through the AxisVM COM API: nodes, line elements, and the slab domain are created programmatically. After specifying the four load cases (self-weight, superimposed dead load, imposed load, and nominal snow with zero value) and assigning them to load groups, the program automatically generates all ULS, SLS, and accidental combinations per EN 1990 [45]. After mesh generation and reinforcement parameter assignment, the 16 column supports are entered on the interactive support selector with 300 × 300 mm cross-sections. The program automatically identifies the column positions: corner columns (β = 1.50) at the slab corners, edge columns (β = 1.40) along the edges, and interior columns (β = 1.15) at interior grid points, all per EN 1992-1-1 Clause 6.4.3 [7].

After the linear static analysis, the framework computes element-wise maximum (envelope) reinforcement areas over all ULS combinations, performs the punching shear check for all columns, and the edge reinforcement check for all free slab edges. Total

execution time from model construction to result output was approximately 1–2 min on a standard engineering laptop computer (Intel Core i7, 16 GB RAM).

3.3. Flexural Reinforcement Comparison

The linear elastic FEM and the equivalent frame method start from different modelling assumptions, so their results require interpretation. The equivalent frame method distributes moments over column and middle strips and reports their mean values. The FEM, by contrast, provides the required reinforcement area element by element; the envelope takes the worst combination for each element individually. Table 1 summarises the FEM-based envelope reinforcement from the framework and the equivalent frame reference results.

Table 1. Comparison of flexural reinforcement results.

Result	Framework (FEM, Envelope)	Reference (Equiv. Frame)	Difference
Bottom reinforcement X (Asbx)	674 mm^2/m	727 mm^2/m	−8.4%
Bottom reinforcement Y (Asby)	732 mm^2/m	727 mm^2/m	+1.9%
Top reinforcement X (Astx)	2278 mm^2/m	~1000 mm^2/m	+128%
Top reinforcement Y (Asty)	2595 mm^2/m	~1000 mm^2/m	+160%

The table compares the FEM-based envelope reinforcement from the framework with the manual equivalent-frame reference values. Bottom reinforcement agrees within 10%, while the higher FEM top-reinforcement values reflect the expected local peak behaviour of linear elastic FEM rather than an implementation error.

The bottom reinforcement values show good agreement: the difference remains within 10% in both directions, which is considered satisfactory given the modelling differences between the two methods.

For the top reinforcement, the FEM envelope value (2278–2595 mm^2/m) considerably exceeds the equivalent frame result (~1000 mm^2/m). This discrepancy is expected and methodological in nature, not indicative of an implementation error.

The equivalent frame method models the slab as an equivalent frame of rigid beams and distributes the column strip moment uniformly across the full column strip width, in this case 3.25 m (half the bay size), yielding a mean demand for the entire column strip. The linear elastic FEM, by contrast, determines the required reinforcement area element by element and identifies local stress concentrations above column heads. The envelope procedure captures the maximum of these local peaks, which concentrate within a few elements of the column face and greatly exceed the column strip mean from the equivalent frame method.

The 128–160% deviation reflects a methodological difference rather than a computational error: the equivalent frame method distributes the column strip moment as a uniform average over the full strip width, whereas the FEM envelope reports the element-wise maximum, which by construction captures the localised peak at the column face. Based on the FEM contour map, this demand is strongly concentrated: only 41% of elements carry a non-negligible top reinforcement demand, and the reported envelope maximum of 2296 mm^2/m occurs only in the small number of elements immediately adjacent to the column. The 70th-percentile value of the top reinforcement demand is 398.5 mm^2/m, which is lower than the equivalent frame column strip mean (~1000 mm^2/m). This difference is expected: the equivalent frame mean represents the average moment across the full column strip width (3.25 m), whereas the 70th percentile of element-wise FEM values excludes the high-demand concentration above the column head. A direct numerical comparison between the two values is therefore not meaningful, as they represent different statisti-

cal measures of fundamentally different spatial distributions. This spatial concentration pattern is consistent with the known local-peak behaviour of linear elastic FEM near concentrated supports [56], and explains the reported deviation without requiring a direct numerical reconciliation between two methods that are computing fundamentally different quantities (a uniform strip average versus a local element-wise maximum). Moreover, as the mesh-convergence check in Section 5.1 demonstrates, this envelope maximum is not itself a mesh-independent quantity: it reflects the theoretical moment singularity that arises at point column supports in linear elastic plate models, and grows without bound as the mesh is refined. The reported value of 2296 mm^2/m should therefore be read as a discretisation-dependent upper bound rather than a converged physical demand, further reinforcing the case for the percentile-based design value adopted in Section 2.6. over the raw envelope maximum. The two methods therefore yield results that are not directly comparable at the element level; they should be assessed against their respective design assumptions rather than reconciled numerically. This finding is consistent with the relevant literature [56], which emphasises that the linear elastic FEM envelope maximum is inherently greater than the equivalent frame column strip mean, and that this is not a shortcoming of the FEM approach but rather a more accurate description of the actual stress distribution.

3.4. Punching Shear Comparison

Table 2 summarises the principal intermediate and final results of the punching shear check and compares them with values computed directly from EN 1992-1-1 [7].

Table 2. Punching shear check comparison: interior column.

Property	Framework	EC2 Manual
Concrete grade	C30/37	C30/37
Effective depth (d)	184 mm	184 mm
Size factor (k)	2.000	2.000
β: interior column	1.15	1.15
β: edge column	1.40	1.40
β: corner column	1.50	1.50
Reinforcement ratio (ρl)	0.153%	0.153%

The table compares the key intermediate and final punching shear quantities computed by the framework with the corresponding EN 1992-1-1 manual calculation. All values, including the shear resistance vRd,c and the β eccentricity factors for all column types, agree exactly with the manual reference.

The punching shear resistance (vRd,c = 0.399 MPa) and the effective depth (d = 184 mm) agree exactly with the EN 1992-1-1 [7] manual calculation. The β eccentricity factors are correctly applied for all three column types. The automatic column classification, which assigns types based on coordinates distance-to-contour classification, correctly identified the 4 corner columns, 8 edge columns, and 4 interior columns. When computing ρl, the framework uses the mean of the raw FEM envelope reinforcement before the As,min correction (ρl = 0.153%), which agrees exactly with the Structville manual calculation value.

3.5. Edge Reinforcement Check

Table 3 compares the material properties used in the edge reinforcement check with values directly computable from EN 1992-1-1 [7] Clauses 9.2.1.1 and 8.2(2).

The table lists the concrete and reinforcement material properties used in the edge reinforcement and anchorage length check, computed automatically by the framework. All values match those obtained from a direct manual EN 1992-1-1 calculation.

Table 3. Edge reinforcement check: material properties.

Property	Framework	EC2 (Manual)
fctm	2.90 MPa	2.90 MPa
fctk,0.05	2.03 MPa	2.03 MPa
fbd	3.04 MPa	3.04 MPa
As,min	289 mm^2/m	289 mm^2/m
ρmin	0.1506%	0.1506%

3.6. Assessment

The following conclusions may be drawn from the Structville-based comparison. The bottom flexural reinforcement agrees with the reference within 10% in both directions, which is satisfactory given the different modelling approaches of the two methods. The punching shear resistance and the edge reinforcement material properties agree exactly with the EC2 [7] manual calculation. Automatic application of the β eccentricity factors and column classification performed correctly.

The top reinforcement yields higher values from the FEM for methodological reasons. This is a known and expected property of linear elastic analysis [56], not an implementation error. Overall, the validation confirms that the framework implements the EC2 [7] requirements correctly and that its results are consistent with the published reference.

4. Benchmark Studies

4.1. Slab with Non-Standard Geometry: Application Example

4.1.1. Geometry and Loading

To illustrate the more general applicability of the framework, a design problem is presented for a regular square-plan reinforced concrete flat slab with a mixed support system. The slab plan is 4.0 m × 4.0 m, thickness 250 mm, concrete C25/30 and reinforcing steel B500B; cover top and bottom ct = cb = 25 mm, bar diameters dxb = dyb = 12 mm. Loads: self-weight 6.25 kN/m^2 (computed automatically), superimposed dead load 1.50 kN/m^2, imposed load 3.00 kN/m^2, snow load 1.00 kN/m^2 (Figure 6).



Figure 6. Slab floor plan.

The figure shows the 4.0 m × 4.0 m square slab with its mixed support system: a fixed line support along the top edge and two fixed column supports at the bottom corners.

This configuration was chosen to test the framework on an asymmetric, non-standard support arrangement.

The support system is mixed: a fixed line support runs along the top edge; two fixed column supports are placed at the two bottom corners at coordinates (0.0; 0.0) and (4.0; 0.0) m, both 300 × 300 mm. The framework automatically generates 16 load combinations per EN 1990 [45]: 10 ULS persistent/transient, 4 accidental, and 2 SLS quasi-permanent. The finite element mesh contains 150 elements.

4.1.2. Reinforcement Results

Table 4 summarises the ULS envelope reinforcement results, the percentile-based reinforcement plan, and the SLS check results.

Table 4. Summary of results for Problem 4.1.

Result	Value
Bottom reinforcement X max (Asbx)	211 mm^2/m
Bottom reinforcement Y max (Asby)	239 mm^2/m
Top reinforcement X max (Astx)	378 mm^2/m
Top reinforcement Y max (Asty)	348 mm^2/m
As,min	292 mm^2/m
Base mesh: all directions	Ø8/15 cm = 335 mm^2/m
Governing deflection (SLS)	0.41 mm (wmax = 16.0 mm)
Crack width X	wk = 0.183 mm ($\leq$0.30 mm)
Crack width Y	wk = 0.214 mm ($\leq$0.30 mm)

The bottom reinforcement is governed by the minimum reinforcement ratio As,min in both directions. The top reinforcement demand (Astx = 378 mm^2/m, Asty = 348 mm^2/m) exceeds As,min locally near the fixed wall and corner columns. The top reinforcement demands (Astx = 378 mm^2/m, Asty = 348 mm^2/m) locally exceed the base reinforcement requirement near the fixed wall and corner columns. However, the 75th-percentile governing demand within the corresponding supplementary zones remains below the base mesh capacity of 335 mm^2/m; consequently, no supplementary reinforcement is assigned by the percentile-based zoning procedure for this benchmark. The base mesh therefore satisfies the adopted percentile-based design criterion in all four reinforcement directions. The local envelope maxima of 378 and 348 mm^2/m are not explicitly covered by the resulting reinforcement layout and are therefore not claimed to be fully satisfied under the default p = 75% setting. Both deflection and crack width remain well within their limits.

The maximum bottom reinforcement values (211–239 mm^2/m) fall below As,min (292 mm^2/m), so the minimum geometric reinforcement governs the flexural demand throughout the slab. The top reinforcement concentrates near the fixed wall and the corner columns, consistent with the expected local peak behaviour of a linear elastic FEM [56].

4.1.3. Punching Shear Check

Table 5 shows the punching shear check results for both column supports. The effective depth is d = 213 mm (C25/30, ct = 25 mm, dxt = dyt = 12 mm); the shear resistance is vRd,c = 0.248 MPa.

vEd > vRd,c at both corner columns; punching shear reinforcement is required. The framework automatically determines the required number of reinforcement rings (1 ring) and the governing steel area (Asw = 67 mm^2/ring, minimum 4 × Ø8). The table shows the punching shear verification at the two corner columns of the mixed-support slab, where the design shear stress vEd exceeds the resistance vRd,c. This confirms that punching

shear reinforcement is required at both corners, consistent with the concentrated reaction expected at a corner not restrained by the fixed wall.

Table 5. Punching shear check results.

Column	Type	NEd [kN]	β	vEd [MPa]	vRd,c [MPa]
(0.0; 0.0)	Corner	52.0	1.50	0.263	0.248
(4.0; 0.0)	Corner	52.0	1.50	0.263	0.248

4.1.4. Assessment

This problem illustrates the framework behaviour for a mixed-support slab governed by As,min. The bottom reinforcement is governed by As,min in both directions, indicating a favourable loading level. The top reinforcement (Astx = 378 mm^2/m, Asty = 348 mm^2/m) exceeds As,min locally near the fixed wall and corner columns; however, the 75th-percentile governing value within these zones does not exceed the base mesh capacity, so no supplementary reinforcement is assigned for this benchmark (see Section 4.1.2).

4.2. L-Shaped Slab with Re-Entrant Corner

4.2.1. Geometry and Loading

The second problem is the design of an L-shaped reinforced concrete flat slab illustrated in Figure 7. The floor plan has six vertices at coordinates (0; 9), (0; 0), (10; 0), (10; 3), (4; 3), and (4; 9) m. The slab area is 54.0 m^2, thickness 300 mm, concrete C25/30, reinforcing steel B500B; cover cb = 25 mm, bar diameters dxb = dyb = 12 mm.

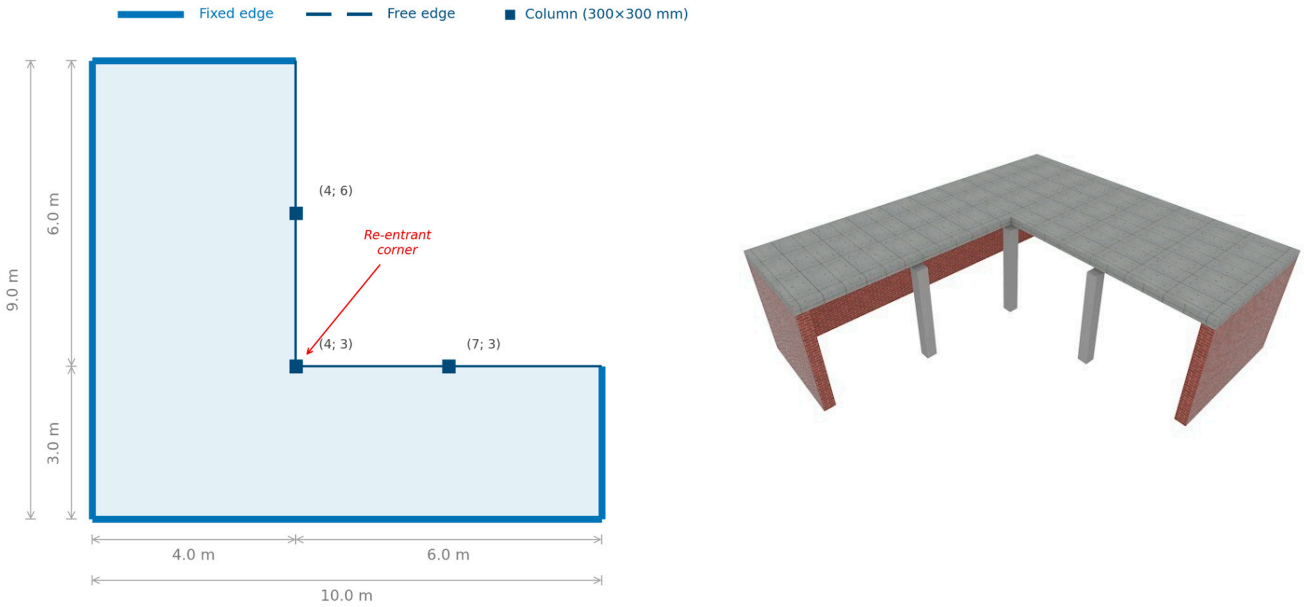


Figure 7. L-shaped flat slab with re-entrant corner. The figure shows the six-vertex, 54.0 m^2 L-shaped floor plan, including the re-entrant corner and the three column and four line supports used in Benchmark 4.2. This geometry demonstrates the framework's ability to handle irregular outlines that the manual equivalent-frame method cannot process directly.

The support system is mixed: four edges carry fixed line supports (E1: left edge 9.0 m, E2: bottom edge 10.0 m, E3: right lower edge 3.0 m, E6: top edge 4.0 m), while the two inner edges at the re-entrant corner (E4 and E5) are free. Three fixed column supports are placed at coordinates (4.0; 3.0), (4.0; 6.0), and (7.0; 3.0) m, all 300 × 300 mm. Loads: self-weight 7.50 kN/m^2 (automatic), superimposed dead load 1.50 kN/m^2, imposed load 3.00 kN/m^2,

snow load 1.00 kN/m^2. The framework generates 16 combinations automatically (10 ULS, 4 accidental, 2 SLS quasi-permanent).

4.2.2. Reinforcement Results

Table 6 summarises the main results.

Table 6. Summary of results for Problem 4.2.

Result	Value
Bottom reinforcement X max (Asbx)	359 mm^2/m (=As,min)
Bottom reinforcement Y max (Asby)	359 mm^2/m (=As,min)
Top reinforcement X max (Astx)	359 mm^2/m (=As,min)
Top reinforcement Y max (Asty)	359 mm^2/m (=As,min)
As, min	359 mm^2/m
Base mesh: all directions	Ø12/30 cm = 377 mm^2/m
Governing deflection (SLS)	0.16 mm (wmax = 12.0 mm)
Crack width X	wk = 0.157 mm ($\leq$0.30 mm)
Crack width Y	wk = 0.157 mm ($\leq$0.30 mm)

The table summarises the reinforcement, punching shear, and serviceability results for the L-shaped slab. Both flexural reinforcement directions are governed by As,min, so the framework recommends a uniform base mesh without any supplementary reinforcement.

Bottom and top reinforcement values are governed by As,min (359 mm^2/m) in both directions, indicating that for a 300 mm slab of C25/30 concrete, the minimum geometric reinforcement covers the flexural demand throughout. The loading level and the stiff support arrangement result in relatively modest moments. The framework therefore recommends a uniform base mesh (Ø12/30 cm = 377 mm^2/m) without supplementary reinforcement. This result illustrates that the framework does not design solely on the basis of internal forces but also enforces code minimum requirements automatically, clearly indicating to the engineer when these govern.

4.2.3. Punching Shear Check

Table 7 shows the punching shear results for all three column supports. The effective depth is d = 263 mm (C25/30, ct = 25 mm, dxt = dyt = 12 mm); the shear resistance is vRd,c = 0.216 MPa.

Table 7. Punching shear check results: Problem 4.2.

Column	Type	NEd [kN]	β	vEd [MPa]	vRd,c [MPa]
(4.0; 3.0)	Corner	126.5	1.50	0.437	0.216
(4.0; 6.0)	Edge	83.2	1.40	0.123	0.216
(7.0; 3.0)	Edge	61.9	1.40	0.091	0.216

Following the distance-to-contour classification fix (Section 2.7.1), the framework correctly identifies the column at the re-entrant corner (4.0; 3.0) as a corner column, while the remaining two columns (4.0; 6.0) and (7.0; 3.0), which lie on the free edges of the re-entrant notch, are correctly classified as edge columns. The corner column requires punching shear reinforcement (vEd = 0.437 MPa > vRd,c = 0.216 MPa); the framework automatically determined 3 reinforcement rings, with a governing shear reinforcement area of 380.9 mm^2 per ring. The two edge columns remain adequate without shear reinforcement (vEd = 0.123 and 0.091 MPa, respectively, both below vRd,c = 0.216 MPa).

4.2.4. Assessment

This problem demonstrates the framework's ability to handle complex, irregular slab geometries. The L-shaped floor plan with re-entrant corner is a geometry that the manual equivalent frame method cannot handle directly; the FEM-based approach treats it naturally. The geometry-aware column classification (Section 2.7.1) correctly identified the column at the re-entrant corner as a corner-type support and the two remaining columns, located on the free edges of the notch, as edge-type supports—a distinction a bounding-box-based classifier would have missed. This correctly triggered a punching shear reinforcement requirement at the corner column, which a naive (interior) classification would have missed entirely. The framework performed all checks automatically within a single unified workflow: flexural reinforcement, punching shear (including the required reinforcement rings), edge reinforcement, deflection, and crack width.

4.3. Rectangular Flat Slab

4.3.1. Geometry and Loading

The third problem is the design of a regular rectangular reinforced concrete flat slab, illustrating primarily the crack width check and its feedback to the reinforcement plan. The floor plan has four vertices at coordinates (0; 8), (0; 0), (12; 0), and (12; 8) m. The slab area is 96.0 m^2, thickness 300 mm, concrete C30/37, reinforcing steel B500B; cover top and bottom ct = cb = 30 mm, bar diameters dxb = dyb = 8 mm. The slab contains an internal opening of 1 × 2 m at coordinates (1; 4)–(2; 6), Figure 8.

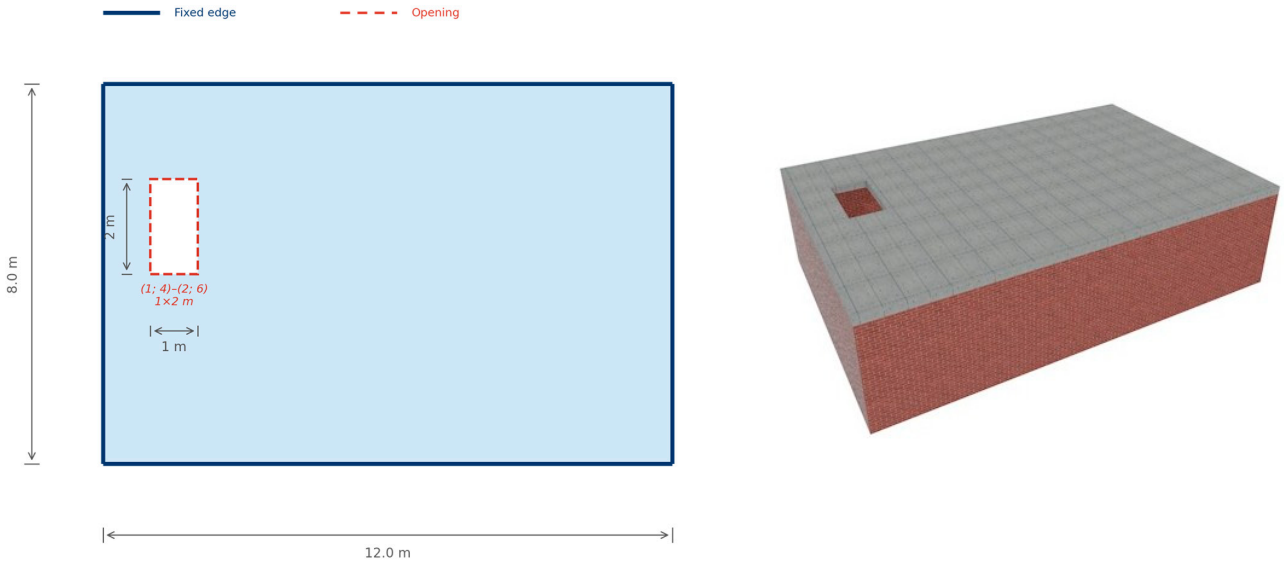


Figure 8. Rectangular flat slab with internal opening. The figure shows the 12 m × 8 m rectangular slab with a 1 × 2 m internal opening, fully supported on all four edges, used in Benchmark 4.3. This geometry was selected primarily to exercise the crack width check and its influence on the resulting reinforcement plan.

Loads: self-weight 7.50 kN/m^2 (automatic), superimposed dead load 1.50 kN/m^2, imposed load 3.00 kN/m^2, snow load 1.00 kN/m^2. The framework generates 16 combinations automatically (10 ULS, 4 accidental, 2 SLS quasi-permanent). All four edges carry fixed line supports; no column supports are present; the finite element mesh contains 882 elements.

4.3.2. Reinforcement Results

Table 8 summarises the main results.

Table 8. Summary of results for Problem 4.3.

Result	Value
Bottom reinforcement X max (Asbx)	196 mm^2/m
Bottom reinforcement Y max (Asby)	383 mm^2/m
Top reinforcement X max (Astx)	497 mm^2/m
Top reinforcement Y max (Asty)	718 mm^2/m
As,min	398 mm^2/m
Base mesh: Bottom X (ULS)	Ø8/10 cm = 503 mm^2/m
Base mesh: Bottom Y (increased for crack width)	Ø10/10 cm = 785 mm^2/m
Base mesh: Top X	Ø8/10 cm = 503 mm^2/m
Base mesh: Top Y	Ø16/30 cm = 670 mm^2/m
Governing deflection (SLS)	1.18 mm (wmax = 48.0 mm)
Crack width X (Ø8/10cm)	wk = 0.242 mm ($\leq$0.30 mm)
Crack width Y (Ø10/10cm)	wk = 0.181 mm ($\leq$0.30 mm)

The table summarises the reinforcement and serviceability results for the rectangular slab with an internal opening, highlighting the governing crack width check.

The ULS envelope bottom reinforcement is governed by As,min (398 mm^2/m) in both directions, so the framework recommends a uniform base mesh. The top reinforcement envelope maxima (Astx = 497 mm^2/m, Asty = 718 mm^2/m) exceed As,min locally near the opening and the fixed edges. The framework does not assign supplementary reinforcement for this benchmark: the 75th-percentile governing values within the high-demand zones remain within the capacity of the base mesh selected for each direction (Ø8/10 cm = 503 mm^2/m for top X; Ø16/30 cm = 670 mm^2/m for top Y, which was increased by the crack width feedback mechanism). The reported envelope maxima represent isolated element-wise peaks that are excluded by the percentile-based design rule (Section 2.6). During the crack width check, however, the original Ø8/10 cm base mesh in the Y direction does not satisfy the limit (wk = 0.333 mm > 0.30 mm) due to the larger moment My,max = 41.35 kNm/m; the system therefore automatically increases the Y-direction base mesh to Ø10/10 cm (wk = 0.181 mm $\leq$ 0.30 mm). This feedback mechanism ensures that the SLS crack width criterion is automatically enforced beyond the ULS-driven reinforcement design.

4.3.3. Assessment

This problem illustrates the automatic feedback of crack width control into the reinforcement design. The ULS envelope bottom reinforcement is governed by As,min in both directions. The loading level and the stiff four-edge restraint result in relatively modest moments, while the top reinforcement locally exceeds As,min near the supports and is handled by the supplementary mesh. During the SLS crack width check, however, the larger moment in the Y direction (My,max = 41.35 kNm/m) causes the original base mesh to fail the limit. The framework automatically identifies the compliant mesh (Ø10/10 cm) and reflects this in the visualisation. This mechanism ensures that the engineer does not receive a base mesh recommendation that fails the SLS crack width criterion, while the ULS check remains valid. Since the slab rests entirely on fixed line supports with no column supports, no punching shear check is required; the framework identifies this automatically and skips the punching check.

5. Discussion

5.1. Scope and Limitations

The present framework was developed for the design of reinforced concrete flat slabs under gravity loading using linear elastic finite element analysis. Identifying the limitations of the method also defines the directions for future development.

Firstly, the percentile-based zoning of Section 2.6. explicitly excludes extreme local peaks from the supplementary-zone design value by construction; while this reflects established design practice, it means the framework does not, by default, guarantee coverage of the absolute worst-case element within that zone unless p is set to 100%.

Numerical validation was performed on the Structville reference problem [55] described in Section 3, whose manual solution is available through the equivalent frame method per EN 1992-1-1 [7]. The comparison yielded the following main findings. The bottom flexural reinforcement agrees within 10% in both directions (FEM: 674–732 mm^2/m; manual: ~727 mm^2/m). The punching shear resistance (vRd,c = 0.399 MPa) and the edge reinforcement material properties agree exactly with the EC2 [7] manual calculation. For the top reinforcement, the FEM envelope value (2278–2595 mm^2/m) considerably exceeds the equivalent frame result (~1000 mm^2/m), which is a methodological difference: the linear elastic FEM identifies local stress concentrations above column heads, whereas the equivalent frame method distributes these uniformly across the column strip width [56].

The two additional application examples in Section 4 demonstrate the framework's extensibility to more general geometries. In the first example (Section 4.1), a 4.0 m × 4.0 m mixed-support slab with a fixed line support and two corner columns was designed, representing a support configuration different from the Structville problem. The results are consistent: punching shear reinforcement was required at both corner columns (vEd = 0.263 MPa > vRd,c = 0.248 MPa), while deflection (wEd = 0.41 mm, utilisation 2.5%) and crack width (wk $\leq$ 0.214 mm) both satisfy the EC2 [7] requirements.

In the second example (Section 4.2), an L-shaped slab with a re-entrant corner (10.0 m × 9.0 m, h = 300 mm, C25/30) was examined. This is a geometry the manual equivalent frame method cannot handle directly. Following the distance-to-contour classification revision (Section 2.7.1), the framework correctly identified the column at the re-entrant corner (4.0; 3.0) as a corner column and the remaining two (4.0; 6.0) and (7.0; 3.0) as edge columns, rather than classifying all three as interior. The corner column requires punching shear reinforcement (vEd = 0.437 MPa > vRd,c = 0.216 MPa); the framework automatically determined 3 reinforcement rings, with a governing shear reinforcement area of 380.9 mm^2 per ring. The two edge columns remain adequate without shear reinforcement (vEd = 0.123 and 0.091 MPa, respectively, both below vRd,c = 0.216 MPa). Where the re-entrant corner coincides with a column position, as in this example, the framework's column-based punching check takes precedence over the geometry-only re-entrant corner estimate of EC2 Clause 6.4.6 (Section 2.7.1); the latter's detection logic was verified separately by confirming it correctly reports no unassigned re-entrant vertex once the coincident column has been accounted for. Edge reinforcement sizing along the free edges of the re-entrant corner, and the deflection (wEd = 0.16 mm) and crack width (wk = 0.157 mm) checks, all passed. Total execution time was 1–2 min across all four problems.

Of the four problems presented, three involve slabs without openings; the fourth (Section 4.3) includes an internal opening, demonstrating the framework's suitability for such cases. The geometry-definition canvas handles arbitrary floor plans, including slanted edges and internal openings (stairwells, lift shafts), without restriction. At re-entrant corner columns and near openings, the experimental literature documents more complex punching behaviour [10,11] that can only be simulated with limited accuracy using linear elastic models.

Secondly, the linear elastic finite element model does not account for moment redistribution due to cracking or plastic behaviour [57,58], which is the basis of the yield-line design procedure of Eurocode 2 Clause 5.6. The percentile-based zoning algorithm applied in the framework partially mitigates the conservatism of elastic analysis by not treating local stress peaks, for example near re-entrant corners, as governing over the entire slab. However, code-based justification of this approach is not currently available. The limitations of linear elastic FEM for flat slab design are discussed in detail by S. A. Skorpen [56]. The relationship between parametric flat slab design and material consumption optimisation has been studied by Jayasinghe et al. [29].

A further limitation concerns the deflection check (Section 2.7.3), which is based on the uncracked linear elastic stiffness of the global finite element model rather than a cracked or effective stiffness. Consistent with EN 1992-1-1 Clause 7.4.3, a more rigorous treatment would apply reduced stiffness to account for cracking, which the current implementation does not perform. This may lead to non-conservative (underestimated) deflection results for slabs with significant cracking, and represents a priority direction for future refinement.

Thirdly, the present implementation handles a single slab domain. Modelling structures with varying slab thicknesses, band beams, or ribs currently requires modification of the program.

It should be noted that only the regular, symmetric Structville geometry was validated against an independently published reference solution; the three additional benchmark problems (Section 4), which showcase the framework's handling of irregular geometries, were verified for internal consistency and code compliance but not against an independent external reference, since no directly comparable published solutions for these specific geometries were identified. Establishing such independent benchmarks for irregular flat slab geometries remains an important direction for future validation work.

A mesh-convergence check was performed on Benchmark 1 (Section 4.1) using three element sizes (0.75 m, 0.50 m default, and 0.25 m; 86, 150, and 598 elements respectively). Reaction-based quantities converged well: the punching shear design stress and resistance changed by less than 3% across all three meshes (vEd: 0.265, 0.263, 0.258 MPa; vRd,c: 0.247, 0.249, 0.250 MPa), as did the bottom reinforcement envelope (within 3%). The envelope maximum of the top reinforcement, by contrast, did not converge: it increased monotonically with mesh refinement (Astx_max: 267, 378, 840 mm^2/m; Asty_max: 299, 348, 771 mm^2/m), consistent with the well-known moment singularity that arises at point column supports in plate and shell finite element models, where the theoretical bending moment directly at the support is unbounded and the discretised peak value grows without limit as the mesh is refined. This finding is not a limitation specific to the present framework but a general property of linear elastic FEM at point supports; it reinforces, rather than undermines, the rationale for the percentile-based zoning approach of Section 2.6., since it confirms that the raw envelope maximum is not a mesh-independent quantity and is therefore unsuitable as a direct design value regardless of computational effort. No sensitivity analysis was performed on the nominal in-plane restraint stiffness introduced in Section 2.2.; given the several-orders-of-magnitude separation between flexural and nominal in-plane stiffness, this is expected to be immaterial, but has not been verified quantitatively.

5.2. Planned Extensions

The framework can be extended naturally in several directions. A medium-term development goal is BIM integration. The IfcOpenShell library [13,26] would allow slab contours, column positions, and material properties to be read automatically from an IFC model, making the interactive geometric drawing step redundant for projects where a detailed architectural BIM model is already available.

In future, key design parameters, such as slab thickness, support positions, and geometry, could be varied automatically by an optimisation algorithm targeting minimisation of total reinforcing steel consumption subject to code compliance. The short 1–2 min execution time makes this computationally feasible. Genetic algorithms [59], evolutionary methods [60], and other metaheuristic procedures [61] have all been applied successfully to flat slab system optimisation.

5.3. Contribution to Engineering Practice

A key practical advantage of the framework is that the entire design process is reproducible and auditable. Every computed quantity, from the punching control perimeter to the anchorage length, is directly traceable to a Python expression that can be inspected and verified at any time. This represents a significant difference from the post-processing modules of commercial finite element programs, where intermediate calculation steps are frequently not visible to the user. The Jupyter Notebook format allows explanatory text, Python code, and numerical results to coexist in the same document, establishing an unambiguous link between the EN 1992-1-1 expressions and their practical implementation.

To position the present contribution relative to prior automated reinforced concrete design tools, Table 9 compares the scope, geometric flexibility, and code-integration level of the framework against representative recent studies.

Table 9. Comparison with representative automated RC design frameworks.

Study	Structural Element	Geometry Flexibility	Automated Code Checks	Automation Interface
Sherif et al. [12]	RC buildings (general)	Regular/BIM-derived	Cost optimisation only	BIM-based
Singh et al. [13]	RC slabs	BIM-derived	Partial	BIM-based (IfcOpenShell)
Bruno et al. [25]	General (SHM)	N/A (model updating)	None	Python–SAP2000 COM
Roszevák and Haris [27]	RC flat slabs	Regular	None (analysis only)	Manual FEM
Farid et al. [28]	Post-tensioned slabs	Regular	Partial (flexure only)	Automated optimisation
Xu et al. [30]	RC slabs (prefabricated)	BIM-derived	None (detailing only)	BIM + evolutionary algorithm
Izhar et al. [31]	RC frames	Regular	Partial (frame analysis)	Python, open-source
Wefki et al. [39]	RC elements (general)	Regular	Partial (optimisation-driven)	Generative design + multi-objective optimisation
Present framework	RC flat slabs	Arbitrary, incl. non-convex (L-shape, re-entrant corners, openings)	Full EC2 suite (punching shear, edge reinforcement/anchorage, deflection, crack width)	Python–COM, script-driven, no BIM required

6. Conclusions

The principal scientific and methodological contributions of the presented framework are as follows:

1. An interactive geometric modelling interface that enables flat slabs with arbitrary floor plans, including slanted edges and internal openings, to be defined without an external BIM model. This eliminates the dependency on pre-existing architectural BIM models and allows the structural engineer to begin the design process directly from the Jupyter Notebook environment, which is particularly valuable in early-stage design where detailed architectural models may not yet be available. The interface

also enables the handling of non-standard, non-convex geometries—such as L-shaped floor plans and re-entrant corners—that are difficult or impossible to define through the standard graphical interfaces of conventional finite element packages.

2. Fully automated model generation via the COM API, reducing model-construction time compared with manual workflows by an order of magnitude. This removes the most time-consuming and error-prone step of the conventional design process—the manual, click-by-click construction of the finite element model through the software's graphical interface—and replaces it with a deterministic, version-controllable script that can be re-run identically at any time. The resulting models are fully reproducible and auditable, which is a significant advantage for both engineering practice and research documentation.
3. Automatic generation of Eurocode load combinations (ULS, SLS, accidental) from load groups and combination factors entered once. This ensures consistent, auditable application of EN 1990 across all limit states and eliminates the transcription errors that commonly arise during manual combination assembly, particularly when multiple variable loads are present and the number of required combinations grows combinatorially.
4. Element-wise ULS envelope reinforcement determination, ensuring no critical slab region is overlooked. Unlike the equivalent frame method, which distributes moments uniformly across column and middle strips, the element-wise envelope captures local stress concentrations and provides a spatially resolved representation of reinforcement demand across the entire slab. This is a more accurate basis for design, particularly in non-regular geometries where strip-based simplifications may not be applicable.
5. A percentile-based reinforcement zoning algorithm converting FEM results into practically constructable, two-level reinforcement plans. This is a key methodological contribution of the paper: the algorithm formalises the transformation from continuous finite element output to a discrete, constructable reinforcement layout—a design step that every structural engineer performs implicitly during manual interpretation of FEM results, yet one that has not previously been documented as a reproducible, parameterised algorithm in the published literature. The percentile-based approach provides a principled and user-adjustable balance between design economy and coverage of local demand peaks, as validated by the mesh-convergence study (Section 5.1).
6. Automatic, geometry-aware column position classification enabling position-dependent punching shear verification. The distance-to-contour classification algorithm correctly handles non-convex slab geometries such as L-shapes and re-entrant corners, where conventional bounding-box-based approaches would misclassify columns and produce non-conservative punching shear results. This was demonstrated in the L-shaped slab benchmark (Section 4.2), where the algorithm correctly identified a corner column at the re-entrant corner that a naive bounding-box classifier would have missed entirely, triggering a punching shear reinforcement requirement that would otherwise have been overlooked.
7. Integrated edge reinforcement, anchorage length, deflection, and crack width checks within the same workflow. This ensures that the engineer receives a comprehensive assessment from a single workflow run, rather than having to perform separate manual verifications for each limit state. The crack width check (Section 2.7.4) includes an automatic feedback mechanism that increases the base reinforcement when the SLS crack width limit is exceeded, as demonstrated in the opening benchmark (Section 4.3).

The complete calculation workflow is typically executable in 1–2 min on a standard engineering laptop computer (Intel Core i7, 16 GB RAM). This represents a substantial time saving compared with conventional manual modelling, while reducing the risk of

errors from manual data entry and ensuring the full reproducibility and auditability of the workflow.

In its current form, the framework is applicable to design problems based on linear elastic analysis under gravity loading. Numerical validation was performed on four problems: the Structville reference problem, an asymmetric mixed-support slab, an L-shaped slab, and a rectangular slab with an internal opening governed by crack width control. In each case, the results were consistent with the EC2 [7] requirements for the checks implemented; the deflection verification is based on uncracked elastic stiffness and should be interpreted as a screening-level indicator (Section 5.1). Planned extensions include BIM integration via IFC import, exploration of optimisation opportunities, and the incorporation of non-linear analysis. A critical assessment of the EC2 deflection check method has been carried out by D'Antino and Pisani [62].

Overall, the presented framework demonstrates that the open Python ecosystem, Jupyter Notebooks, and the programming interfaces of commercial finite element software can be effectively combined into reproducible engineering workflows that serve both research and practical structural design, with scope for further development.

Author Contributions: Conceptualization, D.G.; Methodology, D.G.; Software, A.D.T., F.I. and D.G.; Validation, A.D.T. and F.I.; Formal analysis, A.D.T., F.I. and D.G.; Investigation, A.D.T. and F.I.; Resources, A.D.T. and F.I.; Data curation, D.G.; Writing—original draft, A.D.T.; Writing—review & editing, A.D.T. and D.G.; Visualization, F.I. and D.G.; Supervision, J.S. and D.G.; Project administration, J.S. and D.G.; Funding acquisition, D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are not publicly available because the framework relies on a proprietary commercial FEM software license (AxisVM), but the source code is available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Park, R.; Gamble, W.L. *Reinforced Concrete Slabs*, 2nd ed.; Wiley: New York, NY, USA, 2000.
2. Beeby, A.W.; Narayanan, R.S. *Designers' Guide to Eurocode 2: Design of Concrete Structures*; Thomas Telford: London, UK, 2005.
3. Moe, J. *Shearing Strength of Reinforced Concrete Slabs and Footings Under Concentrated Loads*; Bulletin D47; Portland Cement Association: Washington, DC, USA, 1961.
4. Sharaky, S.T.; Salam, A.A.E.; Nafie, A.; Kersh, M.A.A. Punching Shear of Slab-Column Connections with Beams Attached to One or Two Sides. *Structures* **2023**, *57*, 105216. [CrossRef]
5. Fraile, D.H.; Setiawan, A.; Santos, J.B.; Muttoni, A. Punching Tests on Edge Slab-Column Connections with Refined Measurements. *Eng. Struct.* **2023**, *288*, 116166. [CrossRef]
6. Muttoni, A. Punching Shear Strength of Reinforced Concrete Slabs Without Transverse Reinforcement. *ACI Struct. J.* **2008**, *105*, 440–450. [CrossRef] [PubMed]
7. *EN 1992-1-1*; Eurocode 2—Design of Concrete Structures—Part 1-1: General Rules and Rules for Buildings. European Committee for Standardization (CEN): Brussels, Belgium, 2004.
8. *EN 1992-1-1:2023*; Eurocode 2—Design of Concrete Structures—Part 1-1: General Rules. European Committee for Standardization (CEN): Brussels, Belgium, 2023.
9. de Paula Randi, R.; Trautwein, L.M.; Marques, M.G.; Ramos, A.P. On the Behavior of Flat Slabs with Asymmetric Openings Near Columns, Structural Concrete. *Struct. Concr.* **2025**, *27*, 73–95. [CrossRef]
10. Siqueira, J.P.D.A.; de Albuquerque, E.J.P.; Ruiz, M.F.; Melo, G.S. Punching Shear Tests in Flat Slabs Supported on Re-entrant Corner Columns. *Eng. Struct.* **2024**, *308*, 117954. [CrossRef]
11. Ebeido, T.; Soussa, M.S.; Diab, M.A. Punching Shear of Edge Column–Slab Connection Subjected to Unbalanced Moment. *J. Eng. Appl. Sci.* **2025**, *72*, 47. [CrossRef]
12. Sherif, M.; Nassar, K.; Hosny, O.; Safar, S.; Abotaleb, I. Automated BIM-Based Structural Design and Cost Optimization Model for Reinforced Concrete Buildings. *Sci. Rep.* **2022**, *12*, 21616. [CrossRef] [PubMed]

13. Singh, T.; Mahmoodian, M.; Wang, S. Advancing Smart Construction Through BIM-Enabled Automation in Reinforced Concrete Slab Design. *Buildings* **2025**, *15*, 343. [CrossRef]
14. Brown, N.C.; Jusiega, V.; Mueller, C.T. Implementing Data-Driven Parametric Building Design with a Flexible Toolbox. *Autom. Constr.* **2020**, *118*, 103252. [CrossRef]
15. Wimmer, J.; Braml, T. Digital Twins for Engineering Structures—An Industry 4.0 Perspective. *Struct. Concr.* **2024**, *25*, 4202–4218. [CrossRef]
16. Deák, G.; Draskóczy, A.; Dulácska, E.; Kollár, L.; Visnovitz, G. *Vasbeton Szerkezetek—Tervezés az Eurocode Alapján*, 4th ed.; Artifex: Budapest, Hungary, 2018.
17. Degertekin, S.O.; Hayalioglu, M.S.; Gorgun, H. Optimum Design of Geometrically Non-Linear Steel Frames with Semi-Rigid Connections Using a Harmony Search Algorithm. *Steel Compos. Struct.* **2009**, *9*, 535–555. [CrossRef]
18. Degertekin, S.O.; Hayalioglu, M.S.; Gorgun, H. Optimum Design of Steel Frames Using Harmony Search Algorithm. *Struct. Multidiscip. Optim.* **2008**, *36*, 393–401. [CrossRef]
19. Martins, J.P.; Correia, J.; Ljubinković, F.; Simões da Silva, L. Cost Optimisation of Steel I-Girder Cross-Sections Using Genetic Algorithms. *Structures* **2023**, *55*, 379–388. [CrossRef]
20. Girardet, A.; Boton, C. A Parametric BIM Approach to Foster Bridge Project Design and Analysis. *Autom. Constr.* **2021**, *126*, 103679. [CrossRef]
21. Liao, W.; Lu, X.; Huang, Y.; Zheng, Z.; Lin, Y. Automated Structural Design of Shear Wall Residential Buildings Using Generative Adversarial Networks. *Autom. Constr.* **2021**, *132*, 103931. [CrossRef]
22. Mangal, M.; Li, M.; Gan, V.J.L.; Cheng, J.C.P. Automated Clash-Free Optimization of Steel Reinforcement in RC Frame Structures Using Building Information Modeling and Two-Stage Genetic Algorithm. *Autom. Constr.* **2021**, *126*, 103676. [CrossRef]
23. Verduzco, L.F.; Horta, J.; y Hernández, M.A.P.L.; Hernández, J.B. CALRECOD—A Software for Computer Aided Learning of Reinforced Concrete Structural Design. *Adv. Eng. Softw.* **2022**, *172*, 103189. [CrossRef]
24. Rabi, M.; Abarkan, I.; Sarfarazi, S.; Ferreira, F.P.V.; Alkherret, A.J. Automated Design and Optimization of Concrete Beams Reinforced with Stainless Steel. *Struct. Concr.* **2025**, *27*, 3247–3270. [CrossRef]
25. Bruno, G.; Ruggieri, S.; Uva, G.; Nettis, A.; Parisi, F. SAP.PY: Automated Model Updating Using SAP2000 OAPI and Python for Structural Health Monitoring. In Proceedings of the 10th European Workshop on Seismic Behaviour of Irregular and Complex Structures, Milan, Italy, 30 June–5 July 2024.
26. Singh, T.; Mahmoodian, M.; Wang, S. Enhancing Open BIM Interoperability: Automated Generation of a Structural Model from an Architectural Model. *Buildings* **2024**, *14*, 2475. [CrossRef]
27. Roszevák, Z.; Haris, I. Finite Element Analyses of Cast-in-Situ RC Flat Slab with an Equivalent Frame for Horizontal Loads. *Period. Polytech. Civ. Eng.* **2024**, *68*, 289–304. [CrossRef]
28. Farid, A.F.; Kamal, A.; Rashed, Y.F. An Automated Optimized Design of Practical Post-Tensioned Slabs. *J. Eng. Appl. Sci.* **2024**, *71*, 161. [CrossRef]
29. Jayasinghe, A.; Orr, J.; Ibell, T.; Boshoff, W.P. Minimising Embodied Carbon in Reinforced Concrete Flat Slabs Through Parametric Design. *J. Build. Eng.* **2022**, *50*, 104136. [CrossRef]
30. Xu, C.; Zheng, X.; Wu, Z.; Zhang, C. Automated Prefabricated Slab Splitting Design Using a Multipopulation Coevolutionary Algorithm and BIM. *Buildings* **2024**, *14*, 433. [CrossRef]
31. Izhar, T.; Ahmad, S.A.; Ahmed, T.; Mumtaz, N. StrucPy.RCFA: Object-Oriented Python Package for Structural Analysis of Reinforced Concrete Frames. *J. Open Res. Softw.* **2024**, *12*, 10. [CrossRef]
32. Shen, Y.; Sun, J.; Liang, S. Interpretable Machine Learning Models for Punching Shear Strength Estimation of FRP Reinforced Concrete Slabs. *Crystals* **2022**, *12*, 259. [CrossRef]
33. Alkhatib, S.; Deifalla, A. Punching Shear Strength of FRP-Reinforced Concrete Slabs Without Shear Reinforcements: A Reliability Assessment. *Polymers* **2022**, *14*, 1743. [CrossRef] [PubMed]
34. Nasab, M.N.; Jahangir, H.; Hasani, H.; Majidi, M.-H.; Khorashadizadeh, S. Estimating the Punching Shear Capacities of Concrete Slabs Reinforced by Steel and FRP Rebars with ANN-Based GUI Toolbox. *Structures* **2023**, *50*, 1204–1221. [CrossRef]
35. Abood, E.A.; Abdallah, M.H.; Alsaadi, M.; Imran, H.; Bernardo, L.F.A.; De Domenico, D.; Henedy, S.N. Machine Learning-Based Prediction Models for Punching Shear Strength of Fiber-Reinforced Polymer Reinforced Concrete Slabs Using a Gradient-Boosted Regression Tree. *Materials* **2024**, *17*, 3964. [CrossRef] [PubMed]
36. Abdallah, M.H.; Thoeny, Z.A.; Henedy, S.N.; Al-Abdaly, N.M.; Imran, H.; Bernardo, L.F.A.; Al-Khafaji, Z. The Machine-Learning-Based Prediction of the Punching Shear Capacity of Reinforced Concrete Flat Slabs: An Advanced M5P Model Tree Approach. *Appl. Sci.* **2023**, *13*, 8325. [CrossRef]
37. Derogar, S.; Ince, C.; Yatbaz, H.Y.; Ever, E. Prediction of Punching Shear Strength of Slab-Column Connections: A Comprehensive Evaluation of Machine Learning and Deep Learning Based Approaches. *Mech. Adv. Mater. Struct.* **2024**, *31*, 1272–1290. [CrossRef]

38. Wu, Y.; Zhou, Y. Prediction and Feature Analysis of Punching Shear Strength of Two-Way Reinforced Concrete Slabs Using Optimized Machine Learning Algorithm and Shapley Additive Explanations. *Mech. Adv. Mater. Struct.* **2022**, *30*, 3086–3096. [CrossRef]
39. Wefki, H.; Salah, M.; Elbeltagi, E.; Alinizzi, M. Generative Design-Driven Optimization for Effective Concrete Structural Systems. *Buildings* **2025**, *15*, 2646. [CrossRef]
40. Inter-CAD Ltd. *AxisVM 20 Reference Manual*; AxisVM: Budapest, Hungary, 2023.
41. Beg, M.; Taka, J.; Kluyver, T.; Konovalov, A.; Ragan-Kelley, M.; Thiéry, N.M. Using Jupyter for Reproducible Scientific Workflows. *Comput. Sci. Eng.* **2021**, *23*, 36–46. [CrossRef]
42. Kluyver, T.; Ragan-Kelleyb, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K. Jupyter Notebooks: A Publishing Format for Reproducible Computational Workflows. In *Proceedings of the ELPUB 2016*; IOS Press: Amsterdam, The Netherlands, 2016; pp. 87–90.
43. Inter-CAD Ltd. Axisvm: A Python Package for AxisVM, PyPI. 2024. Available online: https://pypi.org/project/axisvm/ (accessed on 3 September 2026).
44. *EN 1991-1-1*; Eurocode 1—Actions on Structures—Part 1-1: General Actions. European Committee for Standardization (CEN): Brussels, Belgium, 2002.
45. *EN 1990*; Eurocode—Basis of Structural Design. European Committee for Standardization (CEN): Brussels, Belgium, 2002.
46. Jobbágy, D.; Völgyi, I. Changes in Punching Shear Design of Reinforced Concrete Slabs: Hungarian Standard, Eurocode 2, Model Code 2010. *Vasbetonépítés* **2019**, *21*, 32–39. [CrossRef]
47. Muttoni, A.; Ruiz, M.F.; Simoes, J.T. The Theoretical Principles of the Critical Shear Crack Theory for Punching Shear Failures and Derivation of Consistent Closed-Form Design Expressions. *Struct. Concr.* **2017**, *19*, 174–190. [CrossRef]
48. Vargas, D.; Lantsoght, E.O.L.; Genikomsou, A.S. Flat Slabs in Eccentric Punching Shear: Experimental Database and Code Analysis. *Buildings* **2022**, *12*, 2092. [CrossRef]
49. Abu-Salma, D.; Vollum, R.L.; Macorini, L. Influence of Flexural Reinforcement Detailing on Punching Shear Resistance at Edge and Corner Columns of Flat Slab Buildings. *Structures* **2024**, *61*, 106024. [CrossRef]
50. Fédération Internationale du Béton (fib). *Model Code for Concrete Structures 2010*; fib Bulletins 65/66; Ernst & Sohn: Berlin, Germany, 2013.
51. Ricker, M.; Siburg, C. Punching Shear Strength of Flat Slabs: Critical Review of Eurocode 2 and fib Model Code 2010 Design Provisions. *Struct. Concr.* **2016**, *17*, 457–468. [CrossRef]
52. Liu, T.; Cakiroglu, C.; Islam, K.; Wang, Z.; Nehdi, M.L. Explainable Machine Learning Model for Predicting Punching Shear Strength of FRC Flat Slabs. *Eng. Struct.* **2024**, *301*, 117276. [CrossRef]
53. Mangalathu, S.; Shin, H.; Choi, E.; Jeon, J.-S. Explainable Machine Learning Models for Punching Shear Strength Estimation of Flat Slabs Without Transverse Reinforcement. *J. Build. Eng.* **2021**, *39*, 102300. [CrossRef]
54. Mangalathu, S.; Hwang, S.-H.; Jeon, J.-S. Failure Mode and Effects Analysis of RC Members Based on Machine-Learning-Based SHapley Additive exPlanations (SHAP) Approach. *Eng. Struct.* **2020**, *219*, 110927. [CrossRef]
55. Uzodimma, U.O. *Structville, Structural Design of Flat Slabs to Eurocode 2: A Worked Example*; Structville Research: Awka, Nigeria, 2019. Available online: https://structville.com/structural-design-of-flat-slabs (accessed on 3 September 2026).
56. Skorpen, S.A. The Application and Interpretation of Linear Finite Element Analysis Results in the Design and Detailing of Hogging Moment Regions in Reinforced Concrete Flat Plates. *J. S. Afr. Inst. Civ. Eng.* **2014**, *56*, 2–16.
57. Mosley, B.; Bungey, J.; Hulse, R. *Reinforced Concrete Design to Eurocode 2*, 7th ed.; Palgrave Macmillan: Basingstoke, UK, 2012.
58. Winkler, B.; Hofstetter, G.; Niederwanger, G. Experimental Verification of a Constitutive Model for Concrete Cracking. *Proc. Inst. Mech. Eng. Part L* **2001**, *215*, 75–86. [CrossRef]
59. Aidy, A.; Rady, M.; Mashhour, I.M.; Mahfouz, S.Y. Structural Design Optimization of Flat Slab Hospital Buildings Using Genetic Algorithms. *Buildings* **2022**, *12*, 2195. [CrossRef]
60. Rady, M. Application of Evolutionary Algorithms to Design Economic Flat Slab Buildings Based on the Intended Function. *Sci. Rep.* **2024**, *14*, 8826. [CrossRef] [PubMed]
61. Kaveh, A.; Bakhshpoori, T. *Metaheuristics: Outlines, MATLAB Codes and Examples*; Springer: Cham, Switzerland, 2019.
62. D'Antino, T.; Pisani, M.A. A Proposal to Improve the Effectiveness of the Deflection Control Method Provided by Eurocodes for Concrete, Timber, and Composite Slabs. *Materials* **2021**, *14*, 7627. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.