

A GIS-ASSISTED PYTHON TOOLCHAIN FOR LOCATION-BASED ENVIRONMENTAL PARAMETER RETRIEVAL AND LOAD COMPUTATION BASED ON ROMANIAN CODES

Adrian-Alexandru SAVU - Lecturer, PhD, Technical University of Civil Engineering, Faculty of Civil, Industrial and Agricultural Buildings, e-mail: adrian.a.savu@utcb.ro

Andrei-Dan SABĂU - Assistant Lecturer, PhD candidate, Technical University of Civil Engineering, Faculty of Civil, Industrial and Agricultural Buildings, e-mail: andrei-dan.sabau@phd.utcb.ro

Abstract: Structural engineering design in Romania relies on location-dependent environmental parameters and zonation maps for snow, wind, seismicity, and frost depth. Manual reading of zonation maps is time-consuming, error-prone near zone boundaries, and difficult to reproduce at scale. This paper presents a GIS-assisted Python toolchain that retrieves site parameters directly from georeferenced national layers and unified composite KML overlays, computes snow and wind actions for multiple roof typologies according to CR 1-1-3/2012 and CR 1-1-4/ 2012 and supplies seismic parameters (peak ground acceleration - a_g and response spectra control period - T_c) per P100-1/2013 and frost depth per STAS 6054/77. The workflow integrates updated baseline maps from the CCERS collection, population density and tree cover density for the classification of the snow exposure and terrain conditions, while the frost-depth map is reconstructed in QGIS according to STAS. A second version of each map was created based on the MDLPA official seismic map creation approach for uniform administrative reporting. Two composite overlap maps were also produced to visually validate multi-factor assignments and support transparent, repeatable site characterization. Case studies across Romania demonstrate consistent parameter retrieval for both point coordinates and administrative level aggregation, enabling fast reporting and automated batch evaluations suitable for design offices and research pipelines.

Keywords: GIS, Python, zonation maps, snow load, wind load, seismic hazard, frost depth

1. Introduction

Design actions in Romania are governed by national regulations that provide spatial zonation for climatic and seismic loads. In practice, designers must identify values from multiple maps and then compute corresponding loads and coefficients for the structural configuration. This process becomes cumbersome for distributed portfolios and is sensitive to manual interpretation errors, especially near zonal boundaries or if administrative-unit reporting is required.

This work introduces a reproducible toolchain combining GIS preprocessing (QGIS) with a lightweight Python engine to retrieve location-based parameters (snow, wind, seismic, frost depth) from georeferenced datasets, derive exposure and terrain modifiers from population and tree density layers, and consistently compute building effective loads or load coefficients according to CR 1-1-3/2012 and CR 1-1-4/2012.

2. Data sources and map provenance

2.1 Baseline zonation maps (CCERS)

Baseline zonation layers for seismic and climatic actions were sourced from the CCERS (Seismic Risk Assessment Research Center) resource collection [1], which

provides reproductions of the relevant national zonation maps: the ground snow-load characteristic values map from CR 1-1-3/2012 [2], the dynamic wind pressure reference values map from CR 1-1-4/2012 [3], the peak ground acceleration map and the response spectra control period map from P100-1/2013 [4]. These maps were subsequently updated by the author for consistent integration: georeferencing checks, updated fine-tuned boundaries, and alignment with the toolchain's lookup schema. The updated maps are shown in the figures below (Fig. 1, Fig. 2).

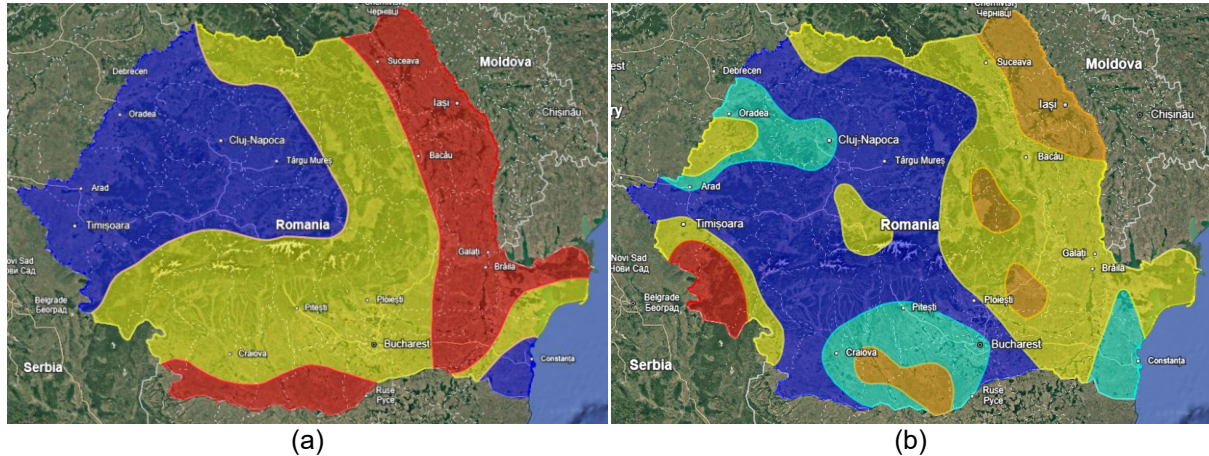


Fig. 1 - Updated ground snow-load characteristic values (a) and dynamic wind pressure reference values (b) zonation maps [1, 2, 3]

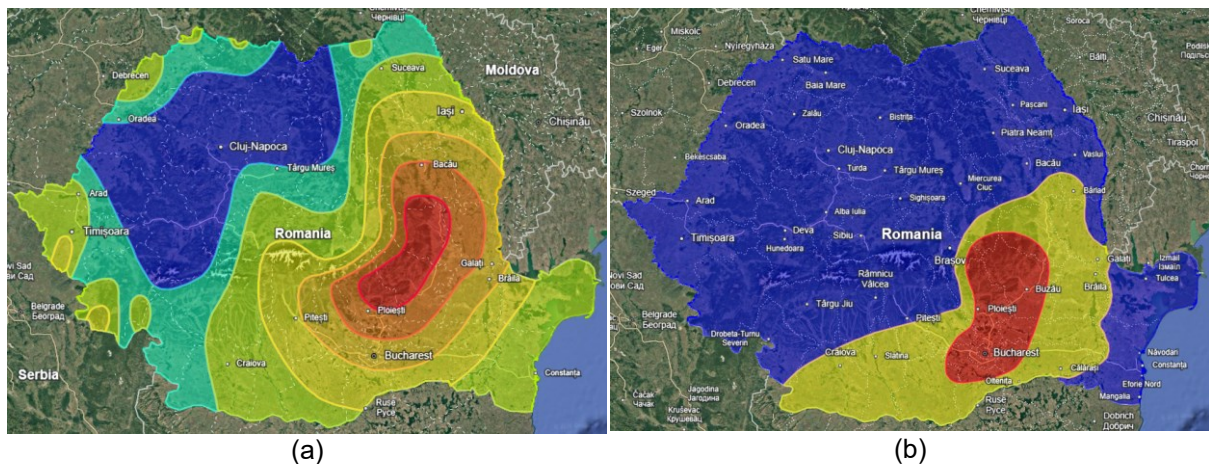


Fig. 2 - Updated peak ground acceleration (a) and response spectra control period (b) zonation maps [1, 4]

2.2 Population density and tree cover density layers (exposure/terrain modifiers)

To classify snow exposure and wind terrain conditions, the toolchain uses a population density layer and a tree cover density layer implemented directly in the toolchain's classification module and runner workflow, presented in Fig. 3. These are used to compute the snow exposure factor C_e and the terrain category (I–IV) used in wind profile calculations.

The population density map [5] is a vector geo-package built from Kontur Population: Global Population Density for 400m H3 Hexagons Vector H3 hexagons with population counts at 400m resolution. It is a fusion of European Union's Copernicus Service Information (GHSL), Facebook HRSL, Microsoft Buildings, Copernicus Global Land Service Land Cover, Land Information New Zealand and OpenStreetMap data.

The tree cover density map [6] is a vectorized raster map clipped to Romania's borders for which the data is available on-line from <http://earthenginepartners.appspot.com/science-2013-global-forest>. The map is also updated for 2023 values generated using European Union's Copernicus Land Monitoring Service information [7].

Several modifications were made to the population and tree cover density map data in order to implement them correctly in the toolchain, such as sieving and merging values for a slight decrease in resolution, while optimizing speed of access.

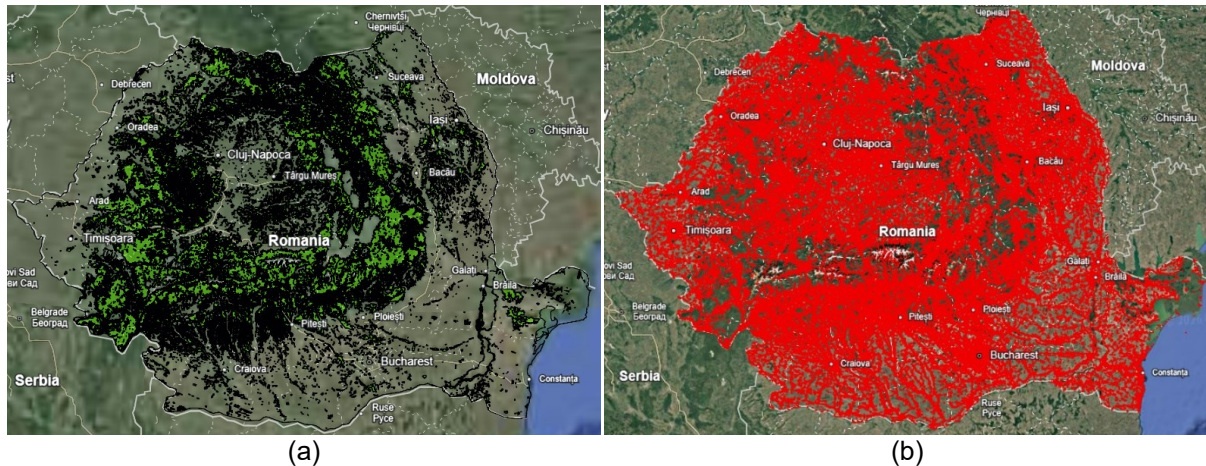


Fig. 3 - Tree cover density (a) and population density (b) maps [7, 5]

2.3 Frost depth map (STAS 6054/77) reconstructed in QGIS

The frost-depth zonation layer was constructed by the author in QGIS according to STAS 6054/77 [8], then exported to GIS/KML formats for automated lookups. It explicitly defines the Danube Delta and the Danube floodplains areas as an exception, being zones for which the maximum frost depth should be established on local observations. The same exception applies to zones for which the altitude is greater than 1000m - the constructed map does not take this into consideration per se, but the toolchain bypasses the map interrogation based on the result from the altitude map (2.4).

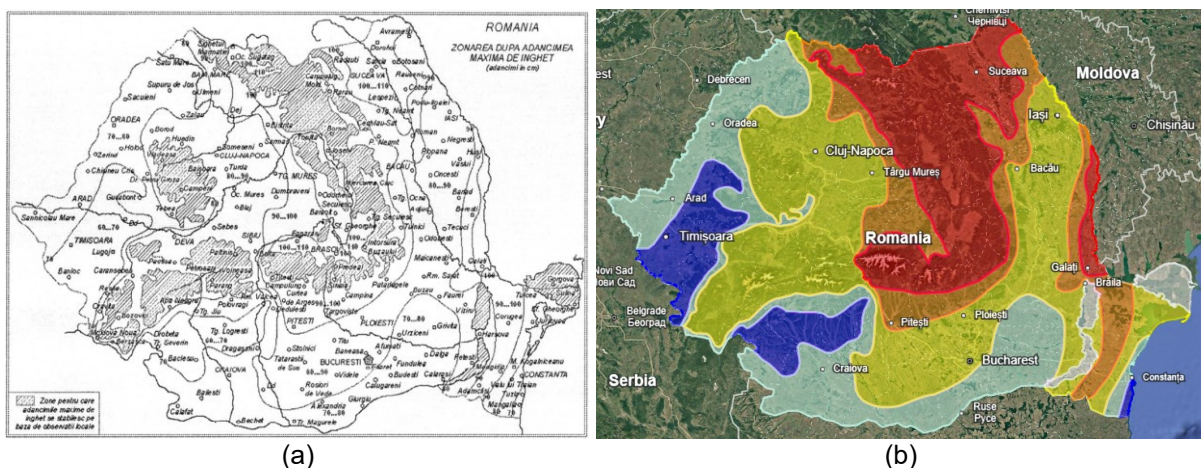


Fig. 4 - Maximum frost depth zonation map - (a) original, (b) constructed in GIS [8]

2.4 Online altitude query

In addition to zonation layers derived from national maps, the toolchain uses site altitude as an auxiliary parameter whenever it is required by the normative workflow

(correction of the ground snow-load characteristic value or the dynamic wind pressure reference value) or by the project's internal ruleset (skipping the frost depth map interrogation for altitudes greater than 1000m). Altitude can be supplied explicitly by the user, but when it is not provided, the toolchain performs an online lookup as an automatic fallback implemented through a lightweight HTTP query to the Open-Elevation endpoint [9].

2.5 Uniform administrative level derived maps

Starting from the MDLPA-hosted official interactive seismic zonation reference [10], additional administrative level derivate maps were created for each environmental factor layer (Fig. 5, Fig. 6, Fig. 7), ensuring consistent parameter attribution when results must be reported or audited per administrative unit rather than per point coordinate. The maps were created by merging polygon attributes from the original (updated) maps with an administrative limit map [11], keeping the maximum value across the whole administration.

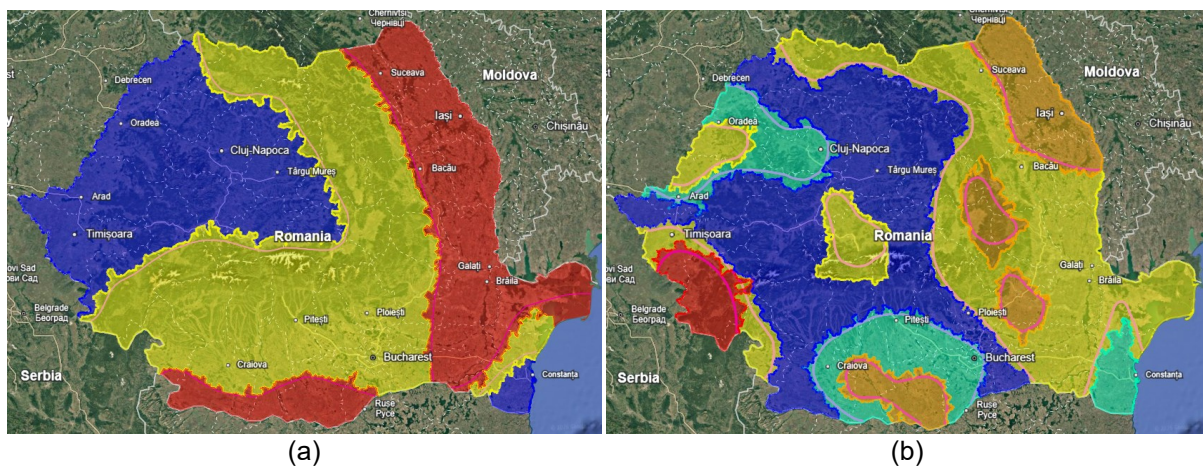


Fig. 5 - Administrative level ground snow-load characteristic values (a) and dynamic wind pressure reference values (b) zonation maps

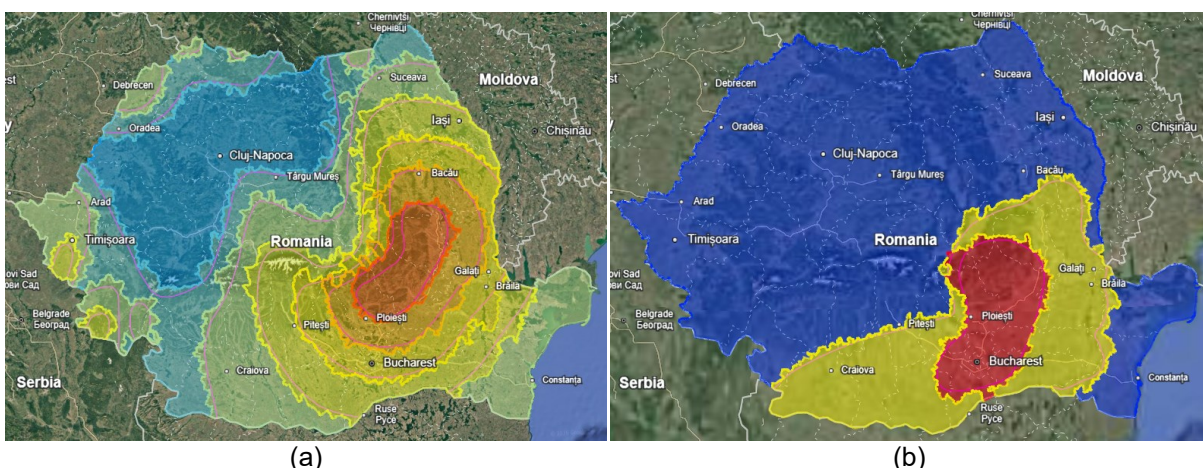


Fig. 6 - Administrative level peak ground acceleration (a) and response spectra control period (b) zonation maps

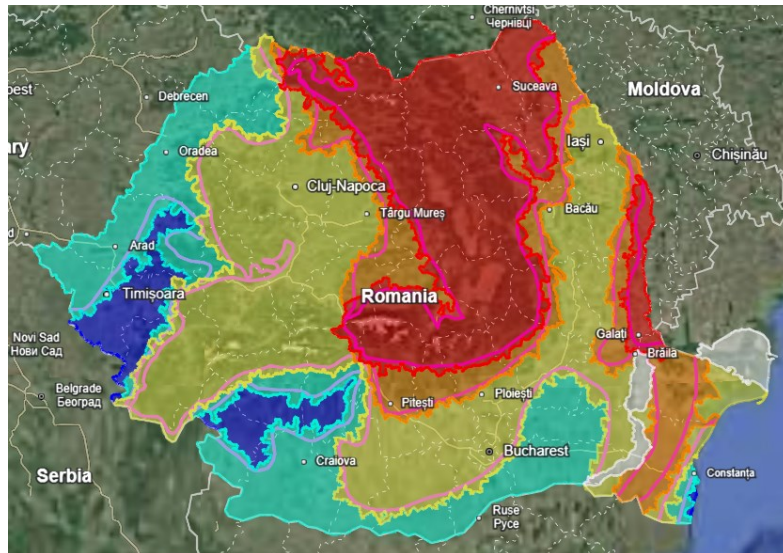


Fig. 7 - Administrative level maximum frost depth zonation map

These administrative level layers are used in the toolchain as an alternative lookup mode (alongside point-based zonation), enabling consistent aggregation, batch processing, and traceable reporting aligned with administrative workflows.

2.6 Multi-factor overlap outputs

To support transparent verification of zone attribution across multiple environmental factors, two composite maps were generated and used throughout the workflow: a point-based factor overlay map and an administrative level factor overlay map (Fig. 8). These composite maps are not just visualization layers - they are functional artifacts used by the Python toolchain to retrieve the primary zonation parameters in a single operation. Specifically, the runner performs a single interrogation of these maps, depending on the selected mode, and obtains all values (snow, wind, seismic and frost depth) from the containing polygon's ExtendedData attributes.

While the composite overlays provide the primary normative parameters, the pipeline still queries population density and tree cover density as separate auxiliary layers to compute derived modifiers. This separation keeps the composites focused on normative zonation outputs while allowing classification rules to evolve independently.

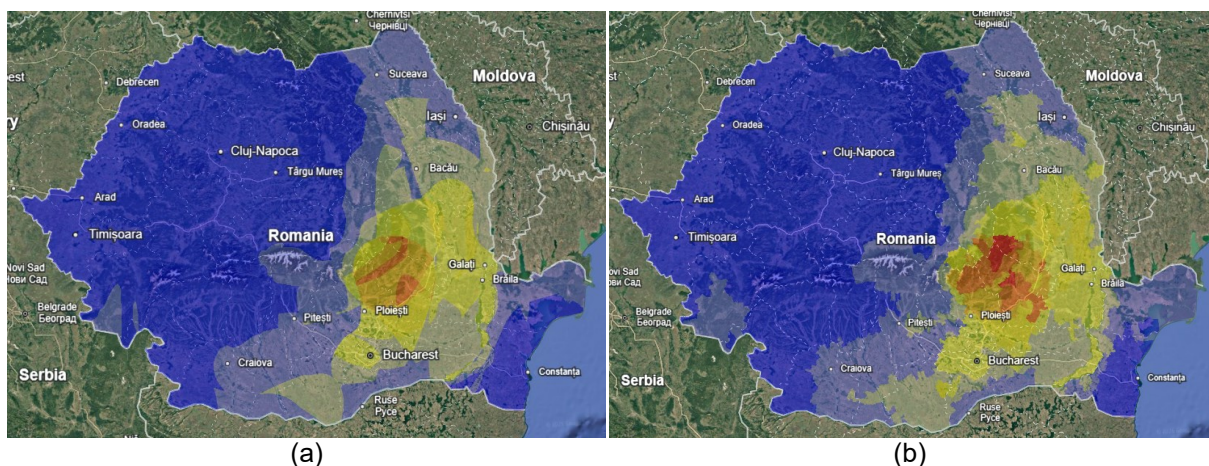


Fig. 8 - Point-based factor overlay (a) and administrative level factor overlay (b) maps

3. Python toolchain architecture and workflow

3.1 Design goals and guiding principles

The Python toolchain is designed to retrieve location-dependent environmental parameters from GIS-exported KML layers with minimal dependencies, compute code-based loads and coefficients deterministically and produce traceable, reproducible outputs suitable for single-point use and batch workflows (Fig. 9).

Two implementation principles are central:

- a. Single-pass retrieval for primary zonation parameters: snow (sk), wind (qb), seismic (ag, Tc), and frost depth are retrieved via a single composite KML interrogation for either the point-based factor overlay map or the administrative level factor overlay map, minimizing parsing overhead and ensuring consistent polygon assignment across factors.
- b. Separate auxiliary layers for classification modifiers: population density and tree cover density are queried independently and used only for classification into snow exposure factor Ce and wind terrain category, keeping composites “normative-only” and allowing classification rules to evolve independently.

3.2 Module responsibilities

The codebase is organized under the “functions” folder and orchestrated via a runner entry point. Each component has a narrow responsibility:

- functions/cli.py: command-line interface, batch execution from CSV/YAML, JSON output, flat CSV export, optional DOCX report creation.
- functions/runner.py - central orchestrator: altitude acquisition, composite map interrogation, auxiliary lookups (population / tree coverage), derivations (Ce, terrain, altitude adjustments), and dispatch to snow / wind modules.
- functions/kmlio.py: dependency-free KML parsing, ExtendedData extraction, polygon and multi-geometry support (including holes), point-in-polygon tests, and “layer-aware” folder selection.
- functions/classify.py: Ce and terrain classification rules, Romania bounds utility, and altitude adjustment of sk.
- functions/snow.py: computation of snow load cases as well as local effects, according to CR 1-1-3/2012.
- functions/wind.py: computation of wind load cases / coefficients for walls, flat / mono / duo / hip roofs, cylindrical / dome dispatch, canopies and isolated elements, internal pressure handling, and case packaging.
- functions/report.py: optional reporting layer: KML→PNG rendering via matplotlib, and DOCX report creation via python-docx.
- run_loads.py: minimal entry point to the CLI.

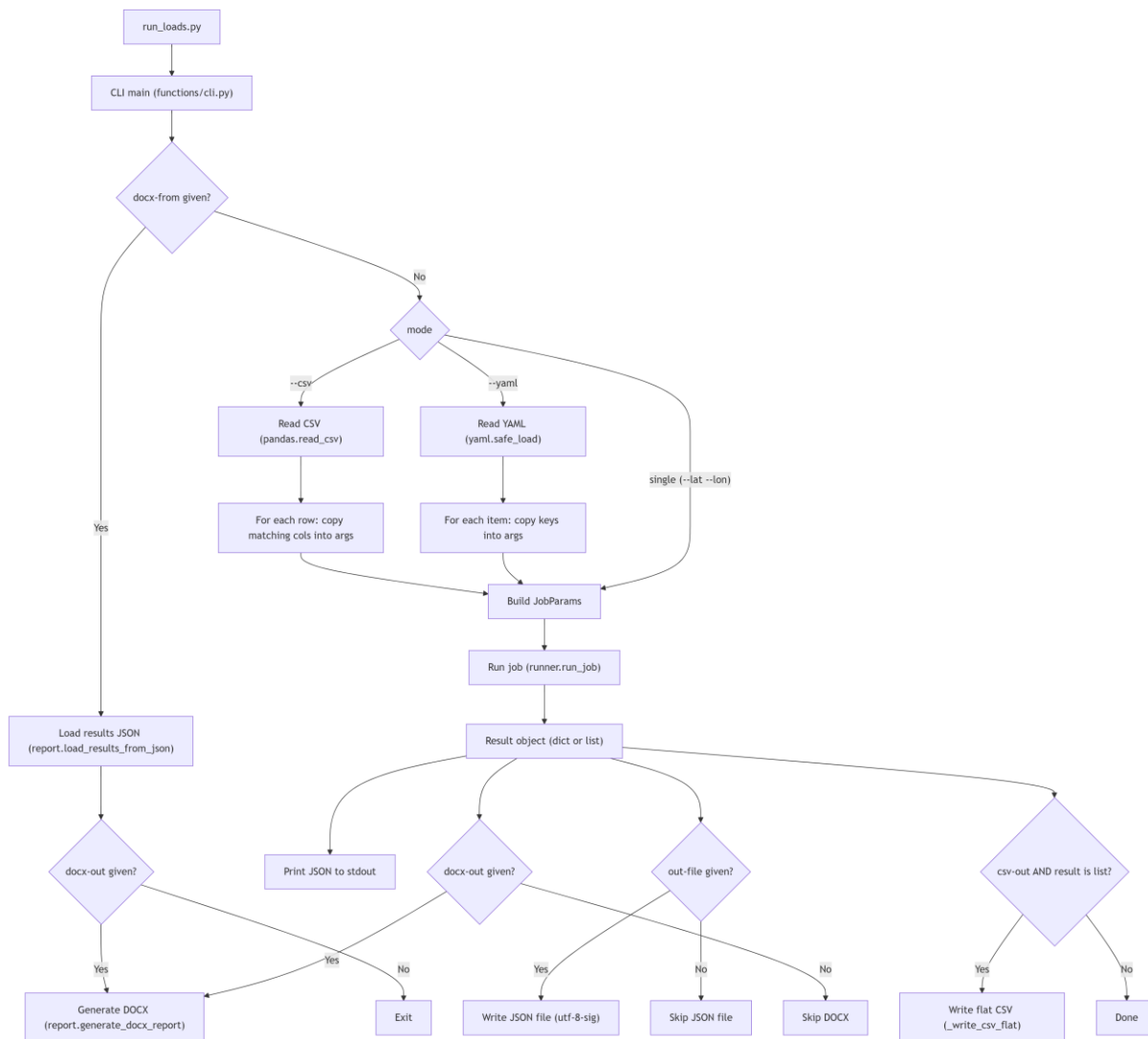


Fig. 9 - Python toolchain workflow

3.3 Input model and interfaces

The main runner uses a JobParams dataclass that captures location, roof typology, geometry, load coefficients, internal pressure options, and map mode selection. Important fields include:

- latitude, longitude (required), altitude (optional override), kml_dir (map directory).
- roof type enumerations including canopy-snow and canopy-open, explicitly differentiated. Canopy-snow defines an awning attached to a building and it is an exceptional case included in CR 1-1-3/2012 for snow computation and canopy-open defines an isolated self-supporting canopy and it is a current case included in CR 1-1-4/2012 for wind computation.
- building geometric dimensions b (width), l (length), h (height), and curved-roof rise f (optional), plus parapet height hp (optional).
- coefficients: gamma_ls, Ct (snow), gamma_lw (wind).
- wind internal pressure controls: dominant_ratio, cpe_at_openings, cpi_override.

- isolated canopy element inputs: `canopy_type`, `phi_obstruction`, `has_corner`.

3.4 CLI execution and batch inputs

The CLI supports both single-run and batch modes and provides several outputs:

- single location: `--lat` and `--lon`,
- batch: `--csv` or `--yaml`,
- outputs: JSON to stdout (`--out`) and/or JSON file (`--out-file`), plus optional flat CSV (`--csv-out`) and DOCX reporting (`--docx-out` or `--docx-from`).

Batch mode preserves traceability, so if a CSV/YAML contains name, it is copied into `meta.name` in outputs. The flat CSV exporter flattens nested dictionaries into dotted-key columns and writes Excel-friendly UTF-8 with BOM.

3.5. Lightweight KML engine (parsing + spatial querying)

3.5.1 KML layer catalogue

The runner (`runner.py`) maintains a KMLS dictionary defining per-layer file name, folder/document name and list of candidate fields to extract. Configured layers include snow (`CR113.kml`), wind (`CR114.kml`), frost (`FRZ_depth.kml`), seismic ag (`PGA_P100-2013.kml`), seismic Tc (`TC_P100-2013.kml`), and auxiliary population / tree layers (`pops.kml`, `TCD_2023_RO_color.kml`).

3.5.2 Point-based vs administrative level map mode switching

The toolchain can switch between point-based and administrative level KMLs via `--use_uat_maps`. For the normative layers {snow, wind, frost, ag, TC}, the file name is rewritten to `*_UAT.kml` when administration level mode is enabled. Population and tree coverage layers remain unchanged. This implements the paper's dual reporting concept: coordinate-based queries and administrative-level aggregation queries using consistent interfaces.

3.5.3 Parsing and layer scoping

`kml.io.py` parses KML using only Python's standard library (`xml.etree.ElementTree`). It supports selecting placemarks from either the entire file, or a specific layer (Folder/Document name match).

3.5.4 Geometry support: polygons, multi-geometries, and holes

The engine extracts polygons from both direct Polygon elements and MultiGeometry containers. Inner boundaries are treated as holes, represented as a compound object: `{"outer": ring, "holes": [ring...]}`. Containment uses ray casting (`point_in_ring`) and a hole-aware predicate (`point_in_polygon`) that returns inside outer AND not inside holes.

3.5.5 Attribute extraction robustness (*ExtendedData*)

Attributes are read from `ExtendedData/Data/value`, and `ExtendedData/SchemaData/SimpleData` with keys normalized (lowercase). The parser attempts numeric conversion with comma-to-dot handling, preserves frost bin strings like "90...100", and can fall back to parsing numeric tokens from placemark description or name if needed.

3.6. Runner orchestration - end-to-end dataflow

3.6.1 Step 1 - *Altitude acquisition (override - online - fallback)*

If altitude is not provided, the runner queries an elevation endpoint and returns 0.00m on any exception. This ensures the pipeline remains functional even without network availability.

3.6.2 Step 2 - *Single composite interrogation for normative parameters*

The runner selects either Full_map.kml (point mode) or Full_map_UAT.kml (administrative level mode), performs a single lookup (lookup_props) to retrieve all key normative values from ExtendedData and maps the specific fields (lowercased) corresponding to snow (CR 1-1-3) - s_k , wind (CR 1-1-4) - q_b , seismic (P100-1/2013) - a_g , T_c and frost (STAS) - frost depth. These are placed into a standardized layers dictionary.

3.6.3 Step 3 - *Auxiliary lookups for modifiers (population + trees)*

Population density and tree cover density are queried via read_all_layers restricted to keys ["pop","trees"]. These values are appended to the same layers output structure, but remain conceptually separate from the normative composite.

3.6.4 Step 4 - *Frost depth rule application*

The runner applies a project rule: if altitude ≥ 1000 m, frost depth is replaced by a flag string "ALT > 1000", otherwise the composite-derived frost depth is used.

3.6.5 Step 5 - *Derivations: Ce, terrain, altitude adjustments*

Before load computation, the runner derives snow exposure label and factor C_e via classify_snow_exposure(pop, trees), wind terrain category via classify_terrain(pop, trees), corrected snow zonation value s_k _corrected via adjust_sk_for_altitude(s_k , altitude) (if numeric) and corrected wind reference pressure q_b _corrected, if numeric, via an altitude corrector c_z . These are stored under derived alongside the raw zonation values and used altitude.

3.6.6 Step 6 - *Geometry-derived parameters for computation*

The runner computes the effective height for wind z_e , parapet ratio h_p/h when parapet height is provided, curved-roof ratios f/d and h/d for cylindrical / dome logic using $d=b$. These values are included in derived for auditability.

3.6.7 Step 7 - *Dispatch to computation modules (snow and wind)*

The runner maps the CLI roof to computation-specific roof type enums, builds SnowInput and WindInput, and executes compute_snow(snow_in) and compute_wind(wind_in), storing outputs under snow and wind respectively - see 0, 0 for description of computation modules.

3.6.8 Output packaging

run_job() returns:

- input: all user inputs (serialized),
- kml: retrieved layer values,
- derived: computed modifiers and derived geometry ratios,
- snow case outputs: direct fields (μ , s , etc.) or a list of named cases with μ and s values, or nested dicts for special configurations and in many cases include derived local-effect values,

- wind case outputs: qpe, qpi and a list of dicts containing zone, cpe, w_ext, w_net, and cpnet flag.

This “self-describing result object” supports reproducibility and enables batch post-processing without re-running retrieval or calculations.

An optional reporting layer is provided in report.py which can generate DOCX outputs and render KML polygons to PNG with zone highlighting and a position marker, including a legend derived from KML styles and placemark descriptions. This layer is triggered through CLI flags and does not affect computational correctness.

3.7. Snow computation module (functions/snow.py)

3.7.1 Input contract (SnowInput) and general formula

Snow computations are driven by the SnowInput dataclass, which includes zonation value sk, factors gamma_ls, Ce, Ct, roof type selector (e.g., single, double, multi, cyl, and several special/exceptional variants), geometry parameters (slopes, spans, heights) used by specific cases. The output is a structured dictionary whose content depends on roof type but preserves governing μ coefficients and resulting roof snow action values.

3.7.2 Roof-type dispatcher (compute_snow)

compute_snow() acts as a dispatcher and returns a dict with a stable type field and case payload. Implemented roof families include:

- Single pitch (single): computes μ via mu1_of_alpha and returns {type:"single", mu, s}.
- Double pitch (double): returns multiple distributions including drift variants, packaged as a list of named cases with μ and s values.
- Multi-span (multi): provides “non-drift” plus valley drift case (μ_2) and the governing s_max.
- Cylindrical (cyl): computes μ_3 as function of geometry, returns half-span and max values.
- Adjacent/obstacle/parapet and exceptional families: provide additional case logic packaged as nested dicts (e.g., {"type":"exceptional_canopy","exc":{"...}}).

The runner maps CLI roof types into snow module types (e.g., canopy-snow → exceptional_canopy, hip → double).

3.7.3 Embedded local effects

A distinctive part of the snow module is the systematic attachment of a local effects bundle under locals.hanging, computed from the governing s_max across the case. This bundle includes s_max, se (local effect result), k and d intermediate quantities and is embedded into the snow output dictionary. Architecturally, this means downstream reporting can include not only global roof loads but also a consistent computed local-effects artifact without recomputation.

3.8. Wind computation module (functions/wind.py)

3.8.1 Input contract (*WindInput*) and pressure evaluation structure

Wind computations are driven by *WindInput*, which includes *qb*, terrain category *terr*, factors *gamma_lw*, heights *z_e* and optional *z_i* (defaults to *z_e*), geometry *h*, *b*, *l*, roof type selector (*walls*, *flat*, *mono*, *duo*, *hip*, *cyl*, *dome*, *canopy*, *iso_wall*), internal pressure options (*dominant_ratio*, *cpe_at_openings*, *cpi_override*), canopy/isolated parameters (*phi_obstruction*, *has_corner*). The module computes $q_{pe} = q_p(z_e, qb, terr)$ and $q_{pi} = q_p(z_i, qb, terr)$, then enumerates zonal cases and returns a list of case dictionaries.

3.8.2 Internal pressure handling and resulting net pressure envelope

Internal pressure is handled in a controlled precedence order:

1. explicit override if *cpi_override* is given,
2. dominant-opening case if both *dominant_ratio* and *cpe_at_openings* are given,
3. otherwise a default non-dominant set (positive and negative).

For each external zone coefficient (*cpe*), the module computes external pressure *w_ext* and net pressure *w_net* by combining external and internal pressures (selecting the governing magnitude), unless a case is of type *cpnet*, where internal pressure is intentionally excluded. Outputs preserve zone label, coefficient, external pressure, net pressure, and whether the case is *cpnet*.

3.8.3 Roof / structure dispatch: implemented typologies

The wind module implements multiple dispatch branches that correspond to typical and special cases used in practice:

- Walls: zones A–E via interpolation keyed on *h/l*.
- Flat roofs: zones F–H and I (+/–), with parapet-dependent selection when *hp/h* matches configured values.
- Mono/Duo/Hip roofs: coefficient tables interpolated by roof angle; evaluated for two principal wind directions.
- Cylindrical roofs: uses *f/d* and *h/d* (computed in runner). If $f/d < 0.05$, the module dispatches to flat roof logic and records a dispatch note, otherwise it computes cylindrical zones A (possibly two branches), B, and C.
- Domes: similarly uses *f/d* and *h/d* and dispatches to flat roof if $f/d < 0.05$; otherwise computes dome zones A, B, C using helper functions.
- Canopies: returns *cp,net* cases for one-slope and/or two-slope canopies depending on *canopy_type*, with obstruction factor ϕ used to interpolate minimum branches.
- Isolated walls: returns *cp,net* based on *l/h* and corner presence; enforces $\phi \geq 0.8$ for validity in the implemented logic.

The toolchain includes explicit helper logic for curved roofs: cylindrical roof *cpe,10* functions and plot helpers exist under *cyl_roof_cpe.py*, and dome roof *cpe,10* functions and plot helpers exist under *dome_roof_cpe.py*. These helpers implement piecewise-linear reconstructions to match the normative diagrams as used by the toolchain and include plotting utilities used for verification/documentation.

4. Case studies and experimental setup

4.1. Case study A - Zonation parameters

4.1.1 Scenario description

To demonstrate the toolchain's batch capability and the consistency of the administrative level reporting workflow, a set of 50 randomly generated locations within Romania was evaluated (Fig. 10). These 50 locations span 32 counties, providing broad national coverage for checking that the composite map assignment is stable across diverse regions. Each location was assigned an automatically generated descriptive label ("County, near Locality") and processed using administrative-level composite overlays (use_uat_maps = true).



Fig. 10 - Case study A - Randomly generated locations

The batch was intentionally standardized to isolate the influence of zonation parameters and classification modifiers from geometric variability. Therefore, all 50 locations used the same canopy geometry and analysis settings:

- roof scenario: canopy-open (wind canopy case),
- canopy type: single-slope canopy (canopy_type = "single"),
- obstruction coefficient: $\phi = 0.5$,
- geometry: $b = 12$ m, $l = 30$ m, $h = 6$ m, $f = 0$ m, parapet not used,
- coefficients: $\gamma_{ls} = 1.0$, $C_t = 1.0$, $\gamma_{lw} = 1.0$, $c_d = 1.0$,
- roof slope input: $\alpha = 25^\circ$.

This controlled setup ensures that the observed variability in computed loads is driven primarily by spatially retrieved normative values (sk, qb, ag, Tc, frost), altitude-dependent corrections (sk_corrected, qb_corrected) and rule-based classification modifiers (Ce, terrain category).

4.1.2 Results - zonation parameters

All 50 locations yielded complete retrieval of the core normative parameters from the administrative level composite overlay: snow (sk), wind (qb), seismic (ag, Tc), and frost depth (or an altitude flag when applicable). The following ranges were observed across the batch:

- a. Altitude ranged from -2 m to 2043 m (median 272 m, mean 413 m).
- b. Snow zonation (ground snow load)
 - base zonation value sk: 1.5–2.5 kPa (median 2.0 kPa)
 - corrected value sk_corrected: 1.5–5.46 kPa (max occurs at high altitude)

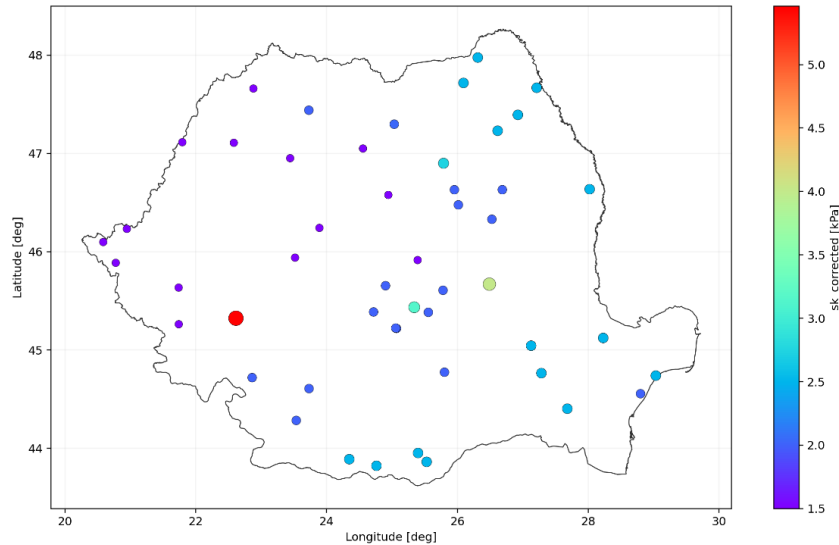


Fig. 11 - Case study A - Zonation gradient for ground snow load

- c. Wind zonation (reference dynamic pressure)
 - base qb: 0.4–0.7 kPa (median 0.6 kPa):
 - corrected value qb_corrected: 0.4–1.06752 kPa (high-altitude correction visible in extremes).

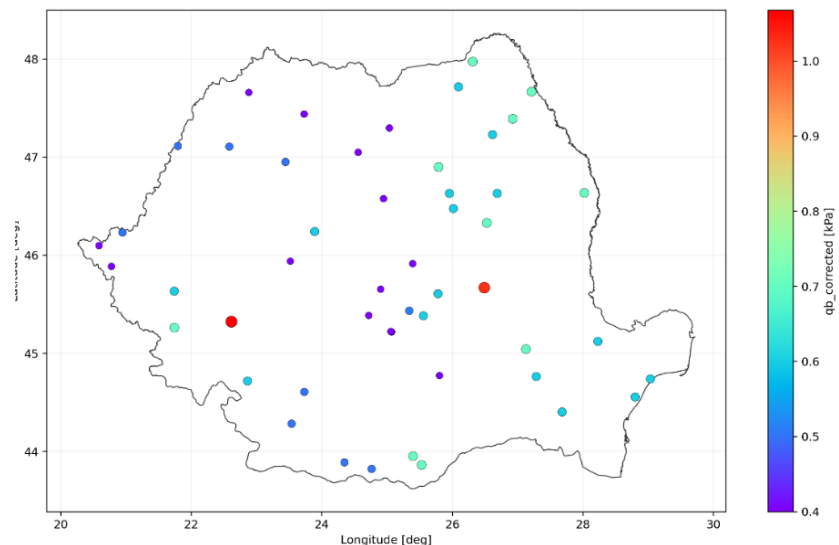


Fig. 12 - Case study A - Zonation gradient for reference dynamic wind pressure

- d. Seismic parameters:
 - ag: 0.1–0.4 g (median 0.2 g)
 - Tc: predominantly 0.7 s (median 0.7 s), with maxima up to 1.6 s
- e. Frost depth is reported either as a bin (e.g., “70...80 cm”) or flagged when altitude exceeds 1000 m:
 - frost depth mid-bin values span 60 - 110 cm (excluding altitude-flagged sites)
 - 6 / 50 sites (12%) were flagged for altitude (map interrogation bypassed per ruleset).

4.1.3 Results - derived classification outputs (Ce and terrain) and their distribution

Using population density and tree cover density, the toolchain computed snow exposure and wind terrain category. Across the batch:

- a. Snow exposure label: “Normal” - 38/50 (76%), “Complete” - 12/50 (24%)
- b. Terrain category: I - 9/50 (18%), II - 29/50 (58%), III - 12/50 (24%)

A notable practical aspect is that population density values were missing in 26/50 cases, and tree density values were missing in 6/50 cases in the exported batch dataset; This means that the values are 0 in these cases, so the pipeline still produced Ce/terrain outputs for all sites.

4.1.4 Results - computed snow loads for the standardized roof configuration

Although the roof scenario is “canopy-open” (wind-focused), the toolchain still computes the snow action for the standardized slope and factors as part of the overall location characterization.

For the batch, computed roof snow load s (single-pitch case with $\mu = 0.8$, $\gamma_{ls} = 1.0$, $C_t = 1.0$, and C_e derived per site) ranged from 0.96 kPa (lowest) to 4.368 kPa (highest) with median 1.6 kPa and mean 1.646 kPa.

The local-effect bundle (“hanging snow”, §6.2) was produced for every case. The derived s_e ranged from approximately 0.295 kN/m to 13.104 kN/m, with higher values associated with high-altitude sites where the corrected snow load is largest.

4.1.5 Results - computed wind pressures for single-slope canopy (cp,net)

For each site, the wind module computed q_{pe} and q_{pi} and returned a fixed set of 8 canopy cases (“A/B/C/MAX” × “max desc / min asc”). Because the canopy formulation is treated as cp,net, internal pressure is not combined and cpnet = True throughout.

Across the 50-site batch, q_{pe} ranged from 0.286 to 0.920 kPa (median ~0.466 kPa, mean ~0.471 kPa). The maximum absolute net pressure across zones per site ($|w_net|$ envelope) ranged from 0.888 to 2.851 kPa, median 1.443 kPa, mean 1.459 kPa.

4.2. Case study B - Construction parameters

4.2.1 Scenario description

To demonstrate computation breadth across roof typologies and parameter branches, a second batch used 15 selected locations (covering lowland, coastal, and mountainous zones) and executed 33 scenarios, producing 495 runs (Fig. 13).

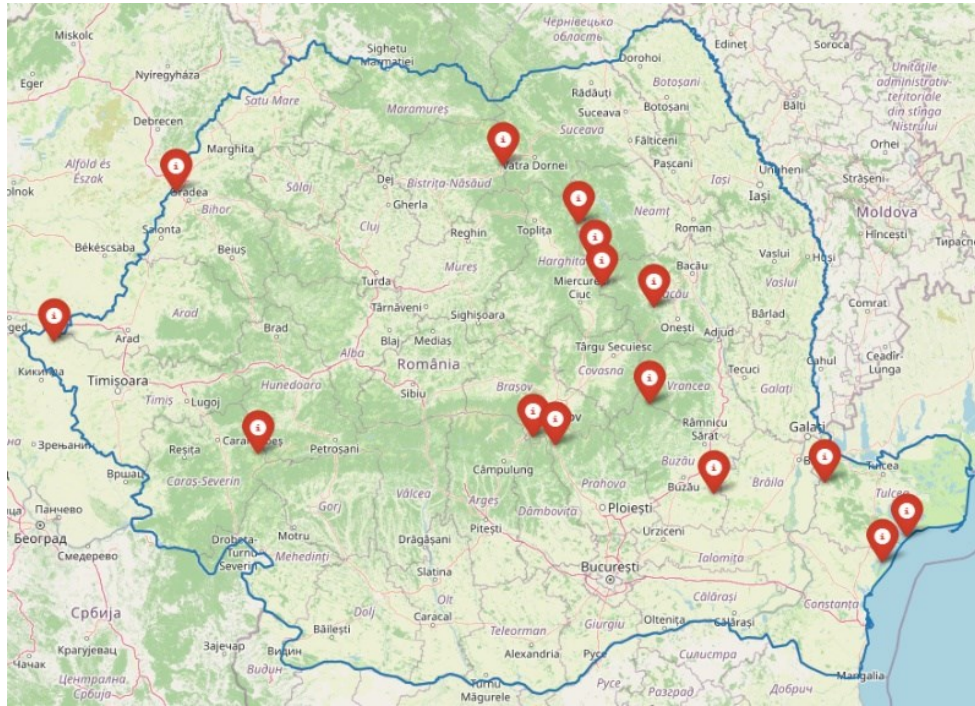


Fig. 13 - Case study B - Selected locations

The following scenario matrix was chosen to activate all major roof dispatch branches implemented in the wind and snow modules:

- Roof / typology coverage: canopy-open, canopy-snow, single, double, hip, multi, flat, cyl, and dome.
- Roof angles: α_1 values used were 0° , 10° , 25° , 30° , 40° ; α_2 values used were 10° (where relevant).
- Canopies: canopy_type $\in$ {single, double, both}, and obstruction factor $\phi \in$ {0.0, 0.5, 1.0} (explicit sensitivity sweep).
- Geometry variants: $b \in$ {12, 20} m, $l \in$ {30, 60} m, $h \in$ {6, 10, 15} m, with curved-roof rise $f \in$ {0, 0.5, 2.0} m and parapet $h_p \in$ {0.25, 1.0} m where applicable.
- Internal pressure branches: some scenarios include dominant openings (dominant_ratio values) and explicit cpi overrides, exercising the wind internal-pressure decision paths.

4.2.2 Results - zonation parameters

All 495 runs include complete retrieval of the site normative parameters (s_k , q_b , a_g , T_c , frost depth) via the composite overlay. These values are identical for a given location across all scenarios, and only the computed loads/envelopes change with construction inputs.

4.2.3 Results - snow outputs (roof actions and local effects)

Even when the focus is wind breadth, the pipeline computes snow actions consistently for each scenario/roof type and also reports the attached local effects (Fig. 14). The largest computed roof snow load in the breadth suite is $s = 4.368$ kPa, corresponding to the maximum corrected snow zonation in the dataset ($sk_corrected = 5.46$ kPa) at the high-altitude site Caraş-Severin, near Complex Muntele Mic. The corresponding local effect result reaches $s_e = 13.104$ kN/m in the same extreme case.

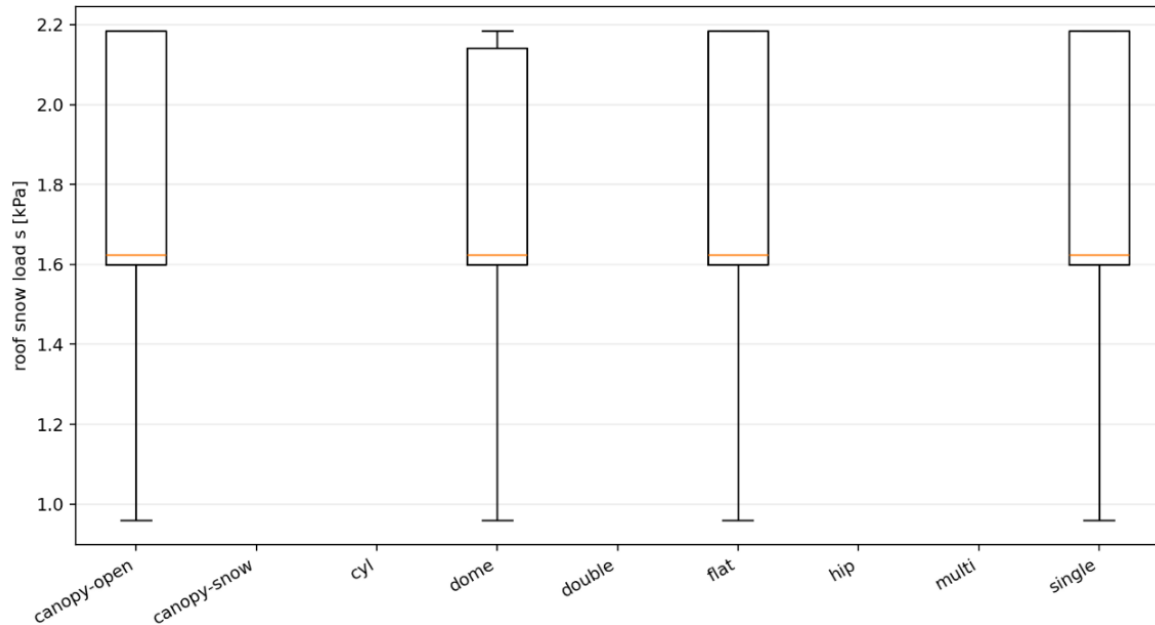


Fig. 14 - Case study B - Roof snow load envelope by roof type

4.2.4 Results - wind outputs (envelopes and sensitivity)

Across the breadth suite, wind outputs vary by typology and scenario parameters (Fig. 15, Fig. 16):

- The maximum absolute net pressure envelope $|w_net|$ across wind zones spans 0.339 to 2.943 kPa over all 495 runs.
- In canopy scenarios (cp,net), the obstruction factor ϕ and canopy slope influence the envelope as expected; the dataset-level correlation between ϕ and $|w_net|$ is ~ 0.33 (moderate), reflecting that ϕ acts in specific branches (not uniformly in all zones).

Illustrative extremes (wind):

- The largest envelope observed in the suite is $|w_net| = 2.943$ kPa, occurring for Vrancea, near Secuiu under scenario canopy_single_phi1_a30 ($\alpha=30^\circ$, $\phi=1.0$).
- The next highest is $|w_net| = 2.879$ kPa for Caraş-Severin, near Complex Muntele Mic under the same scenario family (high altitude $\rightarrow$ high $qb_corrected$).

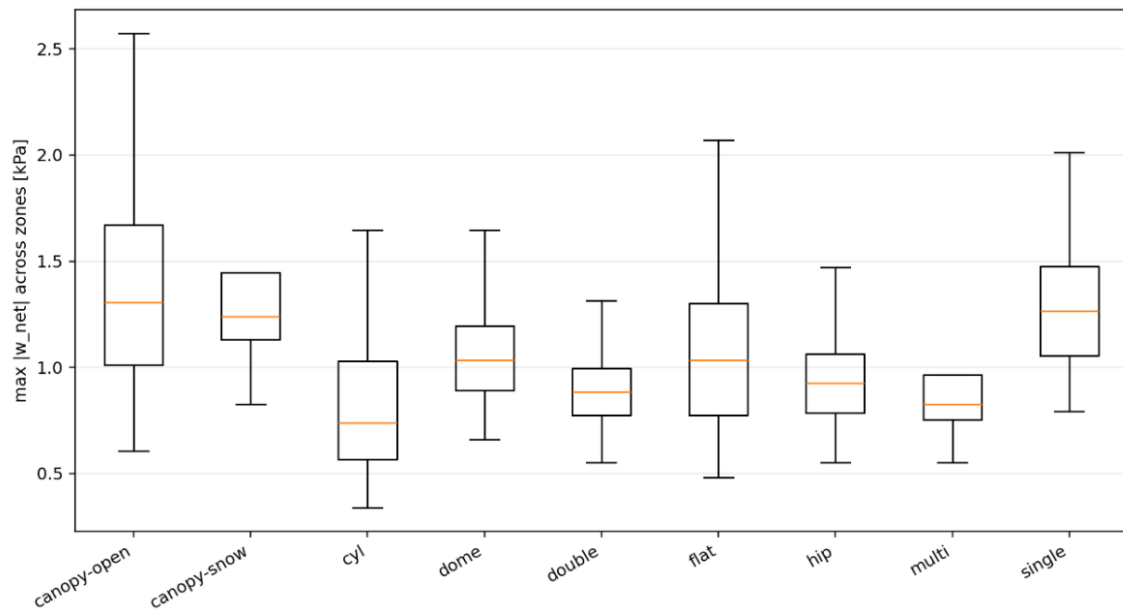


Fig. 15 - Case study B - Wind pressure envelope by roof type

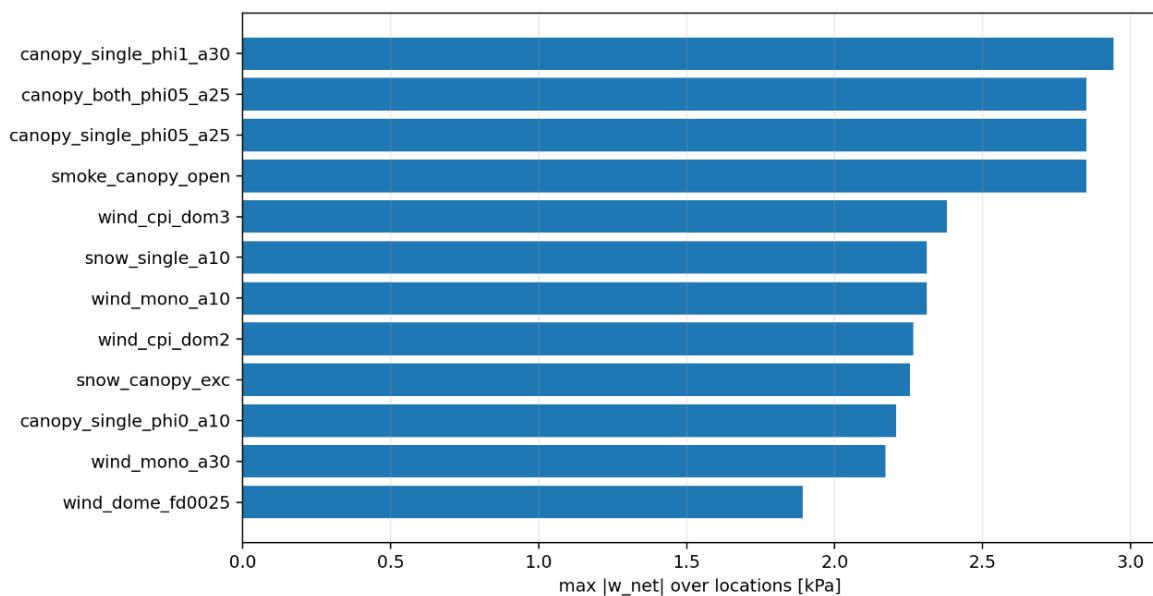


Fig. 16 - Case study B - Wind pressure scenario leaderboard (top 12)

4.3. Validation, Quality Assurance / Quality Control

4.3.1 Traceability of assignments

Each result object stores inputs, per-layer KML values including placemark identifiers, derived modifiers (Ce, terrain, altitude-used, corrections), and computed outputs (snow + wind cases). This “self-describing” output structure supports point-level auditing and scenario comparisons without rerunning the pipeline.

4.3.2 Cross-format consistency

In the 50-location run, results were also emitted as a DOCX report, and spot-checking shows consistent parameter values and computed outputs between JSON and the

generated narrative blocks (e.g., Olt–Obârșia: $s_k=2.5$ kPa, $q_p=0.5$ kPa, $a_g=0.2$ g, $T_C=1.0$ s, frost depth=70...80 cm; computed canopy cases and snow outputs match across formats).

4.3.3 Guardrails and explicit non-support cases

For certain geometries where the adopted normative basis does not provide a defensible snow μ (e.g. domes), the toolchain uses explicit guardrails (flagging as warning). This is treated as QA/QC as it prevents silent generation of ungrounded coefficients.

5. Conclusions

This work demonstrates a GIS-assisted Python toolchain that retrieves location-dependent Romanian environmental parameters from composite KML overlays and computes snow and wind loads across different building geometries.

Case study A demonstrates spatial consistency and map-position robustness under a controlled single-geometry setup. Case study B complements this by showing that, at the same locations, the toolchain produces substantially different load case sets and envelopes when switching between roof families (mono / duo / hip / flat / cyl / dome / canopy) and scenario parameters (ϕ , parapets, curvature regime, internal pressure). Together, the two case studies separate and validate the toolchain's two main claims: reliable site characterization and broad code-based computation coverage.

The batch results highlight the value of the toolchain for practice-oriented workflows, showing that there is a large variability across Romania even for a standardized structural scenario, meaning “one-size-fits-all” assumptions can lead to unsafe or overly conservative designs and that administrative level outputs facilitate uniform administrative reporting (useful for portfolios, regional inventories, and standardized documentation), while still preserving traceability via placemark identifiers.

6. Data availability

The full codebase, including composite overlay maps and auxiliary layers (population and tree cover density) and the case study data are available on demand from the authors.

7. Funding

This research was funded by the Technical University of Civil Engineering Bucharest (Grant no. *GnaC^{ARUT}₂₀₂₄ – UTCB – 34*).

References

- [1] CCERS-UTCB. *Seismic Risk Assessment Research Center – Zonation Maps*. Available online: <http://ccers.utcb.ro/index.php/utile> (accessed on March 2026).
- [2] UTCB, Cod de proiectare. Evaluarea acțiunii zăpezii asupra construcțiilor, indicativ CR 1-1-3/2012.
- [3] UTCB, Cod de proiectare. Evaluarea acțiunii vântului asupra construcțiilor, indicativ CR 1-1-4/2012.
- [4] UTCB, Cod de proiectare seismică - Partea I - Prevederi de proiectare pentru clădiri, indicativ P100-1/2013.

- [5] Kontur. *Romania: Population Density for 400m H3 Hexagons (2023)*. Humanitarian Data Exchange (HDX). Available online: <https://data.humdata.org> (accessed on March 2026).
- [6] M. Hansen, P. Potapov, R. Moore, M. Hancher, S. Turubanova, A. Tyukavina, D. Thau, S. Stehman, S. Goetz, T. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. Justice and J. Townshend, "High-Resolution Global Maps of 21st-Century Forest Cover Change," *Science*, vol. 342, no. 6160, pp. 850-853, 2013.
- [7] European Union's Copernicus Land Monitoring Service information, "Tree Cover Density 2023 (raster 100 m), Europe, yearly," Available online: <https://land.copernicus.eu/en/map-viewer> (accessed on March 2026).
- [8] ***, "STAS 6054-77 - Teren de fundare - Adâncimi maxime de îngheț - Zonarea teritoriului României".
- [9] Open-Elevation. *Open Elevation API (public service)*. Available online: <https://www.open-elevation.com/> (accessed on March 2026).
- [10] Ministerul Dezvoltării, Lucrărilor Publice și Administrației, "Hartă interactivă cu valorile de vârf a accelerației terenului pentru proiectare la cutremur la nivelul României," Available online: <https://www.mdlpa.ro/subarticles/8/hartariscseismic> (accessed on March 2026).
- [11] D. Runfola, A. Anderson, H. Baier, M. Crittenden, E. Dowker, S. Fuhrig and et. al, "geoBoundaries: A global database of political administrative boundaries," *PLoS ONE*, vol. 15, no. 4, p. e0231866, 2020.