The Cost of Neglect

A State-Dependent Framework for Quantifying the Flood Risk of Historic Masonry Buildings

Bart Koppejan

Delft University of Technology

The Cost of Neglect

A State-Dependent Framework for Quantifying the Flood Risk of Historic Masonry Buildings

by

Bart Koppejan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, June 24, 2026 at 15:00

Student number:	5218160	
Project duration:	Oktober 2025 – June 2026	
Thesis committee:	Dr. S. (Simona) Bianchi	TU Delft, First Supervisor
	Dr. B.A. (Barbara) Lubelli	TU Delft, Second Supervisor
	Dr. N.M.J.D. (Nico) Tillie	TU Delft, Committee Member

Cover: Photo by collectie I.M. van Loo, Gemeentearchief Reimerswaal, Watersnoodmuseum Ouwerkerk.

TU Delft

Acknowledgements

I would like to express my deepest gratitude to my primary supervisor, Simona Bianchi for being so generous with her time and her valuable feedback. I am grateful for how you pushed me to find a topic that I was really interested in myself. I would also like to express my appreciation to my secondary supervisor, Barbara Lubelli for her thorough knowledge on masonry damages, her interest in this project, and her constructive feedback. Furthermore, I would like to thank my university delegate, Nico Tillie, for being part of my thesis committee and ensuring fair treatment.

This endeavour would not have been possible without Jonathan Ciurlanti and his invaluable insights and opinions from the practical field of flood risk assessment. Also, many thanks to Paul Korswagen and Francesco Messali for meeting with me to discuss the structural mechanics of unreinforced masonry and its failure mechanisms. You have both provided valuable insights and literature references that helped me build this thesis. Thanks also, to my friends and peers who have supported me by listening to me rambling on about this project, brainstorming with me, providing moral support, and the necessary distraction.

I am forever grateful to my family, especially my parents, and Jasmijn for supporting me. You have kept my spirits up and my motivation high during this process. Thank you for always believing in me.

Above all, thanks be to Him.

Abstract

Driven by climate change, the increasing frequency and intensity of flood events impose growing risks on urban infrastructure. To quantify these projected risks, conventional flood risk assessments rely on generic, univariate depth-damage curves that assume a pristine building stock. As a result, these methods systemically underestimate the vulnerability of historic unreinforced masonry dwellings that exhibit pre-existing structural degradation, exposing a critical scale gap between mesoscale risk modelling and microscale structural engineering physics. To bridge this gap, this thesis develops a multivariate, state-dependent probabilistic vulnerability framework that integrates component-level structural capacities with sociotechnical recovery timelines. The framework is applied to two residential case study locations, which reveals that initial degradation shifts damage onset to lower flood depths. Mean repair costs yield a 5 to 8 percentage point divergence in median repair costs. Crucially, crossing into extensive structural damage triggers compounding procurement and permitting delays that severely elevate re-occupancy downtime, extending community displacement from two months to over a year. While component adaptation mitigates direct interior losses, pre-flood structural restoration is more effective in reducing median re-occupancy downtime by up to 40.7%. This framework establishes a scalable tool to improve current flood risk assessments and enable targeted climate adaptation and heritage preservation investments within ageing urban environments.

Reading Guide

This thesis is structured in seven chapters. Chapter 1 introduces the problem statement, objective and motivation, research questions, relevance and scope of the thesis. Next, Chapter 2 describes the research design. It outlines how the theoretical blocks from Chapter 3 are operationalized into a progressive, multi-stage research strategy. Chapter 3 presents the theoretical framework underpinning the research. It reviews the state of the art in flood risk modelling, characterizes the physical demand mechanisms acting on building facades, and derives the mechanics-based parameterization of unreinforced masonry (URM) out-of-plane capacity and its degradation across different conservation states. The chapter concludes by synthesizing the collected knowledge into a unified state-dependent vulnerability modelling methodology.

Chapter 4 presents the case study definition and the architecture of the probabilistic flood risk model. The chapter is structured around the four sequential phases of the simulation engine: archetype definition, stochastic data initialization, vulnerability evaluation, and post-processing into risk metrics. The archetype is defined in terms of geometry, material properties, and component inventory. Chapter 5 then presents the model outputs for the specified case studies across all conservation states. Results cover the hazard demand, archetype vulnerability curves, downtime distributions, the effectiveness of mitigation strategies, and location-specific risk estimates for two selected case study sites.

Based on the data in Chapter 5, Chapter 6 first reflects on the broader implications of state-dependent vulnerability modelling for flood risk management and heritage conservation. Then, it critically discusses modelling choices, assumptions, and limitations, and includes a reflection on directions for future research. Lastly, the research questions are answered. The thesis closes in Chapter 7 with a reflection on the thesis process.

Contents

List of Tables

List of Figures

Nomenclature

Abbreviation	Meaning
CS	Conservation State
DR	Damage Ratio
DS	Damage State
DT	Downtime
EFM	Equivalent Frame Method
EUL	Equivalent Uniform Load
FDA	Flood Damage Assessment
FEM	Finite Element Method
FRA	Flood Risk Assessment
GIS	Geographic Information System
MCS	Monte Carlo Simulation
MRQ	Main Research Question
OOP	Out-of-Plane
SQ	Sub-Question
URM	Unreinforced Masonry

Symbol	Description (Unit)
c_w	Crack width (mm)
C_b	Blockage coefficient
C_depth	Depth coefficient
C_drag	Drag coefficient
C_str	Building structure coefficient
d	Flood depth (m)
d_DS	Depth threshold to damage state DS (m)
dh	Equivalent head due to low velocity flood flows (m)
f_HD	Hydrodynamic line load $(\mathrm{kN/m})$
f_HS	Hydrostatic line load $(\mathrm{kN/m})$
f_k	Characteristic flexural strength $(\mathrm{kN/m^2})$
F_i	Debris impact force (kN)

Symbol	Description (Unit)
$F_{\mathrm{res,DS}}$	Resistance threshold to damage state DS $(\mathrm{kN/m})$
g	Gravitational acceleration $(\mathrm{m/s^2})$
L	Wall height (m)
L_i	Sampled wall height for simulation i (m)
L_{ref}	Height of wall from reference finite element method model (m)
M_{max}	Maximum bending moment (kNm)
P_i	Sampled pre-compressive stress for simulation i (kPa)
P_{ref}	Pre-compressive stress in wall from reference finite element method model (kPa)
$q_{\mathrm{DS,i}}$	Final calculated resistance threshold to damage state DS for simulation i $(\mathrm{kN/m^2})$
q_{lat}	Lateral capacity $(\mathrm{kN/m^2})$
$q_{\mathrm{ref,DS,i}}$	Resistance threshold to damage state DS for simulation i, including material uncertainty $(\mathrm{kN/m^2})$
$q_{\mathrm{res,DS}}$	Resistance threshold to damage state DS $(\mathrm{kN/m^2})$
t	Wall thickness (m)
t_i	Sampled wall thickness for simulation i (m)
t_{ref}	Thickness of wall from reference finite element method model (m)
v	Flood flow velocity $(\mathrm{m/s})$
w_{eq}	Equivalent uniformly distributed load $(\mathrm{kN/m^2})$
w_D	Debris weight (kg)
β_{model}	Dispersion for sampling $\varepsilon_{\mathrm{model,i}}$
γ_i	Geometry scalar for simulation i
γ_M	Material partial safety factor
γ_w	Flood water specific weight $(\mathrm{kN/m^3})$
$\varepsilon_{\mathrm{mat,i}}$	Uncertainty factor for material strength in simulation i
$\varepsilon_{\mathrm{model,i}}$	Uncertainty factor for model simplification in simulation i
λ_{CS}	Capacity reduction factor for conservation state CS
ρ_F	Flood water density $(\mathrm{kg/m^3})$

1 Introduction

The global and regional climate is changing. Increased greenhouse gas concentrations, rising mean temperatures, and an escalation in heavy precipitation and weather extremes are being observed globally (Intergovernmental Panel on Climate Change, 2023). According to recent climate projections, the return periods for extreme hydrological events are decreasing. Floods will occur more frequently and with greater intensity, leading to heightened risks for the built environment (Intergovernmental Panel on Climate Change, 2023). Consequently, the ability to accurately evaluate these risks is becoming increasingly vital for the preservation of human safety and property.

Flood Risk Assessments (FRAs) serve as the primary tool to quantify potential direct and indirect losses resulting from inundation events, mapping risk as a function of hazard, exposure, and vulnerability (Intergovernmental Panel on Climate Change, 2014). The outcomes of these assessments are essential for stakeholder decision-making, policy development, risk mapping, and the planning of localized mitigation and response phases (Aribisala et al., 2022).

Methodologies for executing a Flood Damage Assessment (FDA) are generally divided into empirical and synthetic approaches (Merz et al., 2010). Empirical models rely heavily on post-hazard survey data, which is historically scarce and highly location-specific. Synthetic models bypass this data scarcity by utilizing analytical methods to simulate structural and material responses under varied flood intensities. To maintain computational simplicity across large portfolios, conventional frameworks almost exclusively calculate physical damage as a univariate function of flood depth, as water levels are the most widely available hazard intensity measure. While specialized multivariate models incorporating parameters like flood duration and flow velocity have demonstrated superior predictive performance, standardized risk practices have yet to widely adopt them (Grahn, 2020; Nofal et al., 2020).

Crucially, this focus on hazard-side variables has occurred at the expense of capturing the physical variables that dictate the vulnerability of the asset itself. While advanced component-level FRAs discretize building inventories into distinct architectural components to model localized damage mechanisms, their underlying resistance parameters are built upon rigid simplifications. A strong simplification among these is the assumption that the building stock is structurally intact and pristine at the moment of hazard onset (Stephenson & D'Ayala, 2014).

Historic and ageing structures are predominantly composed of absorbent, porous materials that are frequently compromised by age, weathering, and deferred maintenance long before a flood occurs (Stephenson & D'Ayala, 2014). Therefore, in real-world scenarios, an asset's pre-flood conservation state could significantly govern its resilience to external stresses.

While a traditional probabilistic vulnerability model accounts for general population variance, it assumes a generic, homogenized distribution of asset conditions (blending new, well-kept, and deteriorated buildings into a single "population curve"). When applied to an individual building or an ageing neighbourhood known to be in a poor conservation state, utilizing these population-level assumptions could systematically underestimate the true structural risk. In theory, this overestimation of resilience can be resolved by adjusting the vulnerability curves specifically for the pre-existing state of the asset.

To operationalize state-dependent modelling, a profound scale and data gap must be resolved between two distinct academic domains: macroscopic flood risk analysis and microscopic structural engineering.

On one hand, mesoscale flood risk models favour computational velocity and broad (regional) coverage (Aribisala et al., 2022). They utilize highly aggregated, empirical population damage functions based solely on depth (Merz et al., 2010) that abstract complex buildings. These methods are fast, but they lack the mechanical resolution to account for building-specific structural degradation or flood flow velocity-driven hydrodynamic and debris forces.

On the other hand, structural engineers utilize high-fidelity, mechanics-based simulations to evaluate the exact load capacity of single wall panels under explicit fluid actions (Kelman & Spence, 2003). However, their deterministic formulation and complex model mechanics are far too computationally intensive to be scaled up across an urban portfolio of thousands of buildings.

Consequently, purely engineering-based methods remain trapped at the microscale, while mesoscale risk assessments remain blind to structural realities, potentially overestimating the resilience of ageing assets because no account of material condition is incorporated.

1.1 Problem Statement

In conclusion, the main problem is that current mesoscale FRAs lack an analytical mechanism to integrate the pre-flood structural conservation state of ageing assets into probabilistic vulnerability models, resulting in a likely overestimation of building resilience and systematic underestimation of post-disaster disruption.

The first subproblem is that high-fidelity mechanical models can accurately quantify the residual structural capacity of a degraded wall panel, but their deterministic formulation and complex mechanics are too computationally heavy to be parameterized and scaled across a regional urban portfolio. Second, standard FDAs rely on empirical depth-damage curves that treat the building stock as structurally pristine, failing to capture how pre-existing structural damage manifestations mechanically alter the structural capacity of walls subjected to explicit flood forces. Third, current risk management frameworks prioritize hazard-depth parameters, leaving a critical data gap regarding how pre-flood structural degradation scales against non-structural adaptations in mitigating long-term sociotechnical recovery downtime and direct repair costs.

1.2 Objective and Motivation

The objectives are a direct translation of the problem statements. The main objective of this thesis is to develop a state-dependent probabilistic vulnerability framework that integrates pre-flood structural conservation states into flood risk modelling to accurately quantify building-specific damage and regional post-disaster disruption.

The secondary objectives are three-fold. The first is to design a robust methodology for a critical load-bearing facade element, that translates complex and deterministic structural capacity calculations into a scalable, probabilistic framework. The second sub-objective is to establish a mechanics-based theoretical foundation that quantifies how pre-existing defects in a load-bearing facade element mechanically decrease the structural capacity of the walls under flood demands. The last sub-objective is to implement the state-dependent probabilistic model within a computational engine to evaluate the relative effectiveness of pre-flood structural restoration strategies against non-structural component adaptations (e.g. removing vulnerable components, or moving them to higher floors that are less susceptible to flooding) across variable asset conditions.

1.3 Scope

To be able to meet the objectives, this research is strictly bounded to a single critical load-bearing element: a double-wythe load-bearing unreinforced masonry (URM) facade wall. This component is selected for several technical and strategic reasons. URM represents a dominant architectural typology across a large part of Europe (Figure 1.1). Doubly-wythe walls are common within historic Dutch residential buildings (Rijksdienst voor Ondernemend Nederland, 2022) where conservation states are expected to vary extensively due to decades of environmental exposure and varying maintenance cycles.

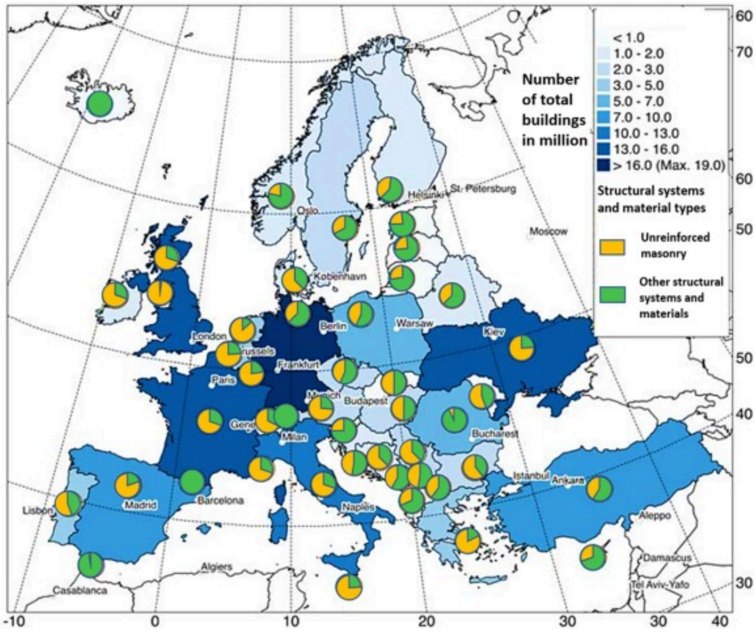


Figure 1.1: Application of URM across Europe relative to the total building stock per country. (Shabani et al., 2021)

The composite nature of brick and mortar creates complex interactions between age-related degradation and flood loads, making it an ideal upper-bound test case for state-dependent vulnerability modelling.

1.4 Research Questions

Directly corresponding to the established problem statements, objectives, and established scope, this thesis aims to answer the following primary research question and contributory sub-questions:

To what extent does the integration of pre-flood structural conservation states alter the quantification of building-specific vulnerability within probabilistic FRA?

- How can the structural capacity of URM wall panels be incorporated into probabilistic, building-specific flood vulnerability models?
- How do pre-existing structural damages alter the structural capacity of URM facade walls when subjected to flood actions?
- How effective are pre-flood structural restoration strategies compared to non-structural component adaptations in reducing repair costs and downtime across different conservation states?

1.5 Relevance

The relevance of this research is centred on establishing a novel, state-dependent framework that bridges the critical scale gap between localized engineering physics and regional risk mapping to transform the field of flood vulnerability modelling. The projected scientific innovation of this thesis lies in the development of an analytical method that explicitly couples an asset's pre-hazard structural degradation state with its expected post-hazard damage severity. Ultimately, this provides a scalable method that provides a robust baseline that supports the future integration of flood vulnerability into multi-hazard asset management.

The projected societal impact centres on protecting vulnerable historic housing stocks and providing emergency management agencies with accurate data to project post-disaster community recovery. By developing a method that explicitly tracks how pre-existing degradation propagates into cascading failures, this delivers an objective, data-driven tool for municipalities, emergency planners, and heritage preservation stakeholders to justify and target pre-flood structural restoration investments. Effectively balancing physical community safety with the preservation of irreplaceable architectural value.

2 Methodology

To establish a scalable, building-specific vulnerability framework that integrates asset degradation into mesoscale flood risk mapping, this thesis departs from empirical population-based damage curves in favour of an analytical, state-dependent simulation design. The research design is structured as a progressive multi-stage approach, where each chapter serves as a distinct methodological block designed to resolve a specific tier of the research questions. Figure 2.1 illustrates this overarching research workflow.

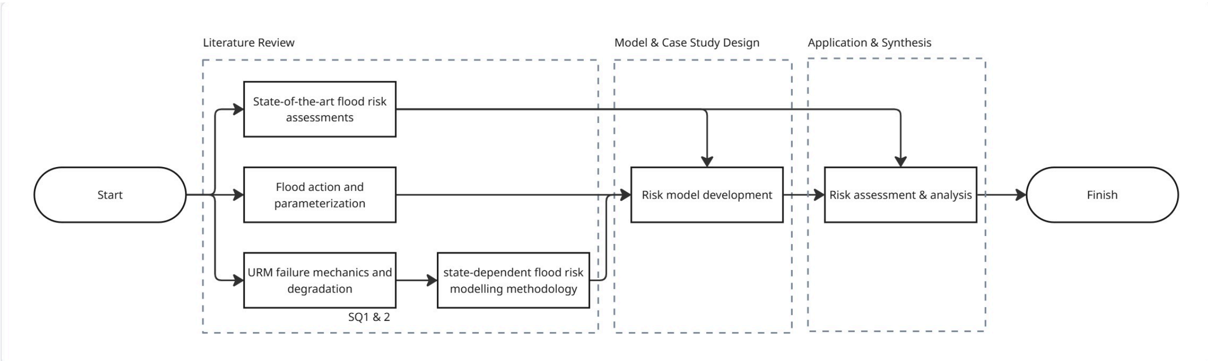


Figure 2.1: A schematic representation the research design.

2.1 Literature Review & Analytical Parameterization

The first phase of this thesis consisted of a literature review, explicitly targeting the first two sub-questions. First, information on state-of-the-art FRA is provided. Second, flood actions and how to quantify their loads are researched, to enable demand quantification from a force-driven perspective later on in the project. Third, the physical and structural rules governing URM failure under environmental actions are reviewed. The literature review established the theoretical baseline necessary to develop the flood risk model and execute the FRA.

To translate microscale structural mechanics into a portfolio-level probabilistic framework, Chapter 3 formulates a new parameterization methodology. This transformation allows deterministic reference structural capacities extracted from experiments or computationally intensive, experimentally validated finite element models to be scaled mathematically using standard geometric handles and linear vertical pre-compression load ratios.

The literature review further captured how initial degradation alters out-of-plane (OOP) structural capacity. By leveraging visual indicators from the international ICOMOS-ISCS glossary (Cartwright et al., 2008) and the Dutch Monument Diagnosis and Conservation System (MDCS)(TNO et al., 2023), pre-existing structural defects were classified by their surface manifestation, completely agnostic to their historical causes. Based on existing damage scales founds in literature, these definitions were mapped to strength reduction factors extracted from post-earthquake masonry frameworks (FEMA, 1998) and analytical micromodeling literature (Ghezelbash et al., 2026), providing a clear mathematical bridge between visible facade conditions and mechanics-based structural capacity reductions.

The literature review finishes with a concluding section where the collected knowledge is synthesized into a state-dependent flood risk modelling methodology.

2.2 Probabilistic Risk Model Development & Case Study Design

The second phase transitioned from theoretical rules to the development of the predictive computational tool for a case study. Chapter 4 describes the architecture of the probabilistic flood risk model engineered in Python to evaluate vulnerability within the aging Dutch building stock.

The computational code builds upon the stochastic Monte Carlo Simulation (MCS) framework defined by Nofal et al. (2020). However, to meet the specific requirements of this research, Nofal's standard depth-driven framework is fundamentally adapted. The code was re-engineered to incorporate a state-dependent structural analysis loop that routes environmental demands and degraded material parameters through to monitor force-driven OOP failure of the load-bearing URM facade. To move beyond direct repair costs, the model incorporated re-occupancy downtime assessment. The framework for the downtime assessment is adapted from the REDi for Floods risk assessment framework as presented by Hogan et al. (2023):

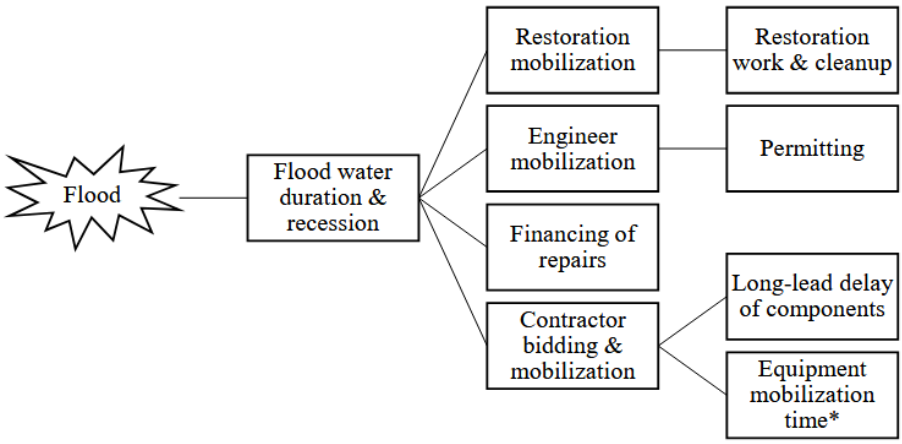


Figure 2.2: Representation of the impeding delay sequences for flood downtime. The longest sequence is governing. (Hogan et al., 2023)

For the risk assessment, it is necessary to parameterize the case study. The case study was defined by the archetype data, URM reference structural capacity thresholds, location flooding data, and mitigation strategies. Archetype geometry data was extracted from the European building typology database (Climate-alliance & IWU, 2026) and official Dutch reference dwellings (Rijksdienst voor Ondernemend Nederland, 2022). For the content library contained within this archetype, the component data provided by Nofal et al. (2020) was adapted based on expert judgement. To fully build up a component library from scratch, the reader is referred to the methodology in their paper. Reference threshold capacities were derived from the experimentally validated results of a non-linear finite element model, as presented by Korswagen et al. (2025). Mitigation strategies were based on expert judgement. They are discussed elaborately in Chapter 4.

To test the archetype under contrasting local conditions, two specific real-world locations were selected to capture the wide applicability of the developed model and methodology. For the selected locations, localized flood depths and return periods for fluvial/coastal flooding and pluvial flooding were retrieved directly from the Dutch National Flood Information Database (LIWO) GIS mapping layers.

2.3 Model Application and Synthesis

In the final phase, the newly developed computational model and methodology were applied to perform a component-level site-specific FRA. By analysing the divergence between pristine baselines and degraded conditions, this phase isolates the true quantitative influence of pre-hazard deterioration. Furthermore, by executing the model with various inputs, the effectiveness of the selected mitigation strategies across different conservation states could be analysed in Chapter 5. The output of this phase delivers the metrics necessary to answer sub-question 3 and the Main Research Question.

To maintain a clean methodological narrative, the granular model variables, case study dataset listings, asset cost matrices, and explicit mathematical algorithms are omitted from this chapter and are instead discussed within the structural descriptions of Chapter 4 and its corresponding appendices.

3 Theoretical Framework

3.1 Flood Risk Assessment

To evaluate the contribution of this thesis and provide the necessary state-of-the-art baseline from which this thesis departs, understanding is required of the broader landscape of contemporary FRA methodologies. Historically, risk is defined as a function of three overlapping core components: the hazard, the exposure, and the vulnerability of the asset portfolio (Eq. (3.1)) (Intergovernmental Panel on Climate Change, 2014). The Dutch Framework for Climate Adaptive Buildings (Dutch Green Building Council & Climate Adaptation Services, 2022) defines the hazard based on its probability, intensity, and geographic size. Exposure refers to the assets within the affected region, and vulnerability covers the resistance of these assets to withstand the hazard. While this conceptual formulation is universal, the exact methods used to quantify each component vary significantly across spatial scales, FRA models, and loss definitions.

$$\text{Risk} = \text{Hazard} \times \text{Exposure} \times \text{Vulnerability} \tag{3.1}$$

Many FRA models exist, which are useful for specific purposes. The operational resolution of an FRA model is dictated by its spatial scale, which governs the level of detail captured on both the exposure and vulnerability sides of the risk function. Three spatial scales can be distinguished: the macroscale, mesoscale, and microscale (Aribisala et al., 2022).

Macroscale FRA methods are designed for national or regional coverage, such as entire countries or entire municipalities. These models utilize highly aggregated spatial indicators. A prominent Dutch example includes the '*Klimaatschadeschatter*' (i.e. climate risk estimator), which estimates regional climate-induced building losses based on extreme precipitation events for the entire country. Losses are fixed at €276 per m^2 for every building considered flooded, independent of its typology or function (Van de Ven et al., 2018).

Mesoscale models evaluate distinct land units or discrete residential and administrative units (Aribisala et al., 2022). The primary institutional model deployed in the Netherlands at this scale is the *Schade- en Slachtoffer Module* (SSM)(translation: Damage and Victim Module), which powers the flood damage maps hosted by the National Flood Information Database, LIWO.

Microscale modelling addresses building-specific risk by explicitly capturing the extreme physical heterogeneity of the exposed building stock (Aribisala et al., 2022). Component-level FRA methods address this scale. Examples are the method developed by Nofal et al. (2020) and the REDi approach for floods as presented by Hogan et al. (2023). They break a single asset down into its individual component inventories, such as structural walls, floor finishes, electrical fixtures, and appliances.

These methods can be either probabilistic or deterministic in nature. A probabilistic method is an analytical approach that explicitly acknowledges and accounts for uncertainty, variability, and randomness within a system. Instead of using fixed values to predict a definitive outcome, a probabilistic method treats inputs as ranges of potential values defined by statistical probability distributions. On the other side, an example of a deterministic method is an approach which uses finite element method (FEM) modelling. This approach is especially effective at capturing complex mechanical failure mechanisms. Advanced FEM capture precise material physics and failure behaviour under deterministic forces (Vaculik, 2012). While microscale assessments can cover building specific risks better, they are significantly more computationally intensive, and time-consuming due to the high density of input parameters required per property.

3.1.1 Empirical versus Synthetic Models

Besides differences in spatial scale, a primary division within flood damage modelling relates to the source and nature of the vulnerability data utilized, distinguishing between empirical and synthetic methods. Empirical models struggle with the scarcity of collected data after flood events, while synthetic models often struggle with limited knowledge on damage mechanisms (Dottori et al., 2016; Merz et al., 2010). Marvi (2020) captures the different characteristics of these two approaches. Empirical models are based on real damage data from historic flood events. They can be based on post-flood survey data, or insurance claims' data. However, this data includes bias, cannot capture changes in building stock, and are dependent on recent flood events in the region of interest. Synthetic models do not share these downsides, but are still based on assumptions regarding susceptibility of the components to water. Synthetic methods can make use of analytical models that mathematically capture the complex real world behaviour of components.

Schröter et al. (2014) compared the results of probabilistic synthetic models to deterministic empirical models and found the first to be more reliable. Still, probabilistic models are scarce and the effect of including uncertainty in damage mechanisms needs to be further researched (Dottori et al., 2016). This is supported by Grahn (2020) his plea for transparency of assumptions and uncertainties in the model results.

3.1.2 Univariate versus Multivariate Models

On the hazard side of the risk function, the parameterization of flood intensity measures (such as water depth, flow velocity, flood rise rate) further divides FRA models into univariate and multivariate methods. Univariate methods are the most common method to assess damage and rely on the classical assumption that water height and its relation to structure height is the single most vital variable determining the expected value of building damage (USACE, 1992; Penning-Rowsell & Chatterton 1977; as cited in Grahn, 2020). Multivariate methods consider two or more concurrent hazard intensity measures. Empirical research consistently demonstrates that multivariate models significantly outperform univariate approaches in their ability to accurately predict damage (Grahn, 2020). However, standardized institutional risk practices have yet to widely adopt multivariate methods (Nofal et al., 2020).

3.1.3 Losses

Bringing hazard, exposure, and vulnerability together, flood risks are expressed in terms of losses. These losses are typically divided into tangible and intangible losses. Tangible losses are those that can be quantified in economic terms. Intangible losses are often more difficult to express, such as culture loss, social disruption, and emotional damages, such as anxiety and stress (Bruijn et al., 2015; Hogan et al., 2023; Nadal et al., 2010). Additionally, a division can be made based on direct and indirect losses. Direct damages occur during the flood due to contact with floodwater, while indirect damages are caused by disruption of activities (Aribisala et al., 2022; Nadal et al., 2010). Most assessment methods for flood risk express the damage only in direct tangible losses, neglecting intangible losses.

Indicative metrics for these losses are the damage ratio and downtime. Since the monetary value of assets change over time, it is common to express the direct tangible losses as a fractional damage factor, the damage ratio, ranging from 0 to 1 based on the total building replacement costs (Nofal et al., 2020). The relation between flood depth and damage ratio is presented as a vulnerability curve. Figure 3.1 is an example of such a vulnerability curve from which the damage ratio can be extracted for a given flood depth. Downtime is an indicative metric for intangible losses. The REDi method for floods defines downtime as the time it takes for a building to recover up to a predefined recovery state (i.e. re-occupancy, functional, or complete recovery)(Hogan et al., 2023).

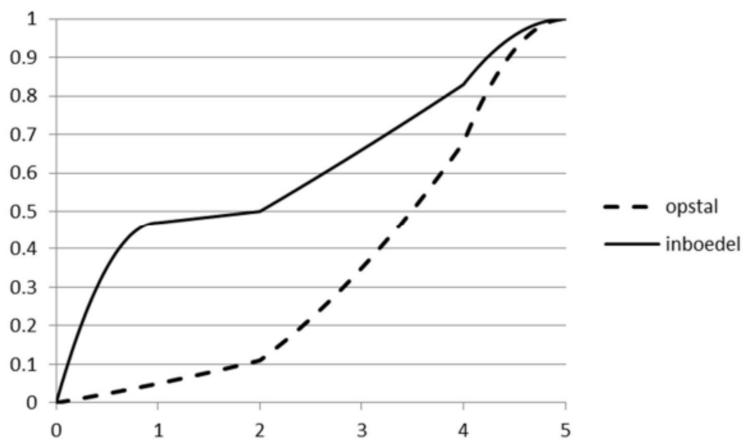


Figure 3.1: Vulnerability curve for direct damages to single-family homes in the Netherlands, as used in SSM-2017. Vertical axis: damage ratio; horizontal axis: inundation depth. Continous line: contents; dashed line: structure. (Slager & Wagenaar, 2017)

3.1.4 Probabilistic Component-Level Modeling

As previously discussed, probabilistic methods capture the uncertainty surrounding material damage mechanisms and the variance of environmental actions. Rather than using a rigid deterministic threshold where an asset instantly fails at a single water depth, probabilistic modelling acknowledges that component capacities exist as statistical distributions. Advanced microscale vulnerability models discretize entire buildings into individual component inventories for which each component is assigned its own stochastic depth-resistance distributions for the flood depth. In the methodology provided by Nofal et al. (2020), these distributions are presented as truncated normal distributions. Figure 3.2 shows how these distributions capture the probabilistic failure depth of a component (depth-resistance). The blue box represents probable placement heights across a larger building portfolio. Failure of a building component can be defined as the point at which repair or replacement is required to restore it to its pre-flood condition (Van de Lindt & Taggart, 2009).

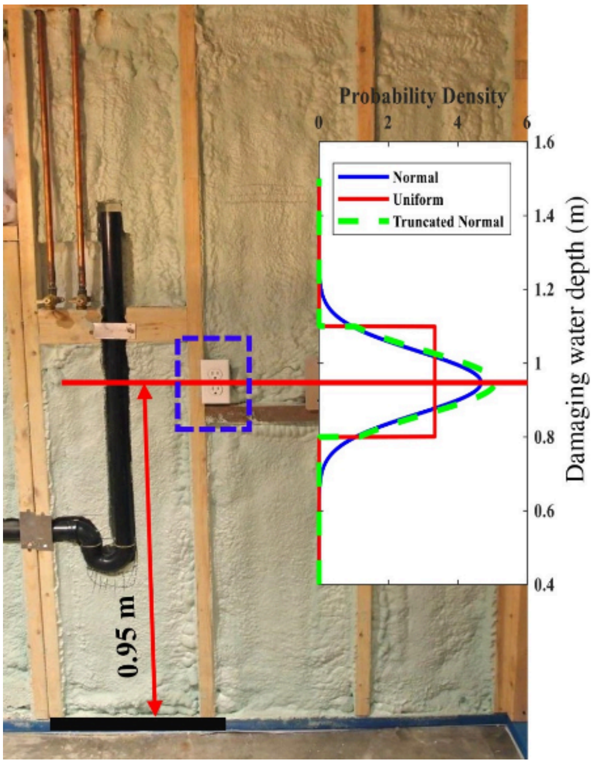


Figure 3.2: An example of the truncated normal distribution modeled for the damaging water depth threshold of an electrical outlet against baseline continuous normal and bounded uniform probability distributions. (Nofal et al., 2020)

Through a MCS loop, thousands of random samples are generated from these component distributions. MCS is a computational method that uses repeated random sampling to simulate probabilistic outcomes. Instead of trying to calculate a single result directly, it solves a problem by running a scenario thousands of times to see what happens on average. Through this method the cumulative failure probability of a component given a certain flood depth can be calculated. This cumulative distribution function (CDF) is plotted as what is called a fragility curve (Figure 3.3).

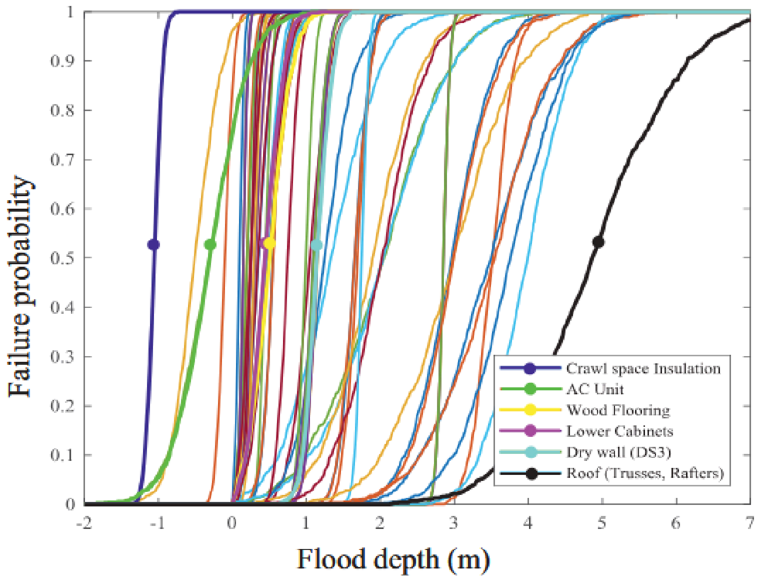


Figure 3.3: A multitude of component fragility curves, displaying the probability of failure against flood depth. (Nofal et al., 2020)

There is a recognized deficiency of data regarding the exact relative flood depths required to fail specific building items (Van de Lindt & Taggart, 2009). On top of limited data, building components have a high variability in their response to inundation. A components relative elevation can change from building to building, but this variability is also a consequence of the mechanical characteristics of the components. Therefore, it can be helpful to systematically categorize components into distinct failure modes, based on their structural and material response to inundation.

Van de Lindt & Taggart (2009) distinguish three failure modes:

The first failure mode governs components where damage variability is insignificant relative to vertical dimensions, rendering the choice between repair and replacement independent of flood depth. These elements require complete replacement upon initial water contact. In a typical house, this applies to non-structural finishes and services like carpeting, wood flooring, and electrical outlets, which instantly suffer unsalvageable water absorption, warping, or corrosive short-circuiting.

The second failure mode consists of immediate segmental failure up to a specified failure depth. The failure depth can be equal to the inundation level, or for porous materials, equal to approximately 30cm above the inundation level due to capillary rise. Anything above the failure depth is still in good condition, meaning the component only requires partial replacement. Examples are internal gypsum boards or plasterwork, which undergo rapid softening and dissolution below the flood line, requiring localized cutting away and replacement while the upper portion remains unaffected.

The third failure mode involves component failure at a critical level of submersion: elements that withstand low-level water contact but fail entirely once a specific inundation level is reached. This mainly includes complex, multi-component assemblies whose economic loss profile scales progressively as a function of rising water levels. This behaviour is typified by domestic appliances, mechanical services, and building utilities (e.g. a dishwasher or refrigerator). Rather than exhibiting an instantaneous complete failure, an appliance contains internal sub-components distributed at varying vertical intervals. As the flood depth increases, successive internal mechanisms ranging from low-lying electronic control units to elevated motors are progressively compromised, causing a step-wise accumulation of repair costs. This gradual vulnerability progression continues until a critical economic threshold is reached, where the cumulative cost of repairing the submerged components surpasses the capital expenditure required for total replacement of the asset.

While this categorization effectively parameterizes non-structural components and architectural finishes based on inundation depth, it exhibits distinct limitations when applied to primary structural elements. For load-bearing URM walls, failure is not driven by material exposure to water contact alone, but it is fundamentally demand-dependent, driven by external forces (see Section 3.2 and Section 3.3)

Therefore, this thesis treats structural components as a different category. This category is defined by gradual, demand-dependent failure mechanisms, transitioning through multiple levels of damage severity (damage states). Rather than failing instantaneously, the component degrades as a function of the hazard parameters. This is where state-dependent fragility modelling comes into the picture.

For non-structural components the pre-flood conservation state of the item has a negligible impact on the final vulnerability, as they require total replacement regardless of whether they were brand new or poorly maintained. Therefore, within this research framework, state-dependent fragility modelling is strictly isolated to the structural envelope, specifically the unreinforced load-bearing masonry walls, where the pre-existing conservation state directly alters the mechanical threshold at which structural failure modes are triggered.

3.1.5 Risk labels

Something not yet touched upon, is the assessment of (multi-hazard climate) risk through scoring systems. These scoring systems ascribe risk labels to specific buildings, based on their characteristic and properties. Examples are the recent Dutch Framework for Climate Adaptive Buildings (Dutch Green Building Council & Climate Adaptation Services, 2022) and the MULTICARE Framework (Bianchi et al., 2025). These frameworks aim to assess the more comprehensive climate risk (considering multiple hazards) on a building level. While they consider flooding as well, they are not specifically tailored for detailed quantitative assessment, but rather provide a more qualitative approach.

3.2 Flood Actions

To move from a large-scale flood hazard to a building-specific vulnerability assessment, the environmental actions imposed by a flood event must first be characterized. The aim of this section is to establish the structural demands that act upon the building envelope during a flood. This understanding provides the physical basis required to later couple external flood demands with the state-dependent structural capacity of the URM walls. The analysis is structured into two sections, covering macroscopic flood characteristics and localized physical forces.

3.2.1 Flood Types

To evaluate the structural vulnerability of a property, the origin of the flood hazard must be categorized. The mechanism of flooding directly dictates the physical boundary conditions of the hazard parameters (i.e. flood depth, flow velocity, debris flows, etc.) that act upon the property. This section discusses four types of flooding that can be distinguished based on its origin: pluvial, fluvial, coastal and groundwater flooding. Discussions of more specific scenarios, like flash floods or storm surges are left out.

Pluvial flooding operates independently of levee breach mechanics and is driven directly by extreme precipitation events, classified into short-duration local intense showers and long-duration heavy precipitation. Short-duration extreme precipitation events can cause sewer systems to be overloaded (3 in Figure 3.4), whereas long-duration events can cause bank overflow of surface water bodies (4 in Figure 3.4).

Fluvial flooding is river-driven. An example is the flooding of unembanked areas due to high water levels in the main rivers (7 in Figure 3.4), but fluvial flooding can also be the result of breaches in the primary or regional flood defences (6 and 5 in Figure 3.4), like dikes and levees.

Coastal flooding is sea-driven and is the result of breaches in the primary flood defences (6 in Figure 3.4). Groundwater flooding is a result of groundwater rising to the surface and is a result of yearly precipitation sums and groundwater management (Nicolai & Oerlemans, 2025) (2 in Figure 3.4).

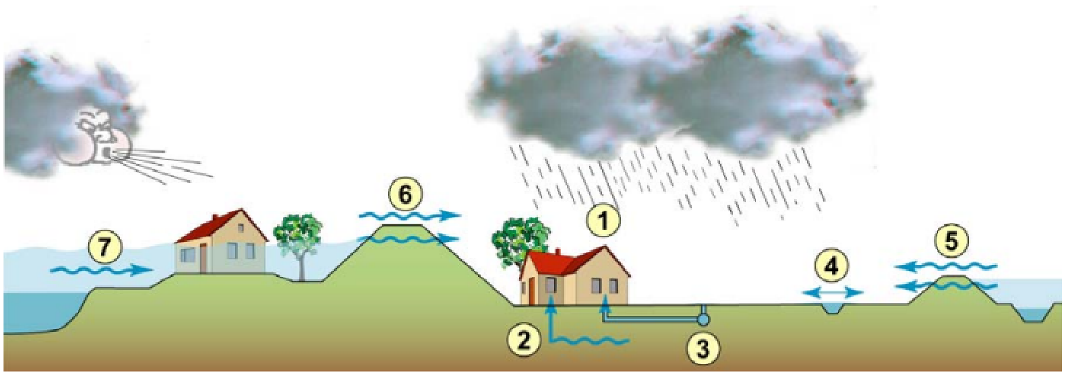


Figure 3.4: Flood types in low-lying areas of the Netherlands. (Kok, 2005)

For this thesis, two types of flooding relating to two specific scenarios will be used for the case studies further on. These scenarios are selected based on their probability and consequence.

Floods with the highest consequences are those with high flood depths or high flow velocities. These floods are mainly driven by the structural breach or overtopping of primary flood defences (including levees, dunes, storm surge barriers, locks, and dams)that insulate the country from major water bodies such as the sea, the main rivers, and large lakes (Nicolai & Oerlemans, 2025). 55% of the Netherlands is prone to flood due to a breach in a primary defence (Figure 3.5). 62% of the Dutch population lives in this area (Mens et al., 2024).

Near the immediate breach location, flow velocities generate devastating dynamic actions (Nicolai & Oerlemans, 2025). While hydrodynamic forces decrease further into the polder, climate change projections indicate that higher sea levels and peak river discharges will increase future hydraulic loads (Dorland et al., 2024). This results in higher potential flood depths and a larger demand on buildings in the flood prone regions if a breach occurs. Although the probability of such an event is extremely low (Pieterse et al., 2009), the national and local governments are working on reducing the probability and consequences of such breaches.

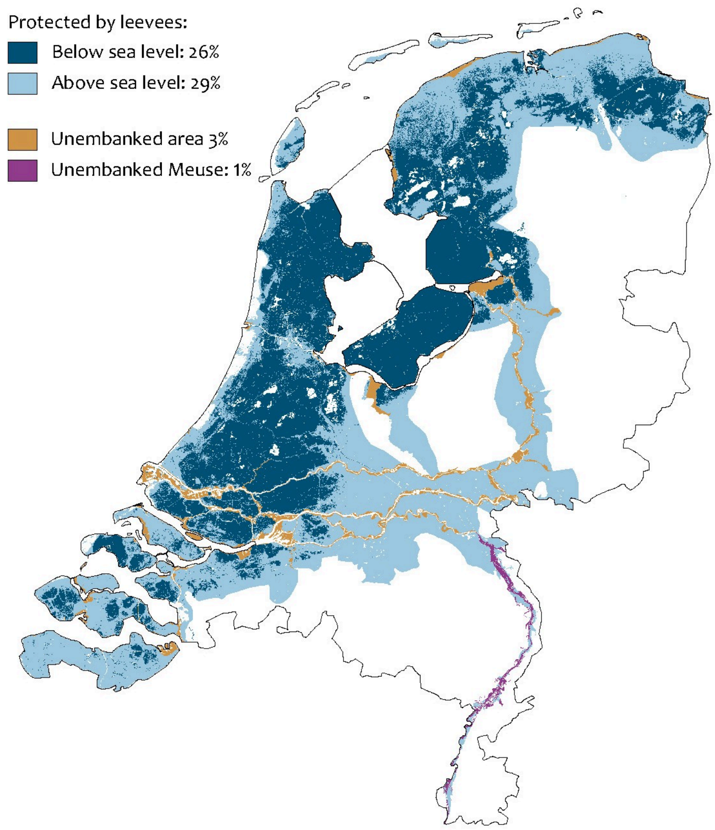


Figure 3.5: Flood prone area due to primary flood defense failure in the Netherlands. (Planbureau voor de Leefomgeving, n.d.)

Flooding due to short-duration extreme precipitation events has a much higher occurrence probability than primary defence failures. The precipitation causes flooding when the volumetric influx of rainwater outpaces the drainage capacity of the sewer networks. According to national reference scenarios mapped within the National Flood Information Database (Landelijk Informatiesysteem Water en Overstromingen, or LIWO), the maximum water depths achieved on street surfaces during these short-duration events are physically limited, typically remaining under $0.5m$. While these floods are rarely life-threatening, the water can still cause significant material damage to ground floors and basements

LIWO serves as the central, authoritative portal for spatial flood hazard data in the Netherlands (Nicolai & Oerlemans, 2025) and is primarily designed to support disaster risk reduction, spatial planning, levee reinforcements, and academic research. It categorizes nationwide flood threats into five unique standardized mapping layers, ranging from primary sea and river defence breaches to localized pluvial events.

3.2.2 Demand Physics

3.2.2.1 Loads on a URM Wall

Multiple loads work on a load-bearing wall that can be categorized into gravity loads, in-plane loads, and out-of-plane loads (Figure 3.6. Although the figure displays a non-URM wall element, the same principle applies for URM walls).

In structural engineering, load-bearing walls are primarily designed based on the gravitational loads they need to carry. These consist of so-called dead and live loads. Dead loads include the permanent weight of the wall itself, along with the walls, floors and roof it supports. Live loads account for temporary, movable loads from occupants, furniture, stored materials, etc.

Flood actions exert lateral forces on the masonry that work on the wall both in-plane and out-of-plane, dependent on the flood flow direction relative to the wall orientation. The structural capacity of the wall to resist flood loads is governed by its resistance to these two forces. Between those two, the OOP structural capacity of a load-bearing wall is governing for typical Dutch residences (Jansen, 2019). Lateral OOP forces work perpendicular to the wall, causing bending moments to occur.

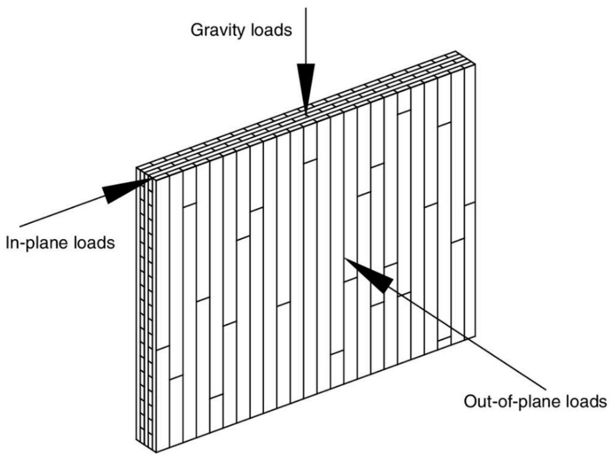


Figure 3.6: A schematic representation of loading scenarios on a wall (Poulin et al., 2017).

3.2.2.2 Flood Actions

Literature provides comprehensive typologies to categorize the environmental actions that work on a building during a flooding event. Kelman & Spence (2004) classify the physical actions exerted by a flood into six distinct classes: hydrostatic actions (including lateral pressure from depth differentials and capillary rise), hydrodynamic actions (comprising velocity-driven drag, localized corner effects, turbulence, and wave actions), erosion actions, buoyancy forces, debris actions (both static and dynamic), and non-physical actions (chemical, biological, or nuclear).

Focusing specifically on historical and cultural heritage structures, Drdácký (2010) expands this understanding by distinguishing fourteen specific actions, highlighting horizontal static pressures, upward buoyancy, low-velocity and high-velocity stream actions, wave impacts, and the dynamic impacts of floating objects. He also notes critical secondary and long-term actions, such as the compacting of soils, changes in subsoil conditions leading to differential settlement, the chemical or biological contamination of porous materials, and the structural saturation of materials where even fired bricks can suffer significant strength reductions and stiffness degradation.

While these comprehensive typologies account for all theoretical flood interactions, Kelman & Spence (2004) argue that parameters like capillary rise, erosion, or chemical actions vary significantly based on localized factors and are difficult to predict deterministically at scale. The lateral pressure differentials and flow velocities represent the highly relevant, predictable actions for estimating structural damage. Similarly, Jansen et al. (2020) state that within standard flood risk and mortality functions, the lateral hydrostatic load and the quasi-steady component of the hydrodynamic load are considered the most critical actions responsible for the structural collapse of residential buildings.

Although Kelman & Spence (2004) argue that debris impacts are difficult to predict at scale, debris action can be an exceptionally influential dynamic load capable of causing extreme damage to structural walls, likely overpowering hydrostatic and hydrodynamic loads (Nistor et al. (2008); Nouri et al. (2010), as cited in Jansen et al. (2020)). Nevertheless, debris is often omitted from standard Dutch flood risk assessments, due to its unlikely occurrence (Jansen et al., 2020).

Other effects like subsoil compacting, and chemical or biological contamination are predominantly long-lasting or post-flood effects that influence residual long-term durability rather than driving immediate, load-induced failure during the hazard event itself (Drdácký, 2010). Consequently, it is common to isolate the lateral hydrostatic pressure, quasi-steady hydrodynamic pressure, and dynamic debris impacts as the three fundamental physical drivers of external demand for fragility models (Moaiyedfar & Deniz, 2025) (Figure 3.7).

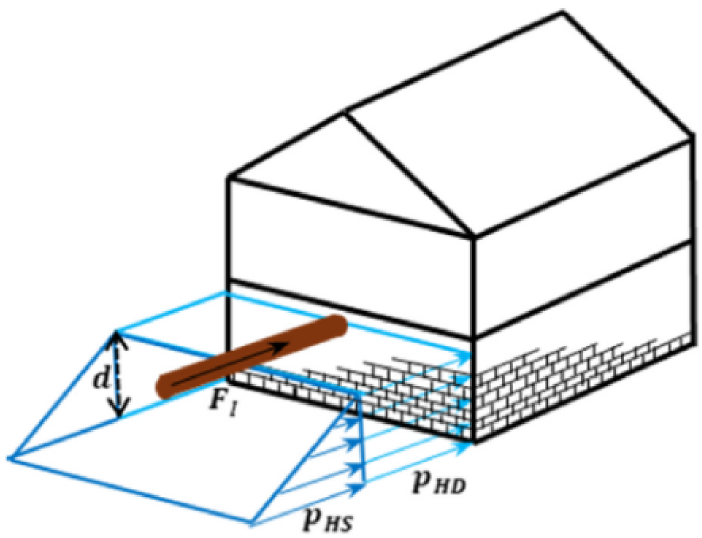


Figure 3.7: A schematic representation of the main flow actions acting on a masonry building, including hydrostatic pressures (p_{HS}), hydrodynamic pressures (p_{HD}), and debris impact forces (F_{I}) (Moaiyedfar & Deniz, 2025).

3.2.2.3 Mathematical Formulations of Primary Forces

To quantify the three primary drivers for structural damage, this section discusses the structural engineering formulations for hydrostatic, hydrodynamic, and debris impact forces prescribed by the Federal Emergency Management Agency (FEMA, 2012) and the formulations used by Moaiyedfar & Deniz (2025), which are based on extensive literature.

The core analytical divergence between these two approaches lies in how the fluid demands are geometrically applied to the masonry wall. FEMA (2012) simplifies the environmental actions for standard engineering workflows by integrating distributed pressures over the wall height into static resultant line loads acting at a single calculated height. Conversely, Moaiyedfar & Deniz (2025) maintain the forces as distributed area pressures across the entire submerged surface, a configuration directly optimized for capturing localized non-linear material strain responses within finite element models.

Lateral Hydrostatic Forces

Hydrostatic pressure is defined as the normal pressure exerted by still water, acting perpendicular to the wall surface and increasing linearly with depth, resulting in a triangular distribution (P_{HS} in Figure 3.7).

While slow-moving overland flows may allow water levels to rise symmetrically inside and outside a structure, thereby reducing net lateral pressures (Stephenson & D'Ayala, 2014), this is highly dependent on the rise rate of the flood and the infiltration rate of the flood water into the building.

The mechanism and rate of floodwater infiltration into a residential building are critical parameters in determining the net out-of-plane forces acting upon the structural envelope. As highlighted by CIRIA & Environment Agency (2003), floodwater exploits multiple entry paths to bypass the building envelope, including seepage through the ground into basements, backflow via overloaded drainage or blocked sewer networks, and direct infiltration through gaps around service utility pipes and cables. Water penetrates directly through the porous brickwork itself, travelling faster across weathered or damaged mortar joints, through cracks in the masonry, or through low-lying air bricks and decayed window or door sealants (Figure 3.8).

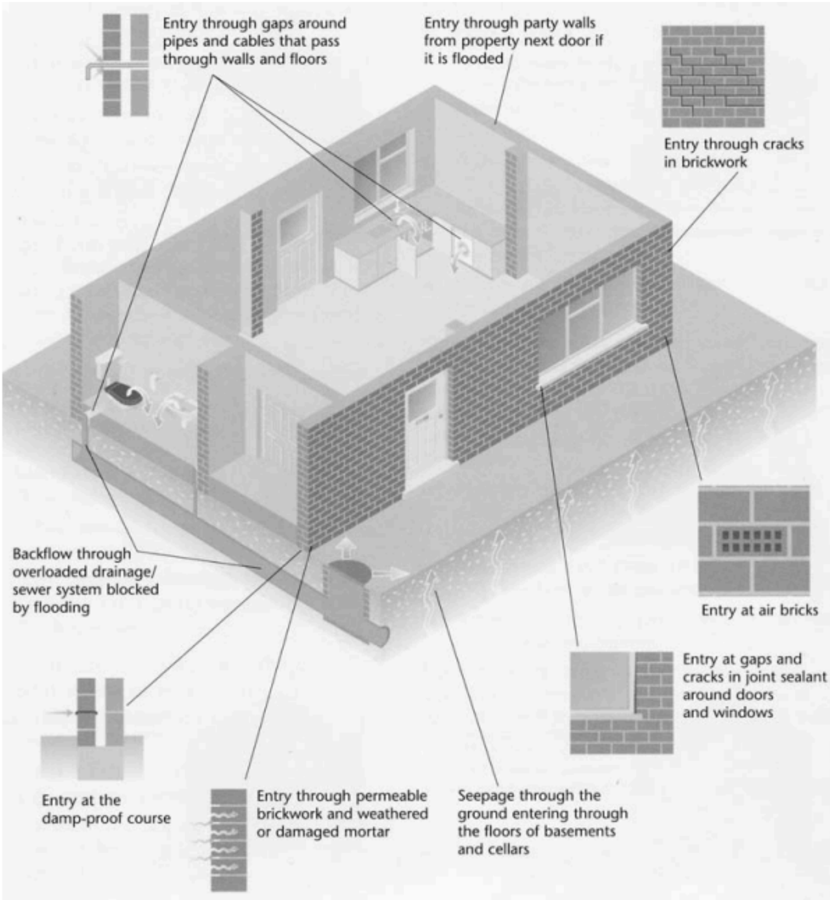


Figure 3.8: (CIRIA & Environment Agency, 2003)

Understanding these diverse infiltration vectors is essential because the velocity of water entry governs how fast internal and external water levels equilibrate; a rapid external rise combined with a low infiltration rate maximizes the net asymmetrical hydrostatic and hydrodynamic head, thereby subjecting the unreinforced masonry walls to its most critical out-of-plane loading condition (Figure 3.9). A quick rise rate prevents immediate internal infiltration, and results in the same critical loading conditions.

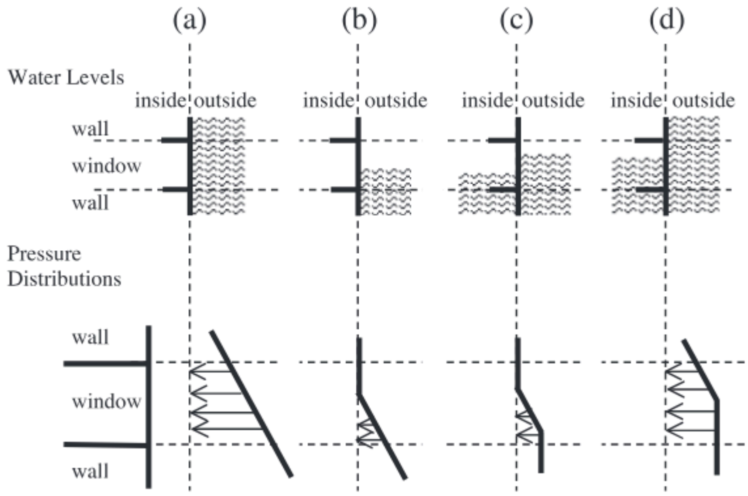


Figure 3.9: A schematic representation of the hydrostatic pressures acting on a wall in four different scenarios (Kelman & Spence, 2004).

The distributed area pressures can be calculated using Eq. (3.2):

$$p_{\mathrm{HS}} = \rho_F \cdot g \cdot d \tag{3.2}$$

where p_{HS} is the lateral hydrostatic pressure in $\mathrm{N/m^2}$, ρ_F is the constant fluid density ($1025\,\mathrm{kg/m^3}$), g is gravitational acceleration ($9.81\,\mathrm{m/s^2}$), and d is the external-to-internal differential water depth (m).

Integrating the triangular distribution into a single lateral line load (f_{HS}) per unit length, designated to act as a concentrated force at a height of one-third of the water depth ($\frac{d}{3}$) above the base, corresponding directly to the center of gravity of the triangular pressure distribution, leads to Eq. (3.3) as provided by FEMA (2012), but adapted to the International System of Units and the parameter definitions by Moaiyedfar & Deniz (2025):

$$f_{\mathrm{HS}} = \frac{1}{2} \cdot \gamma_w \cdot d^2 = \frac{1}{2} \cdot \rho_F \cdot g \cdot d^2 \tag{3.3}$$

where f_{HS} is the lateral hydrostatic force ($\mathrm{N/m}$), and γ_w is the specific weight of the water ($\mathrm{N/m^3}$). The specific weight (γ_w) can be subsituted by the product of metric fluid density (ρ_F) and gravitational acceleration (g).

Hydrodynamic Forces

When floodwaters flow actively around the building envelope, the kinetic energy of the fluid motion imposes hydrodynamic loads that scale as a function of flow velocity and structural geometry.

Moaiyedfar & Deniz (2025) give the formula for a uniform distributed hydrodynamic area pressure (p_{HD} in $\mathrm{N/m^2}$) across the submerged wall, maintaining a constant drag coefficient ($c_{\mathrm{drag}} = 1.0$):

$$p_{\mathrm{HD}} = \frac{1}{2} \cdot c_{\mathrm{drag}} \cdot \rho_F \cdot v^2 \tag{3.4}$$

where v represents the uniform flow velocity ($\mathrm{m/s}$).

FEMA (2012) converts low-velocity (where flow speeds do not exceed $10\,\mathrm{ft/sec}$ or $3.0\,\mathrm{m/s}$) kinetic energy into an equivalent hydrostatic head surcharge (dh)(Eq. (3.5)), which is subsequently translated into a uniform line load (f_{HD} in $\mathrm{N/m}$) applied at mid-height ($\frac{H}{2}$) of the flood depth (Eq. (3.6)).

$$dh = \frac{c_{\mathrm{drag}} \cdot v^2}{2g} \tag{3.5}$$

$$f_{\mathrm{HD}} = \gamma_w \cdot (dh) \cdot d = \frac{\rho_F \cdot c_{\mathrm{drag}} \cdot v^2}{g} \cdot d \tag{3.6}$$

where the drag coefficient scales with the structural aspect ratio, varying from a baseline value of 1.25 for a standard urban building width-to-height ratio between 1 and 12, up to 2.0 for highly elongated structural forms.

Debris Impact Loads

Debris actions represent localized dynamic forces imposed on the vertical masonry envelope. Debris impacts are treated as concentrated dynamic point loads striking the wall at the flood surface level.

Moaiyedfar & Deniz (2025) derive the impact force (F_I) from a theoretical half-sine impulse load profile over an explicit time history:

$$F_I = \frac{\pi w_D v}{2g\Delta t} \tag{3.7}$$

where w_D is the debris weight (N) and Δt is a fixed, deterministic impact duration.

FEMA (2012) determines the impact force using discrete modification coefficients that capture site screening (C_B), shallow-water dampening (C_D), and the structural rigidity of the receiving wall (C_{Str}):

$$F_I = w_D \cdot v \cdot c_{\mathrm{depth}} \cdot c_{\mathrm{b}} \cdot c_{\mathrm{Str}} \tag{3.8}$$

where c_d is a depth coefficient ranging from 0.0 for shallow water under $0.3\ \mathrm{m}$ up to 1.0 for depths exceeding $1.5\ \mathrm{m}$, c_b is a blockage coefficient accounting for upstream sheltering structures, and c_{Str} is an empirical building structure coefficient accounting for the foundation type, factoring the wall's material stiffness directly into the incoming load value. Look-up tables for these parameters are included in Appendix B.

3.2.2.4 Equivalent Uniform Load

Since the different forces are acting on the wall at a different height, it is complex to determine the resulting maximum bending moment. To place all loads on a common basis, the resultants can be converted to an Equivalent Uniform Load (EUL). This results in the uniform pressure w_{eq} that would induce the same peak bending moment in the wall panel as the actual load. Following basic textbook structural mechanics (Hibbeler, 2012) or a simply-supported wall of height Z carrying a concentrated line load P at height y from its base, the reactions at the supports are statically determined and the bending moment reaches its maximum directly beneath the point of application:

$$M_{\mathrm{max}} = \frac{P \cdot y \cdot (Z - y)}{Z} \tag{3.9}$$

A uniformly distributed load w_{eq} on the same simply-supported span produces a parabolic moment diagram with its peak at midspan, given by $M_{\max} = w_{\mathrm{eq}} \cdot Z^2/8$. Equating this to Eq. (3.9) and solving for w_{eq} yields the EUL transformation:

$$w_{\mathrm{eq}} = \frac{8 \cdot P \cdot y \cdot (Z - y)}{Z^3} \tag{3.10}$$

3.3 Unreinforced Masonry

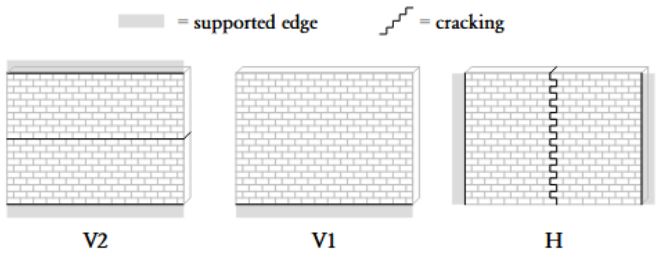
The aim of this section is to discuss the fundamental physical and mechanical principles that govern how unreinforced masonry (URM) walls resist external flood actions. While evaluating the multi-directional and non-linear behaviour of masonry can be highly technical, defining these core material physics is essential to justify the methodology for the state-dependent flood risk model developed in this thesis. The following subsections cover the failure mechanisms of URM walls and outlines how a wall's geometric dimensions, boundary edge supports, and vertical gravity loads influence the walls structural capacity when subjected to flood loads. This section provides the necessary mathematical relationships required to quantify out-of-plane structural resistance thresholds. Ultimately, this theoretical baseline serves as the direct analytical foundation for the operational structural capacity formulations and stochastic sampling regimes developed later within the probabilistic framework in Chapter 4.

3.3.1 Material Mechanics

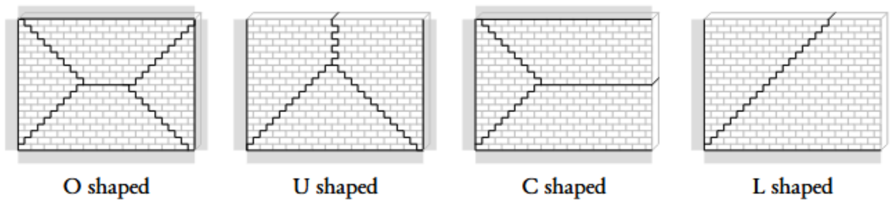
Unreinforced masonry (URM) is a heavy, quasi-brittle, and anisotropic composite material consisting of masonry units (such as fired clay, concrete, or natural stone) set in a mortar matrix (Page, 1996; Sekkaki & Chaaba, 2024; Vaculik, 2012). Anisotropic means the material exhibits distinct directional properties. Characteristic of URM is low tensile (bond) strength at the mortar-unit interfaces, creating potential planes of weakness that directly govern the tensile and shear capacity of the assemblage (Page, 1996). Quasi-brittle materials crack without warning. After the initial cracking the material experiences a rapid reduction in tensile capacity, but some strength still remains (Korswagen, 2024). Consequently, analysing the failure mechanisms of URM is complex due to its heterogeneity and distinct directional behaviour (Sekkaki & Chaaba, 2024) and highly dependent on its loading direction.

3.3.1.1 Out-of-Plane Failure Mechanism

When a URM wall is subjected to lateral or out-of-plane forces, it initially deforms in a linear elastic manner until internal tensile stresses are exceeded, causing cracks to form and widen as elastic behaviour degrades (Korswagen & Rots, 2021; Vaculik, 2012). Depending on the wall's supported edges, it may undergo uniaxial bending (one-way spanning), resulting in cracks parallel to the supports, or biaxial bending (two-way spanning), which leads to a complex combination of vertical, horizontal, and diagonal crack lines driven by normal and torsional shear stresses (Vaculik, 2012)(Figure 3.10).



(a) Vertically (V1 and V2) and horizontally (H) spanning one-way walls.



(b) Two-way spanning walls.

Figure 3.10: Idealized failure mechanisms for different support conditions (Vaculik, 2012).

As deformation continues beyond the ultimate load capacity, progressive joint cracking causes a severe loss of stiffness and initiates a complex softening process (Jansen, 2019; Vaculik, 2012). Ultimately, the fully cracked wall loses strength until it relies purely on gravity-induced rocking and frictional mechanisms. During this post-cracking rocking phase, masonry transitions from a structural component undergoing flexural cracking to a mechanism behaving as a series of rigid blocks.

Figure 3.11 shows how, as lateral displacement continues, the fully cracked section opens, forcing the normal stresses from the vertical gravity loads to be transferred to the foundation through a progressively smaller contact area at the pivot edge or “toe” of the block. This severe reduction in contact area causes a dramatic concentration of compressive stress. The local compressive stresses rapidly increase as the contact area shrinks. The wall reaches its ultimate residual bending moment capacity when these highly concentrated stresses exceed the material’s limits, causing the failure mechanism to shift to compressive crushing of the masonry at the toe and spalling. Spalling refers to the cracking, breaking, chipping, or flaking of the brick’s surface layer (Grünthal et al., 1998). Damage then aggravates until the destabilizing moments balance the internal resisting moments, triggering structural collapse as deformations approach the wall’s thickness (Vaculik, 2012).

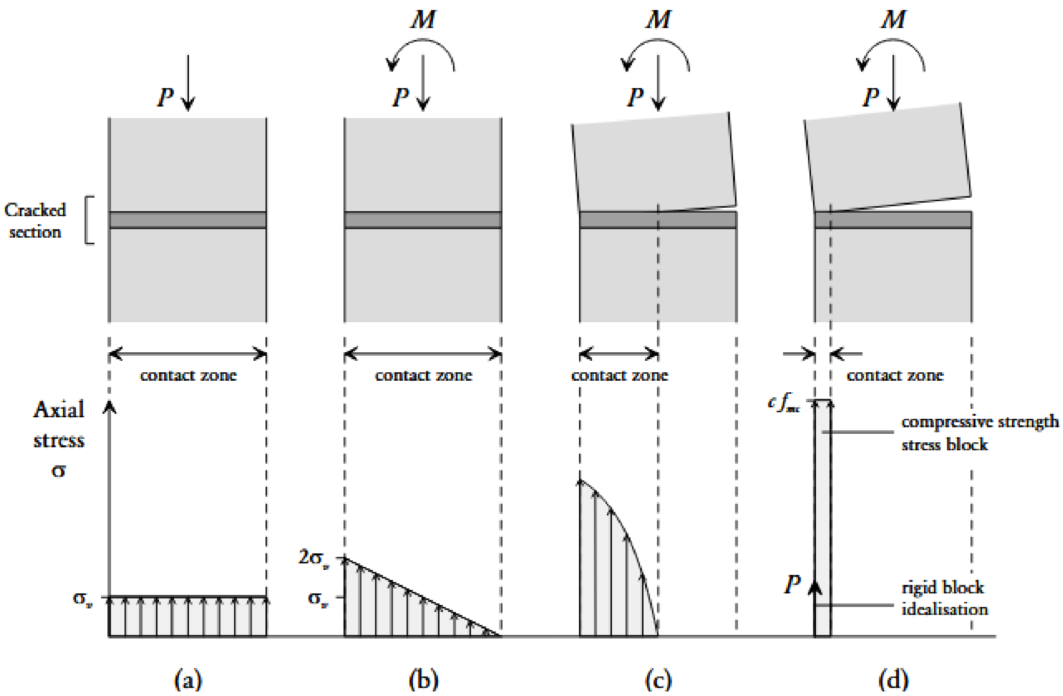


Figure 3.11: A schematic representation of a mortar-brick interface subjected to cracking and increasing rotation (Vaculik, 2012).

While design standards sometimes conservatively allow two-way spanning load-bearing walls to be analysed as vertically spanned one-way walls (Jansen, 2019), plastic analysis or linear analysis with force redistribution demonstrates that two-way action yields a higher ultimate moment resistance than a simple linear elastic one-way assumption (Noor-E-Khuda & Thambiratnam, 2024). Analytic models clearly show this increase in structural capacity dependent on the boundary support conditions. A comparison of four state-of-the-art models is given in Figure 3.12, some of which are used in national standards, such as the Australian AS 3700.

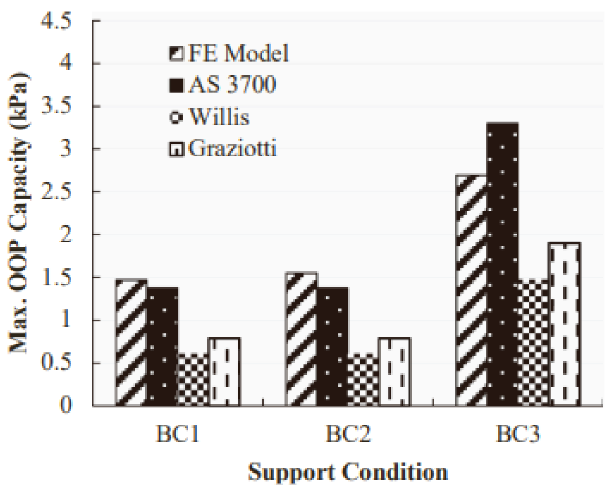


Figure 3.12: Predicted OOP structural capacity for URM walls with different configurations based on four models. BC1: Top and bottom support; BC2: Top, bottom, left support; BC3: Four sides support. (Noor-E-Khuda & Thambiratnam, 2024)

Note how BC1 and BC2 in Figure 3.12 show similar capacities, even though Figure 3.10 (C-shaped) shows two-way bending failure patterns. Apparently, this effect is close to negligible in this support condition, or current state-of-the-art models are not sophisticated enough to capture this behaviour yet.

In a strict vertical one-way bending configuration, typified by a wall supported simply at its top and bottom horizontal boundaries, the cross-section lacks physical redundancy. This means that once the flood loads surpass the flexural tensile strength normal to the horizontal joints, cracks form and no alternative load paths are available. This lack of alternative load paths triggers a brittle snap-through collapse at relatively low hydraulic loads. Figure 3.13 shows the typical appearance of OOP failure under one-way bending. Traditional unreinforced single-wythe panels under pure hydrostatic vertical bending safely resist water depths only up to approximately 90 to 100 cm (Kelman & Spence, 2003; Korswagen et al., 2025).

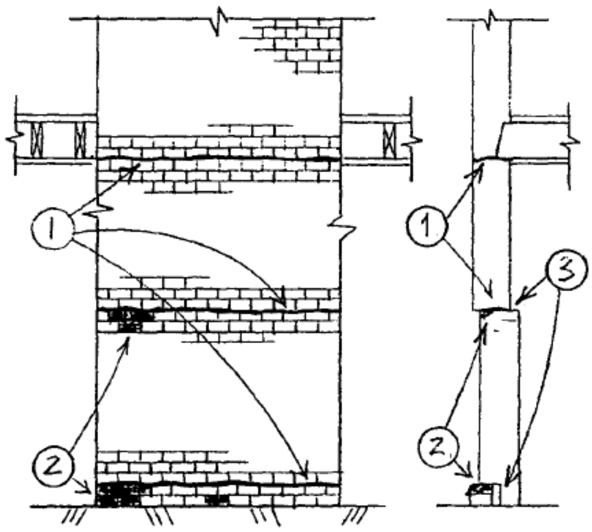


Figure 3.13: Typical appearance of out-of-plane failure under one-way bending. 1: Cracks; 2: Spalling and rounding at edges of units along crack plane; 3: OOP offsets along cracks. (FEMA, 1998)

Conversely, when lateral vertical restraints are present, such as adjacent cross-walls, the structural panel transitions into a two-way failure mechanism. Although lateral boundary conditions do not prevent the initiation of initial flexural tensile cracks, which tend to surface at a similar threshold of around 90 cm water depth (Korswagen et al., 2025), the lateral supports activate a critical secondary load path. Following initial central cracking, the wall successfully redistributes localized pressures into diagonal crack formations that propagate outward toward the supported vertical boundaries (Vaculik, 2012)(Figure 3.10). This structural redistribution capacity significantly extends the ultimate structural capacity of the system, delaying complete out-of-plane failure up to an increased tested water depth of 150 cm (Korswagen et al., 2025).

3.3.1.2 Effect of Wall Geometry

Besides the support conditions and tensile strength of the masonry, the structural capacity of the wall depends on the panel dimensions (i.e. width, height, thickness)(Page, 1996). The wall's height (L) plays a crucial double role. Changing the height alters the wall's structural behaviour in two completely different ways: it simultaneously changes how the wall utilizes its edge supports (its overall bending behaviour) and how flexible the wall cross-section is (its slenderness).

The first role of wall height is related to the overall proportions of the wall, specifically its height-to-width aspect ratio ($\frac{L}{B}$). Even if a walls support conditions imply two-way bending mechanisms, this effect is further governed by the wall geometry. If a wall is very wide and long compared to its height (a low aspect ratio, $\frac{L}{B} < 0.4$, meaning it's over 2.5 times as long as it is high), the side supports are too far away to support the middle of the wall (Noor-E-Khuda & Thambiratnam, 2024). Because the floor and ceiling are the closest supports, the water pressure forces the wall to bend in only one direction (one-way vertical bending) without side-support benefit. Similarly, in a tall, narrow wall ($\frac{L}{B} > 2.0$), the fracture mechanics completely rotate 90°. The main structural failure manifests as a vertical crack line running down the centre of the wall through the head joints and bricks, with diagonal cracks branching off near the top and bottom supports. However, if the wall approximates a square shape, this allows the wall to transition into two-way bending, which delays complete collapse and multiplies its total structural capacity.

The second role of wall height occurs at the material level, where height determines the wall's slenderness ratio ($\frac{L}{t}$) relative to its thickness (t). A taller wall is naturally more slender and flexible, meaning it will bow or deflect much more at mid-height when pushed OOP. From standard physics, the lateral capacity of a wall decreases quadratically as it gets taller, causing the horizontal mortar joints to crack more quickly.

Eurocode 6 is the European standard for the structural design of masonry buildings and civil engineering works and provides a mathematical anchor for the relationship between wall geometry and out-of-plane structural capacity. For a solid unreinforced masonry wall subjected to a uniformly distributed lateral load, the load-bearing capacity (q_{lat}) is governed by the geometric ratio of its thickness (t) to its vertical height (L), as expressed in Eq. (3.11) (as presented by Martens & Vermeltfoort (2017)).

$$q_{\text{lat}} = \frac{f_k}{\gamma_M}\left(\frac{t}{L}\right)^2 \tag{3.11}$$

where f_k represents the characteristic flexural tensile strength of the masonry matrix, and γ_M is the material partial safety factor. By examining the algebraic structure of this standard formula, the individual and joint effects of thickness and height can be assessed.

Because the thickness term is squared (t^2), any modification to this dimension yields a quadratic change in structural capacity. From a physical perspective, increasing the thickness expands the depth of the wall's cross-section, which dramatically multiplies its internal moment of inertia—the geometric property that governs a material's resistance to bending and stretching. Consequently, doubling the wall thickness does not merely double its structural capacity; it multiplies the out-of-plane load-bearing structural capacity by a factor of four.

The vertical height of the wall (L) acts as an explicit mechanical penalty on lateral capacity. Because the height term resides in the denominator and is squared ($\frac{1}{L^2}$), the wall's structral capacity drops sharply as it grows taller. This quadratic reduction represents the structural hazard of slenderness ($\frac{L}{t}$). Under this inverse quadratic rule, doubling the height of a wall cuts its lateral load-bearing capacity down to just one-quarter of its original value.

Notably, the horizontal wall width (B) is completely absent from Eq. (3.11). This omission is a direct consequence of a simplifying structural assumption. The baseline code formula assumes one-way bending, treating the wall as if it were infinitely long or entirely disconnected from its sides and loads are transferred exclusively to the top and bottom support.

3.3.1.3 Effect of Pre-Compressive Loads

Lastly, gravitational loads cause a state of pre-compression in the masonry, increasing its structural resistance (Page, 1996). Pre-compression creates a "clamping effect" on the wall: the vertical weight pushes the horizontal joints shut, directly offsetting the tensile forces generated by out-of-plane water pressure (Willis, 2004). Consequently, a wall's location within a building strongly dictates its baseline structural capacity; a ground-floor wall supporting multiple stories above possesses high pre-compression and requires a much greater lateral force to crack, whereas a top-floor wall carries little weight and fails under much lower lateral demand.

Advanced three-dimensional brick-to-brick numerical simulations conducted by Chang et al. (2021) demonstrate that the ultimate force capacity of an unreinforced masonry (URM) wall subjected to lateral loads scales in a strictly linear relationship relative to the applied pre-compression stress. This linear relationship means that adding a specific increment of pre-compressive stress creates a direct, predictable, and straight-line increase in the wall's lateral capacity.

However, a key finding from this research is that the influence of pre-compression is not an isolated variable; rather, it is highly interdependent with the wall's width-to-height aspect ratio. Physically, this means that the same increment of compressive stress provides a much larger boost in lateral capacity for a narrow wall than it does for a wide wall. This compounding effect occurs because higher pre-compression increases the internal friction across the block-mortar interfaces. Under two-way bending, the masonry can leverage this enhanced friction to dissipate significantly more energy through joint torsion along diagonal cracks, amplifying the total failure threshold of the system.

In a vertical one-way mechanism, the main horizontal cracks form parallel to the joints. Because the vertical pre-compression force acts directly perpendicular to these horizontal crack planes, it operates at maximum efficiency—clamping the cracked joints shut and directly countering the out-of-plane tilting moment. Consequently, based on the findings by Chang et al. (2021) it can be assumed that the lateral OOP capacity increases linearly with the vertical load.

3.3.2 State-Dependent Structural Capacity

Unreinforced masonry structures accumulate damage progressively over time through weathering, repeated minor loading, and deferred maintenance long before any flood event occurs. To capture this pre-existing condition, this section explores damage types and patterns in URM and how they affect the structural capacity of the wall, drawing on the knowledge of Subsection 3.3.1. Next, damage state definitions are discussed, before diving into the literature on structural capacity reduction quantification.

3.3.2.1 Damage Types in URM

To categorize damage objectively, this thesis adopts the international terminology defined by the ICOMOS International Scientific Committee for Stone (ISCS) Illustrated Glossary

(Cartwright et al., 2008) and the Dutch Monument Diagnosis and Conservation System (MDCS) Damage Atlas based on the Thesis dissertation of De Vent (2011).

The core philosophy of the ICOMOS-ISCS system is strictly morphological and descriptive. It focuses entirely on surface appearance, classifying damage patterns solely by “what can be seen” on the material surface without making preliminary assumptions about the underlying structural or environmental causes. Its scope is universal and stone-focused (i.e. not specifically brick-focused), designed to provide a shared international vocabulary for masonry conservation across different geographic regions (Cartwright et al., 2008).

In contrast, the philosophy of the MDCS framework is explicitly diagnostic and component-oriented, tailored specifically to historical brick and mortar assemblies typical of the Dutch built environment. It directly distinguishes between damages occurring in different subcomponents, such as isolating joint mortar degradation from brick unit cracking.

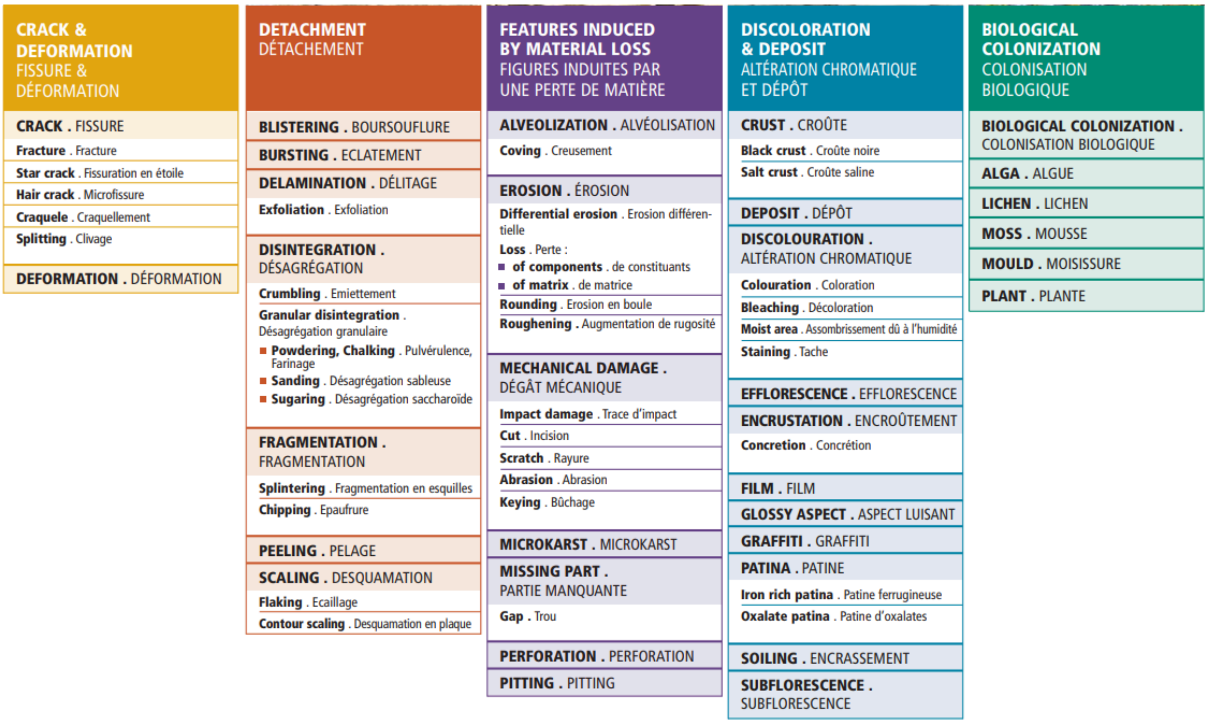


Figure 3.14: Classification of damages according to the ICOMOS-ISCS system. (Grünthal et al., 1998)

This thesis systematically synthesizes both systems to bridge the gap between visual heritage inspection and quantitative structural mechanics. This thesis utilizes the clear, standardized physical definitions from ICOMOS to identify and map the visible condition of the facade elements, while simultaneously leveraging the component division of the MDCS to isolate the effects on OOP capacity, based on whether the damage occurs in the brick or mortar.

Non-structural damages represent superficial changes or surface decay that do not directly compromise the primary load-bearing capacity of the masonry wall. According to the ICOMOS-ISCS classification system, these patterns are primarily grouped into two main classes: discolouration & deposits, and biological colonization (Cartwright et al., 2008).

Discolouration and deposits include aesthetic shifts, surface staining, and the formation of crusts or encrustations. A primary example within brick masonry is efflorescence—the crystallization of soluble salts on the exterior face of the brickwork as moisture evaporates. Biological colonization is characterized by the superficial growth of organisms, such as algae, moss, fungi, and lichens, on the surface of the brick or mortar joints.

From a structural standpoint, these surface alterations in itself are non-critical because they do not reduce the wall's geometry or alter its geometric stability. However, they can be physical indicators of moisture problems for example (De Vent, 2011). Over extended periods, the underlying moisture transport that triggers efflorescence or biological growth can slowly dissolve the binder matrix within historical lime mortars, gradually initiating the transition from non-structural surface weathering to active material degradation (De Vent, 2011).

While non-structural damages provide valuable diagnostic insights regarding environmental exposure, structural damages directly alter the primary load paths and cross-sectional properties of the building envelope, possibly reducing its out-of-plane (OOP) capacity. Here, structural damages are defined as material or geometric defects that directly lower the wall's mechanical resistance to environmental demands. The sections below outline the four critical structural damage categories identified by the ICOMOS-ISCS classification system. For each damage type, it is detailed how it weakens the out-of-plane capacity of the masonry, based on the theoretical background from Subsection 3.3.1.

Cracks

When considering cracks for the state-dependent flood framework, a clear distinction must be made between active cracking caused by fluid pressures as discussed in Subsection 3.3.1 and pre-existing cracking caused by historical events. The progressive formation, rotation, and unzipping of mortar joints under active flood demands represents a load-dependent mechanical response, which has been detailed extensively. Conversely, this section isolates historical, pre-existing cracks and considers them purely as an initial boundary condition before flood onset.

In the ICOMOS-ISCS classification system, cracks are defined as clearly visible fissures resulting from separation of two parts. Integrating this with the theory in Subsection 3.3.1, it can be assumed that at the location of cracks, the local tensile strength is reduced to zero. This effectively pre-divides the wall into independent rigid blocks. When floodwaters now push against the facade, the wall skips its linear-elastic phase entirely and immediately transitions into non-linear, pivoting rigid-body kinematics, significantly lowering the force required to trigger out-of-plane instability.

Deformation

Deformation refers to any structural shift or change in the original shape of the wall partition, without loosing its structural integrity (Cartwright et al., 2008).

Pre-existing deformation alters the load-paths of the gravitational pre-compressive loads that are vital for clamping the masonry joints shut. When a wall is perfectly vertical, the dead load of the upper floors passes straight down through the centre of the cross-section, providing a stabilizing, uniform compressive stress. However, if a wall tilts or bulges outwards, the gravity load line is shifted away from the centre axis, creating an geometric eccentricity resulting in a bending moment. As a result the pre-compressive stresses in the masonry reduce, and hence also the OOP capacity of the wall.

Detachment

Detachment defines the structural separation of outer layers from the core masonry block, visible as blistering, bursting, delamination, disintegration, fragmentation, peeling, and scaling (Cartwright et al., 2008).

The primary effect of detachment processes, is the direct local loss of effective structural wall thickness (t). As established in Section 3.3.1.2, a wall's out-of-plane bending capacity scales quadratically with its thickness. If delamination causes the oute wythe of brick wall to separate from the inner wythe, the effective structural capacity consequently drops significantly, since the effective width is halved. A less severe example is spalling, where the outer surface of brick flakes off. However, due to the quadratic mathematical relation, losing even a small fraction of the wall depth can result in a severe reduction in its moment of inertia, depending on the initial wall thickness, dramatically lowering the panel's capability to resist out-of-plane fluid demand.

Material Loss

Material loss is here described as involving the physical removal of material, either in the brick or mortar. Examples of its manifestation are the loss of pointing, mechanical damages from impacts, and perforation (Cartwright et al., 2008; De Vent, 2011). In the ICOMOS-ISCS classification system this is a broad category, making it difficult to characterize the damage type in more detail or assess the impact on the OOP capacity.

The impact on the OOP capacity is highly dependent on where to damage occurs. Mortar loss directly undermines the interlocking friction between the bricks and the mortar joints. When joint mortar is leached away by water or eroded by age, the contact area between the brick units is severely reduced, reducing the joint's initial cohesion. Alternatively, loss of material in the brickwork can cause the effective thickness of the wall to reduce, causing a similar consequence as detachment.

Moisture Content

An important factor that is not included in the ICOMOS-ISCS classification system, is URM strength reduction due to water saturation. A recent study from Matysek & Witkowski (2026) shows that full water saturation of masonry walls significantly reduces the compressive strength of brick masonry, with reductions of 6-39% reported for historic masonry.

3.3.2.2 Effects on OOP Structural Capacity

To systematically translate pre-existing masonry damages into a quantitative loss of OOP capacity, four distinct interacting characteristics are extracted from the theoretical framework in Section 3.3.2.1: the type of damage, its severity, its spatial extent, and its physical location. Together, these four characteristics determine the reduction in OOP capacity of URM.

Severity measures the physical magnitude of a damage type, establishing the quantitative scale of the structural capacity reduction. E.g. a hairline fracture (< 1 mm}) represents minor joint slippage where the rough surfaces of the bricks remain partially engaged through aggregate interlocking, preserving a fraction of joint friction. Conversely, a wide structural crack (> 5 mm) indicates a complete separation where shear bond strength is entirely lost ($f_t = 0$) and no load can be transferred across the gap. Alternatively, the depth of material loss directly scales the geometric penalty. A superficial section loss of 1 cm due to surface scaling causes a minor reduction in the wall's moment of inertia, whereas deep delamination of 3 cm or a complete loss of an outer brick wythe drastically compromises the cross-section, causing a steep drop in lateral capacity under the quadratic thickness rule.

The spatial extent defines whether a damage pattern is isolated to a specific component or wall section, or is distributed widely across the entire structural system. Isolated patches of eroded joints or a single, short crack represent localized defects. While they degrade local material properties, the surrounding undamaged masonry provides an alternative load path, allowing lateral fluid pressures to safely divert around the flawed zone via surrounding continuous structural lines, limiting the impact. On the other hand, widespread degradation such as joint washout extending across the entire facade surface or a continuous crack spanning the full length of a wall partition, completely breaks the material continuum.

In line with the extent, especially for local damages, the structural impact depends entirely on whether its physical location coincides with the active load paths utilized during a flood event. One can imagine that pre-existing damage located in the inundated wall section is more destructive because it aligns with the location of the structural demand. For cracks, this highlights the importance of their location and direction. E.g. for walls acting in two-way bending, the structural capacity relies heavily on the transfer of forces to the lateral supports. Pre-existing cracks or joint degradation located along the vertical boundary interfaces (the junctions with intersecting party walls) are highly critical. Damage at these boundary locations disconnects the panel from its side supports, preventing the activation of the two-way bending benefit, forcing the wall to collapse prematurely in a weaker, vertical one-way bending mode (Vaculik, 2012).

Lastly, the type of damage dictates which mechanical or geometric property of the structural capacity is explicitly affected (summarized in Table 3.1).

Table 3.1: Damage mechanisms in relation to the impacted structural capacity parameters.

Damage Type	Cause of OOP Structural Capacity Reduction
Cracking	Loss of tensile joint strength Loss of shear and torsional joint strength
Deformation	Reduction of pre-compressive stress
Mortar loss	Loss of shear and torsional joint strength Reduction of effective thickness
General material loss (including as a result of detachment)	Reduction of effective thickness
Moisture saturation	Reduction of compressive strength

3.3.2.3 Repair of URM

The repair of URM is a complex endeavour, especially for historic masonry. Prior to any restoration work, it is important to diagnose the cause of the occurred damage (De Vent, 2011). After the cause has been diagnosed and remedied, the repair strategy can be determined (ICOMOS, n.d.).

In monuments, the preservation of the masonry is very important. Only when damages aggravate, structural stability is threatened, or the damages affect the cultural historic experience too much repair is necessary (Brans, 2012). Dependent on the type of damage, different structural repair strategies can be applied. For joints, if there are cracks or the mortar has been (partially) washed out, the old mortar can be removed, and new mortar can be applied (repointing) (Brans, 2012). Mortar selection is important: introduced mortars should generally be of similar nature to the historic mortar (ICOMOS, n.d.). If damages are present in the bricks, like cracks, or a damaged surface, the damaged bricks can be replaced, or the crack can be injected with a fluid resin or cementitious material to consolidate the masonry (grouting) (Brans, 2012; ICOMOS, n.d.). In severe damage scenarios it can be necessary to apply structural anchors within the joints (stitching, see also Figure 3.15), or to rebuild the wall section completely (Brans, 2012).

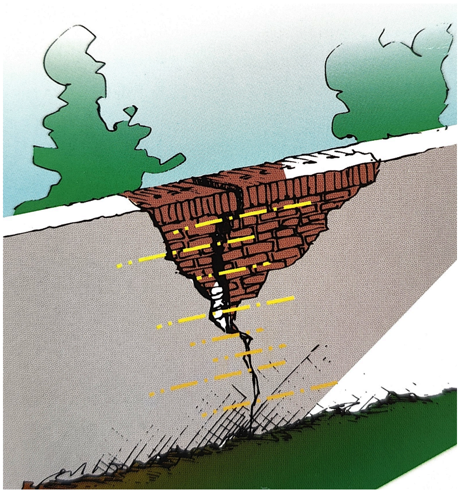


Figure 3.15: Stitching of URM. (Brans, 2012)

3.3.2.4 Damage States

The most fundamental damage scale relies on a binary configuration that distinguishes between an undamaged state and a damaged state. This separation is governed by an explicit threshold value, which represents the point at which a structural lack of performance produces an observable manifestation. To achieve a higher diagnostic resolution, standard damage scales are typically subdivided into multiple additional states or grades, where each discrete state is explicitly linked to a defined range of an observable parameter or a combination of multiple physical parameters. Additionally, when applied to unreinforced masonry structures, these predominantly focus on cracking. (Korswagen, 2024)

Historically, many empirical scales developed to quantify structural movements rely on qualitative descriptors (Korswagen, 2024). An example is the European Macroseismic Scale (EMS-98), where classification of damage is based on qualitative descriptors such as slight, moderate, heavy, or very heavy damage (Figure 3.16). The scale utilizes five progressive damage grades, which are intended to represent a linear increase in the underlying strength of the hazard (Grünthal et al., 1998).

Classification of damage to masonry buildings	
	**Grade 1: Negligible to slight damage** **(no structural damage,** **slight non-structural damage)** Hair-line cracks in very few walls. Fall of small pieces of plaster only. Fall of loose stones from upper parts of buildings in very few cases.
	**Grade 2: Moderate damage** **(slight structural damage, moderate non-structural damage)** Cracks in many walls. Fall of fairly large pieces of plaster. Partial collapse of chimneys.
	**Grade 3: Substantial to heavy damage** **(moderate structural damage,** **heavy non-structural damage)** Large and extensive cracks in most walls. Roof tiles detach. Chimneys fracture at the roof line; failure of individual non-structural elements (partitions, gable walls).
	**Grade 4: Very heavy damage** **(heavy structural damage,** **very heavy non-structural damage)** Serious failure of walls; partial structural failure of roofs and floors.
	**Grade 5: Destruction** **(very heavy structural damage)** Total or near total collapse.

Figure 3.16: Classification of damage to masonry buildings in five damage states, as used in the EMS-98 (Grünthal et al., 1998).

An alternative method to define damage states is to directly map the damages to repair consequences, as is done in the REDi ® flood risk assessment framework (Hogan et al., 2023). By linking damage states to the required intervention, it becomes possible to rapidly assess the financial cost of recovery within a risk model. In practice, this derivation method maps structural damage directly to the maximum width of visible cracks (Korswagen, 2024).

3.3.2.5 State-Dependent Structural Capacity Reduction Factor

Within the scope of this thesis it was considered undesirable to develop a new scale of damage states, based on the discussed damage types, their severity, extent, and location. This is for two reasons. First, it is a complex process to quantify the reduction of structural strength based on damages. This would likely require extensive modelling and experimental validation. Second, this thesis aims to develop a methodology that is usable in multi-hazard risk assessment. For that purpose, it only makes sense to adopt a damage scale already in use for other hazards, such as seismic events. Although these damage scales exclude all other damages that impact the OOP structural capacity of URM, this is considered an operational compromise.

Even when adopting an existing damage scale, the practical challenge remains to translate a damage state into a reduction of its structural capacity. In literature, two damage scales were found that include residual strength (i.e. the strength that pertains after the element has suffered specified damages).

One of these is the report from the Amercian FEMA (1998), the FEMA-306, which provides a systematic framework for evaluating the post-earthquake performance of concrete and masonry wall components by correlating observed physical damage with quantified losses in structural capacity. The damage scale uses qualitative descriptions of damage, but these are linked to quantitative descriptions based on the observed element and its behavior mode, and linked to performance restoration measures. The behavior mode considers the failure mechanism (e.g. in-plane vs. out-of-plane response). Although the framework heavily focusses on in-plane actions common for seismic hazards, three damage states are used for out-of-plane flexural response in combination with a factor used to estimate the loss of out-of-plane wall capacity to damaged URM walls. An overview is given in Table 3.2.

Table 3.2: Damage States descriptions mapped to an OOP capacity reduction factor λ as provided in FEMA-306.

Damage State	Description	Structural Capacity Reduction Factor (λ)
Insignificant	○ Hairline cracks at floor/roof lines and midheight of stories. ○ No out-of-plane offset or spalling of mortar along cracks	1.0
Moderate	○ Cracks at floor/roof lines and midheight of stories may have mortar spalls up to full depth of joint and possibly: ○ Out-of-plane offsets along cracks of up to 1/8″ (< 3.2 mm).	0.9
Heavy	○ Cracks at floor/roof lines and midheight of stories may have mortar spans up to fun depth of joint. ○ Spalling and rounding at edges of units along crack plane. ○ Out-of-plane offsets along cracks of up to 1/2″ (< 12.7 mm).	0.6

The second framework found in literature that explicitly evaluates the residual structural capacity of pre-damaged URM components is developed by Ghezelbash et al. (2026). While FEMA-306 relies on semi-empirical, discrete reduction factors (λ) mapped to qualitative damage classifications, Ghezelbash et al. propose a highly analytical, micromodelling approach designed to quantify the out-of-plane capacity of pre-cracked unreinforced masonry walls. The classified damage states they adopted are heavily supported by literature and are presented in Table 3.3. Their methodology links specific, measurable pre-existing crack widths to a reduction in structural resistance. Table 3.4 displays their results for four wall typologies, where the number of sides indicate how many sides are laterally supported. Only a limited amount of configurations and damage states have researched, so the table contains only the available data.

Table 3.3: Damage classification states with corresponding descriptions, repair levels, and crack width ranges. Adopted from Ghezelbash et al. (2026).

Damage Classification				C_w
State	Class	Description	Repair	
DS0	No Damage	–	–	–
DS1	Light	First Visible Cracks	Easy	1–5 mm
DS2	Moderate	Increased Damages	Medium	5–15 mm
DS3	Severe	Full Crack Pattern	Extensive	15–25 mm
DS4	Near Collapse	Large Deformations	Rebuild	>25 mm

Table 3.4: OOP strength reductions relative to the non-pre-damaged wall for four wall typologies and damage state (Table 3.3). Adopted from Ghezelbash et al. (2026).

Damage State	Strength Reduction Factor (λ)		
	3 sides solid	3 sides with opening	4 sides with opening
DS1	–	–	0.8
DS3	0.82	0.78	0.72

3.4 State-Dependent Modelling Methodology

While sophisticated engineering tools utilize non-linear modelling to explicitly model crack propagation, material plasticity, and localized stress concentrations, simpler analytical formulations remain highly practical (Moaiyedfar & Deniz, 2025). Probabilistic frameworks using Monte Carlo sampling render continuous (non-linear) finite element method modelling of complete building envelopes practically unfeasible. Therefore, this section outlines the theoretical justification for a component-level structural capacity scaling methodology which is applied in the case study. This establishes a foundation for characterizing state-dependent structural capacity of URM walls within a range of building types. To develop the methodology it is necessary to identify the parameters that effect the URM structural capacity, and to define their mechanical roles. To ensure transparency, the localized simplifications, boundary approximations, and structural idealizations embedded within this research model are explicitly quantified and presented.

3.4.1 Damage State and Conservation State

To prevent confusion two operational compromises have been made. First, to clearly distinguish the pre-hazard damage state and the post-hazard damage state, the first is called a conservation state henceforth. Second, to prevent the mixing of state descriptions, both the Damage State (DS) and the Conservation State (CS) are defined on the same four-level damage scale, with identical physical indicators at each level. The distinction lies in the temporal reference, and the absence of CS4, since a building in this state would not experience any more losses during a flood.

The conservation state is a fixed input to the model. It is a pre-existing structural condition that reflects accumulated deterioration from prior loading, weathering, or deferred maintenance. It determines the wall's available structural resistance before any flood demand is applied.

The Damage State, by contrast, is an output of the MCS. It represents the worst structural condition reached during the flood event. Because a building cannot recover from pre-existing damage during a flood, the post-hazard DS is bounded from below by the Conservation State: a wall in CS2 that sustains no further flood-induced damage still exits the event in DS2. This floor is formalised as $\mathrm{DS} = \max(\mathrm{DS}, \max(\mathrm{CS}, 1))$, where the inner $\max(\mathrm{CS}, 1)$ ensures that even a nominally undamaged building (CS0) is assigned at least DS1 whenever floodwater reaches the wall, since cleaning is always required.

The damage scale is derived from the scale presented by Ghezelbash et al. (2026). Likewise, the categorization is governed by the maximum crack width and recovery consequences. In light of the latter, a slight alteration was made to account for the recovery actions drying and cleaning. As a result, damage state 0 to 4 (DS0-4) have shifted down one step on the scale (Table 3.5).

Table 3.5: Damage state description and overview of recovery actions.

Damage State	C_w	Recovery Actions				
		Drying	Cleaning	Repointing	Crack repair	Rebuild
DS0	-					
DS1	< 1 mm	x	x			
DS2	1-5 mm	x	x	x		
DS3	5-25 mm	x	x	x	x	
DS4	> 25 mm					x

Additionally, DS3 and DS4 as described in Table 3.3 have been merged, due to a lack of available data on structural capacity reduction. The values of the structural capacity reduction factor λ_{CS} assigned to each conservation state are based on Section 3.3.2.5. To stay on the conservative side, the lowest structural capacity factors presented have been adopted (Table 3.6).

Table 3.6: State-dependent structural capacity reduction factor λ_{CS} per conservation state.

CS	λ
0	1
1	1
2	0.8
3	0.6

3.4.2 Baseline Structural Capacity Thresholds Derivation

To establish a verified baseline structural capacity for different damage state thresholds, it is recommended here to utilize high-fidelity numerical models that have been experimentally validated. This provides a reliable physical anchor point that reflects actual material behaviour (Korswagen et al., 2025). The specific flood depths at which clear structural damage thresholds are breached in the models are extracted. These discrete depth limits can subsequently be converted into an equivalent lateral line load (f_{HS}) in Newton per meter ($\mathrm{N/m}$) using Eq. (3.3). The lateral OOP-capacity threshold for damage state DS can then be calculated:

$$F_{\mathrm{ref,DS}} = f_{\mathrm{HS}}(d_{\mathrm{DS}}) \tag{3.12}$$

where $F_{\mathrm{ref,DS}}$ is the lateral OOP reference capacity threshold for damage state DS, f_{HS} refers to Eq. (3.3), and d_{DS} is the depth threshold for damage state DS.

For the baseline hydrostatic case, substituting $P = F_{\mathrm{ref,DS}}$ (the triangular pressure resultant per unit wall width) and $y = d_{\mathrm{DS}}/3$ (the centroid of the triangular distribution) into Eq. (3.10) gives the baseline for the equivalent uniform damage state threshold pressure $q_{\mathrm{ref,DS}}$ acting on a simply-supported wall of height Z.

3.4.3 Accounting for Facade Openings

To incorporate the geometric influence of openings within a wall, without transitioning into discrete continuum modelling, it is assumed the facade can be idealized utilizing a simplified equivalent frame model. Normally, an equivalent frame model is used for finite element models and discretizes a wall with openings into an assembly of piers, spandrels, and rigid nodes (Demirel et al., 2012).

To avoid finite element modelling this approach is further simplified for the framework developed in this thesis by discretizing the structural masonry wall into an assembly of piers and spandrels only. These piers are modelled as structural columns spanning the full height of the floor (Figure 3.17). The horizontal spandrels situated above and below openings are not assessed for collapse and do not contribute to out-of-plane flexural capacity. They are solely treated as rigid beams that transfer lateral tributary areas to the adjacent vertical members. By channelling all demands through the vertical piers, the framework captures localized stress concentrations along the primary gravity load paths, ensuring that the analytical model directly accounts for structural openings without requiring computationally intensive modelling.

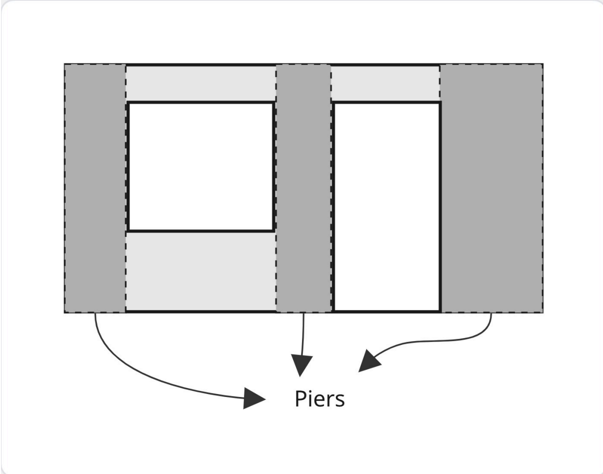


Figure 3.17: A wall configuration with openings and highlighted piers.

Since window frames will not provide lateral support, the outer piers are laterally supported at the top, bottom, and one side, while the middle pier is only supported at the top and bottom. Considering the theory discussed in Subsection 3.3.1 (specifically notice Figure 3.12), out-of-plane bending behaviour for these simplified support conditions, can conservatively be restricted to vertical one-way bending mechanics. This idealization completely omits the secondary lateral stiffness contributions from two-way bending.

While two-way boundary constraints typically shift failure lines into complex yield-line formations and enhance ultimate stuctural resistance, a one-way vertical bending assumption establishes a robust lower-bound structural capacity profile, capturing the most vulnerable mechanical state of a weathered historic wall cross-section.

3.4.4 Configuration Scalar

To translate the discrete structural capacity thresholds derived from a single modelled wall panel to a wall with a different configuration, a configuration scalar is established (γ). The localized structural capacity of a specific wall partition is determined by physical and geometric properties. This scaling logic adjusts the structural resistance based on the geometry and vertical loading conditions. The configuration scaling vector aggregates the mechanical handles of the wall assembly and is defined by Eq. (3.13). In a probabilistic assessment methodology, geometric properties for wall thickness and wall height, and pre-compressive stresses should be sampled (t_{i}, L_{i}, and P_{i} respectively for simulation i). In te presence of openings, P_{i} is dependent on the pier (Figure 3.17) with the lowest pre-compressive stress.

$$\gamma_i = \left(\frac{t_{\mathrm{i}}}{t_{\mathrm{ref}}}\right)^2 \cdot \left(\frac{L_{\mathrm{ref}}}{L_{\mathrm{i}}}\right)^2 \cdot \left(\frac{P_{\mathrm{i}}}{P_{\mathrm{ref}}}\right) \tag{3.13}$$

The geometric modifiers, $\left(\frac{t_{\mathrm{i}}}{t_{\mathrm{ref}}}\right)^2$ and $\left(\frac{L_{\mathrm{ref}}}{L_{\mathrm{i}}}\right)^2$, track the classical flexural mechanics of unreinforced masonry structures following the mathematical relation as defined in Subsection 3.3.1. The vertical loading condition is captured through the linear pre-compression ratio $\left(\frac{P_{\mathrm{i}}}{P_{\mathrm{ref}}}\right)$. For simplicity, the gravitational loads are considered to work on the wall as axial forces, meaning they work in line with central axis of the wall. Opposed to axial loads, eccentric loads are applied off-center, inducing both compression and bending moments.

3.4.5 Residual Structural Capacity Formulation

The residual mechanical capacity of a damaged URM wall can be incorporated into flood risk assessments by implementing a structural strength reduction factor (λ_{CS}, see Section 3.3.2.5), based on the pre-hazard damage state (i.e. state of conservation, CS). For a fully probabilistic approach, uncertainty relating to material properties can be included as a lognormally sampled material variability factor ($\varepsilon_{\mathrm{mat}}$ for simulation i). As a result, the damage state threshold pressure given conservation state CS can be determined with Eq. (3.14):

$$q_{\mathrm{ref,DS,i}}(\mathrm{CS}) = \lambda_{\mathrm{CS}} \cdot q_{\mathrm{ref,DS}} \cdot \varepsilon_{\mathrm{mat,i}} \tag{3.14}$$

where $q_{\mathrm{ref,DS}}$ has been calculated according to Subsection 3.4.2.

3.4.6 Synthesis

The three preceding components can now be assembled into a single expression for the out-of-plane damage-state capacity of an URM wall in simulation run i under conservation state CS. One additional uncertainty source is introduced at this stage: the model uncertainty factor $\varepsilon_{\mathrm{model,i}}(\mathrm{CS})$. While $\varepsilon_{\mathrm{mat,i}}$ in Eq. (3.14) captures aleatory variability in material strength, $\varepsilon_{\mathrm{model}}$ captures epistemic uncertainty arising from the simplifications embedded in the scaling methodology itself (i.e one-way bending, linearised pre-compression, and equivalent frame idealisation). As the conservation state worsens, the wall departs further from the idealised reference geometry on which the scaling was calibrated, so the model's predictive confidence decreases. This is reflected by assigning $\varepsilon_{\mathrm{model}}$ a lognormal distribution whose dispersion parameter β increases with the severity of pre-existing damage (Table 3.7). Bringing everything together, the damage-state structural capacity threshold for pier i is:

$$q_{\mathrm{DS,i}}(\mathrm{CS}) = \gamma_i \cdot q_{\mathrm{ref,DS,i}}(\mathrm{CS}) \cdot \varepsilon_{\mathrm{model,i}}(\mathrm{CS}) \tag{3.15}$$

Expanding by substituting Eq. (3.13) and Eq. (3.14):

$$q_{\mathrm{DS,i}}(\mathrm{CS}) = \left(\frac{t_i}{t_{\mathrm{ref}}}\right)^2 \cdot \left(\frac{L_{\mathrm{ref}}}{L_i}\right)^2 \cdot \left(\frac{P_i}{P_{\mathrm{ref}}}\right) \cdot \lambda_{\mathrm{CS}} \cdot q_{\mathrm{ref,DS}} \cdot \varepsilon_{\mathrm{mat,i}} \cdot \varepsilon_{\mathrm{model,i}}(\mathrm{CS}) \tag{3.16}$$

The failure criterion follows directly: wall i transitions to damage state DS when the total applied Equivalent Uniform Load w_{eq} from Eq. (3.10) exceeds $q_{\mathrm{DS,i}}(\mathrm{CS})$. Because both the demand and the structural capacity are expressed on the same EUL basis, the comparison is dimensionally consistent and can be evaluated as a simple scalar inequality across all N simulations.

Table 3.7: Model uncertainty dispersion β_{model} per conservation state.

CS	β_{model}
0	0.25
1	0.25
2	0.35
3	0.50

4
Case Study & Risk Model

This chapter describes the probabilistic flood risk model developed to quantify how pre-existing wall degradation affects structural vulnerability, direct repair costs, and re-occupancy downtime in an URM terraced house. The following sections describe the physical and probabilistic assumptions underlying each stage in the order they are executed. The main workflow consists of four sequential phases, illustrated in Figure 4.1.

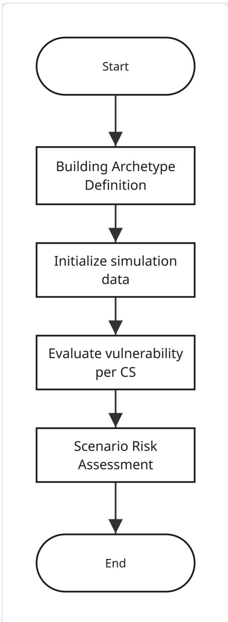


Figure 4.1: Global simulation framework flowchart.

After the initial phase is finished, the risk model is implemented in Python as three sequential steps: a data-generation step that samples all stochastic quantities from specified distribution functions and saves them, an evaluation step that loads those samples and evaluates losses. This separation guarantees exact reproducibility. Post-processing on the same saved data always yields identical results. The model uses MCS to propagate uncertainty from building geometry and material properties through to financial loss and recovery time. Rather than computing a single deterministic answer, the model repeatedly draws plausible combinations of all uncertain quantities and observes the resulting outcomes across a large sample (default $N = 10,000$ simulations). In the last step, location specific risk is extracted for predefined flood scenarios. Statistical summaries of those outcomes form the results section.

4.1 Phase I: Case Study Definition & Data

An archetype is defined as a simplified representation of a real-world system that captures the essential characteristics of the system while ignoring the non-essential details. This chapter discusses the definition of the archetype used for the flood risk modelling in this thesis.

The archetype is based on Dutch residential terraced houses with massive load-bearing URM walls. A terraced house is one of the most common typologies in the Netherlands. In 2006, 37.5% of the existing building stock and still 36.5% of the new build houses in the Netherlands were terraced houses (DGMR Bouw b.v., 2006). Data for the general geometry of the archetype is based on Dutch reference dwellings (Rijksdienst voor Ondernemend Nederland, 2022), and an European building typology database (Climate-alliance & IWU, 2026). Since both sources lack detailed information on the width of the exterior walls, the position of openings, and the floor plan, a reference terraced house for new dwellings is used (DGMR Bouw b.v., 2006) (Figure 4.2) in addition.

The data from these three sources is combined to create a comprehensive geometry for the archetype. The geometry data from the three sources are summarized in Table 4.1. In case of the total floor area, the ground floor area, the closed facade area, the window area, the door area, and the sloped roof area, the data from the RVO and Tabula typologies is averaged to create a single value for the archetype. The building width is based on the data from DGMR. The building depth is a result of the total floor area and the building width. The floor height is assumed to be 2.8m on average. The data used in the case-study is presented in Table 4.2.

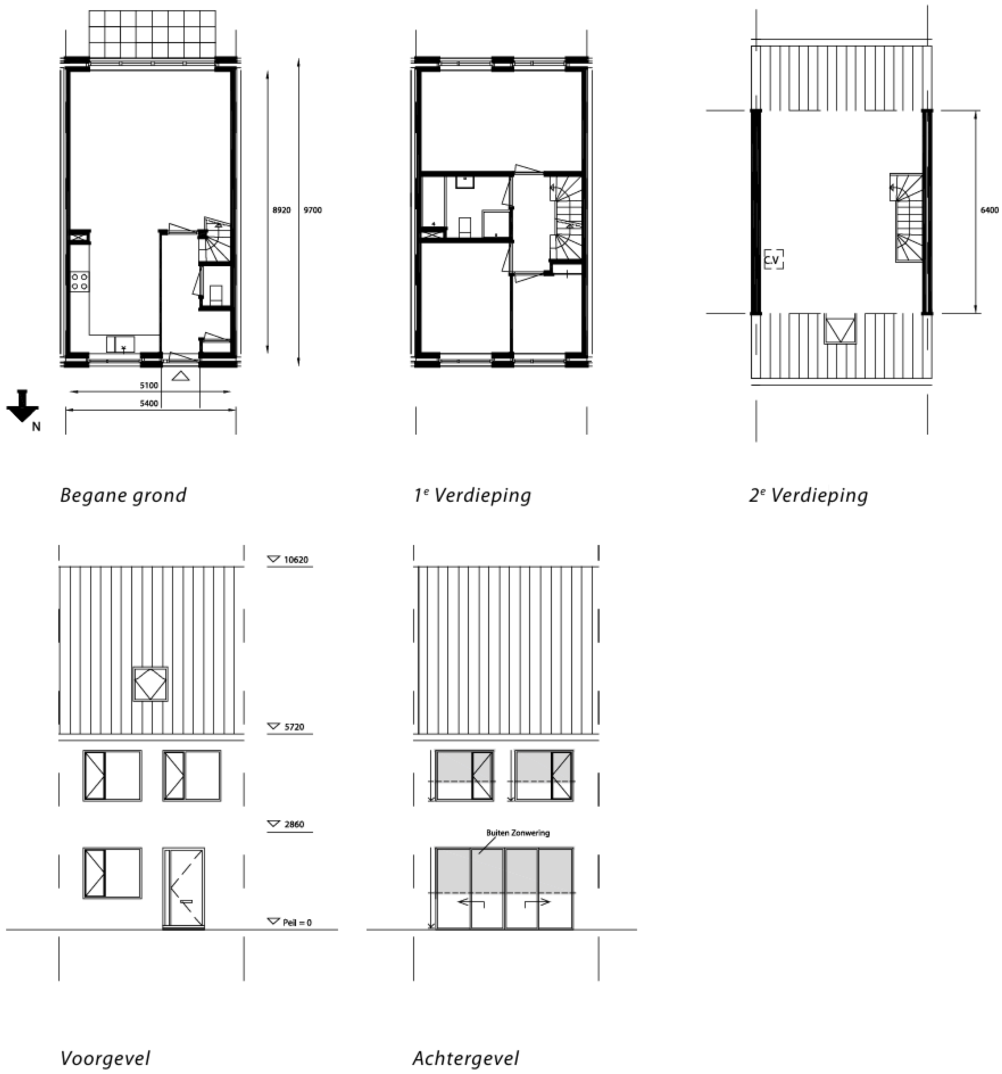


Figure 4.2: A reference terraced house for new dwellings (DGMR Bouw b.v., 2006).

Table 4.1: Geometry data for the archetype.

	RVO		Tabula	DGMR
Building Year	<1946	1946-1964	<1964	2006
Total Floor Area (m²)	108.96	97.87	95.70	
Ground Floor (m²)	48.72	44.21	47	
Closed Facade (m²)	41.58	39.78	49.00	
Window (m²)	19.30	19.86	23.20	
Door (m²)	6.32	4.82	1.30	
Building Width (m)				5.1
Building Depth (m)				8.9
Sloped roof (m²)	42.58	48.00	57.29	

Table 4.2: Case study geometry.

	Case Study
Total Floor Area (m²)	100.8
Ground Floor (m²)	46.64
Closed Facade (m²)	43.45
Window (m²)	20.78
Door (m²)	4.15
Building Width (m)	5.1
Building Depth (m)	9.1
Sloped roof (m²)	49.29

To further capture the heterogeneity of the Dutch terraced house building stock, a variant on the archetype was introduced that includes a basement (Figure 4.3). The basement archetype has a finished floor elevation of 0.5-1.0m above ground. This elevation range was assumed representable, since it allows for a floor construction height of 0.3m and a minimum basement window height of 0.2m.

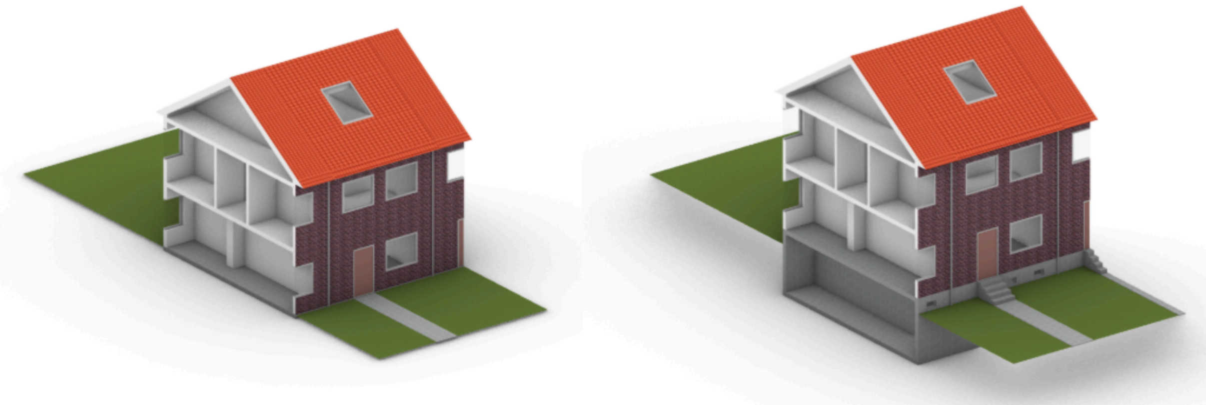


Figure 4.3: Model of the archetype without (left) and with (right) basement.

The asset inventory is discretized into explicitly modelled URM facade elements, which are evaluated via a mechanics-based force engine, and depth-driven asset components (including internal structural elements, finishes, and contents) which are evaluated via depth-damage relationships.

During the flood risk modelling the geometry of the facade is simplified (Figure 4.4). As described in Section 3.3, the structural capacity of URM walls is dependent on the pre-compressive axial loads on the wall. The pre-compressive axial load is dependent on the gravity loads that are carried by the wall and how they can be distributed. It is a combination of the self-weight of the masonry, dead loads from the upper floor and roof, and live loads. For a full breakdown of the load calculations, please refer to Appendix B.

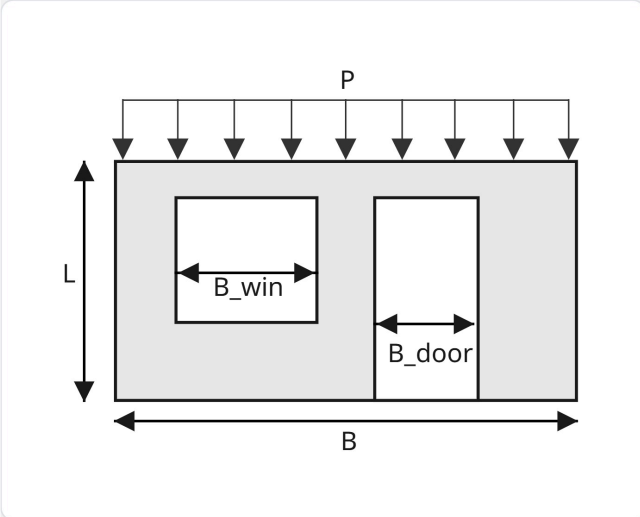


Figure 4.4: Simplified exterior wall configuration of the archetype.

To account for the depth-driven components, the supplementary material from Nofal et al. (2020) was adapted to fit the archetype based on the floor plans and dimensions shown in Figure 4.2. This included changing the unit quantity, elevation, or repair costs of components. As a result, two datasets have been generated with all the non-structural component's depth-resistance, repair cost, and quantity data. One with and one without a basement. For every floor, the contents are categorized in the floor system, building envelope, partition elements, ceiling system, heating & electrical systems, and per room the furniture & appliances. Every component has been ascribed a minimum and maximum flood depth-resistance and repair cost. Based on these limits, a mean and standard deviation is calculated following the methodology of Nofal et al. (2020) to set up a truncated distribution function.

The full list of the components, their flood depth-resistance, repair costs, and quantity are presented in Appendix A. Note that the repair costs are based on a price per unit. The definition of a unit can change from component to component. Hence, a description is included to mark the unit definition per component is included in the appendix.

4.1.1 Structural Capacity Thresholds

This thesis anchors its reference thresholds on the non-linear finite element method model benchmarks presented by Korswagen et al. (2025). Specifically, damage state structural capacity limits are derived from a model that simulated a solid single-wythe wall under one-way bending due to hydrostatic loading.

According to the numerical benchmarks, a pristine wall subjected to an acting vertical stress of $100\ \mathrm{kPa}$ exhibits linear-elastic stability with negligible deformation up to an inundation depth of roughly $90\ \mathrm{cm}$. Beyond this $90\ \mathrm{cm}$ threshold, tensile bond stresses are exceeded, causing small cracks to form along the horizontal joints in the lower third of the wall height where hydrostatic pressures peak. This onset of visible cracking triggers DS2. Utilizing the developed methodology from Section 3.4, this $0.90\ \mathrm{m}$ depth threshold translates to a baseline lateral capacity limit of $q_{\mathrm{ref,DS2}} = 1.16\,\mathrm{kN}\,/m^2$.

The benchmarks show that the assembly transitions to sudden vertical one-way kinematic instability and complete collapse at an increased water depth of $100 \ \mathrm{cm}$, defining the boundary for DS4. Notably, in the one-way bending scenario, DS3 is skipped entirely. This depth threshold translates to a baseline lateral capacity limit of $q_{\mathrm{ref,DS3}} = q_{\mathrm{ref,DS4}} = 1.61 \ \mathrm{kN} \ /m^2$. Although this makes DS3 obsolete within the boundary conditions of this thesis, it should still be maintained to allow scaling the model to include two-way bending wall panels, where DS3 is still relevant. Keeping DS3 as is, also maintains the tight link between CS3 and DS3.

The calculated capacities are adopted as the absolute reference parameters from which all probabilistic state-dependent capacities are derived. The reference capacities per target damage state are given in Table 4.3.

Table 4.3: Reference lateral capacity per URM damage state.

DS	q_{ref} (kN/m²)
DS1	Depth-driven
DS2	1.16
DS3	1.61
DS4	1.61

4.1.2 Case Study Locations and Flood Data

Two buildings were selected to assess building-specific risk. The buildings were selected based on their age, typology, and expected flood depth considering two scenarios: a worst case scenario with a low probability, but likely high consequences, and a pluvial flooding scenario, that has a much lower return period. Building typology was evaluated with Google Maps and Google Street View, while construction year and gross floor area were extracted from the Dutch national database *Basisregistratie Adressen en Gebouwen* (BAG; English: Basic Registration of Addresses and Buildings). The flood depth was extracted from two flood maps provided by Dutch National Flood Information Database, LIWO.

1. Maximum Flood Depth Map: LIWO contains hundreds of synthetic hydrodynamic models that map the maximum local water depth (h) at any given geographical point. These depth maps incorporate hydraulic worst credible flood simulations, based on levee and dike breaches. This scenario is especially relevant for local governments for prioritizing risk reduction measures on an urban or regional level.
2. Pluvial Flood Depth Map: This map isolates scenarios where local extreme rainfall overloads municipal sewer networks. LIWO translates short-duration intense precipitation into national reference grids for specific return periods (typically 10, 100, and 1000 years). The map of the 100-year return period will be used for the assessment, since this return period lies closest to the life-expectancy for buildings. This scenario is specifically relevant for home-owners, since it is very location specific and the risk are likely easier to mitigate on a building level.

The first building (Figure 4.5) is located at Kamperfoeliestraat 71 in Groningen, the Netherlands. It was built in 1932 and has a gross floor area of $67m^2$. The expected flood depths are 0.3 m and 1.32 m for the pluvial and worst case scenario respectively. The building has no basement.

The second building (Figure 4.6) is located at ‘s-Gravendijkwal 54 in Rotterdam, the Netherlands. It was built in 1912 and has a gross floor area of $327m^2$. The expected flood depths are 0.2 m and 2.18 m for the pluvial and worst case scenario respectively. Contrary to the first building, this building has a basement.

Figure 4.7 shows how the flood depth data was extracted from the flood map at the location of building 1 for the worst case scenario.

Figure 4.5: The building of location 1. (Google, n.d.)

Figure 4.6: The building of location 2. (Google, n.d.)

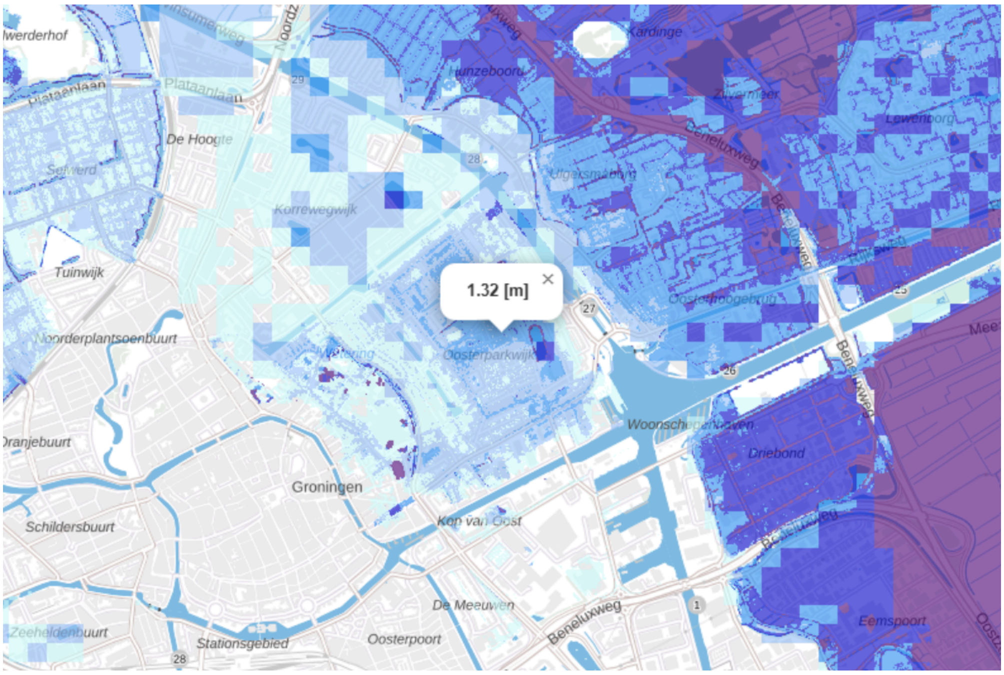


Figure 4.7: LIWO maximum flood depth map, zoomed in on location 1 . (LIWO, n.d.)

4.1.3 Mitigation Strategies

Vulnerability and risk contribution of the components can be evaluated from two distinct perspectives: component susceptibility and severity of risk contribution.

Components that have a higher contribution to the accumulative risk should be prioritized when mitigating building risk. This includes components critical for the utility functions of the building and components with high financial severity. To illustrate, low-lying base electrical outlets introduce systemic electrical risks to the entire circuit loop, while the gas boiler represents a critical utility single-point-of-failure that compromises total building functionality even if structural components do not fail. Additionally, certain components drive aggregate building risk disproportionately due to their high replacement costs. E.g. in the archetype, lower kitchen cabinets possess a mean replacement cost of $12,000 at a mean failure height of $0.45\ \mathrm{m}$, making them the largest financial driver at a low elevation.

To reduce component susceptibility, two characteristics govern the risk: the component's elevation-based exposure and its material susceptibility. From the perspective of elevation-based exposure, components characterized by elevations lower than the expected inundation level can be elevated above this level, to reduce total repair costs.

While this works for components like the base electrical outlets, not all components can be elevated higher (e.g. doors or stairs). For these components, one strategy is to change material susceptibility. Material susceptibility reflects the components material behaviour under submersion and the cascading effects of component failure. By replacing a moisture sensitive component to a "wetproof" component (i.e. able to withstand inundation), the repair costs for that component can drop to cleaning costs only. A critical insight here is the systemic assembly vulnerability of flooring systems. Even if a finished floor is upgraded to a 100% water-resilient material like ceramic tile, moisture trapped beneath it will rot the underlying plywood sub-flooring and insulation. Consequently, the finish flooring must still be destructively removed to allow drying and cleaning of the subfloor, maintaining a high vulnerability baseline.

To conclude, three strategies will be considered for mitigation. The first strategy will be the repair of the damaged URM (i.e. moving from CS1-3 back to CS0 before flooding occurs). The second strategy focusses on all other components. This strategy requires a bit more elaboration, which will be given hereafter. The third strategy considers the combined effect of the mitigation measures. A summary of the alterations made in the components' dataset resembling mitigations is provided in Appendix A.

4.2 Phase II: Data Initialization

All stochastic sampling is concentrated in the first phase so that subsequent analyses are deterministic given the saved data. Five categories of arrays are generated (Figure 4.8) containing the sampled or calculated data per simulation:

1. **Geometry samples:** Wall height (L), facade width (B), wall thickness (t), axial loads (P_{gf}, P_{ff}), and finished floor elevation (L_{FFE}), each drawn once from truncated normal distributions.
2. **Component resistance libraries:** Flood-depth thresholds for non-structural components and lateral force capacities for URM walls, derived per conservation state (CS0-CS3).

3. **Component cost libraries:** Repair costs for every component, sampled from truncated normal distributions.
4. **Downtime samples:** Total re-occupancy time (impeding delays + repair time) for each post-hazard damage state.
5. **Demand matrices:** Hydrostatic, hydrodynamic, and debris impact forces expressed as Equivalent Uniform Loads for every flood depth increment.

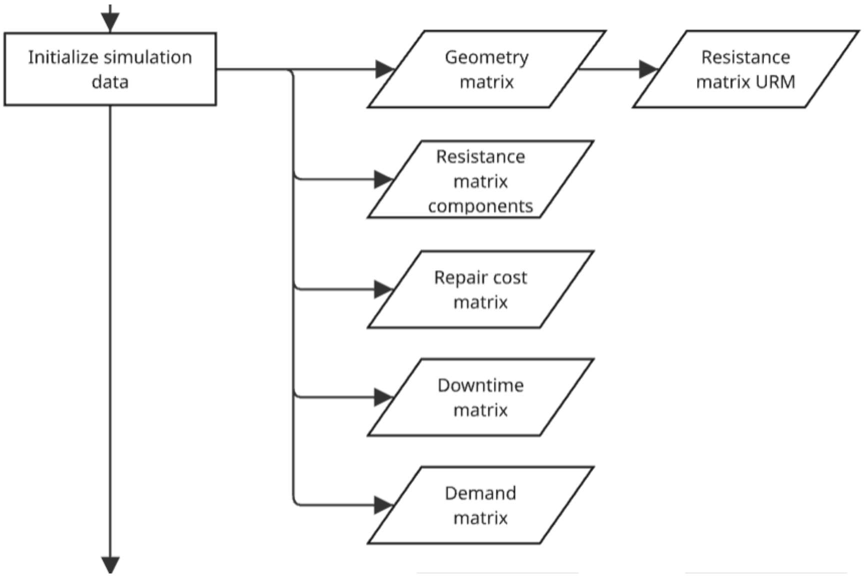


Figure 4.8: Flowchart representing the outputs of the simulation data initialization phase.

4.2.1 Demand Generation

The demand on the URM facade is computed for a continuous range of flood depths (configurable; default 0 to 10 m in 0.01 m increments). Three load types are considered: hydrostatic pressure, hydrodynamic pressure, and debris impact forces. Because the URM structural capacity is expressed as an equivalent uniform load, all three demands must be converted to the same metric before comparison. The total demand on the wall is the sum of the three EUL components. Upper-floor demands activate only once the flood depth exceeds the first floor ceiling level. Figure 4.9 presents a flowchart for the demand generation in the model code.

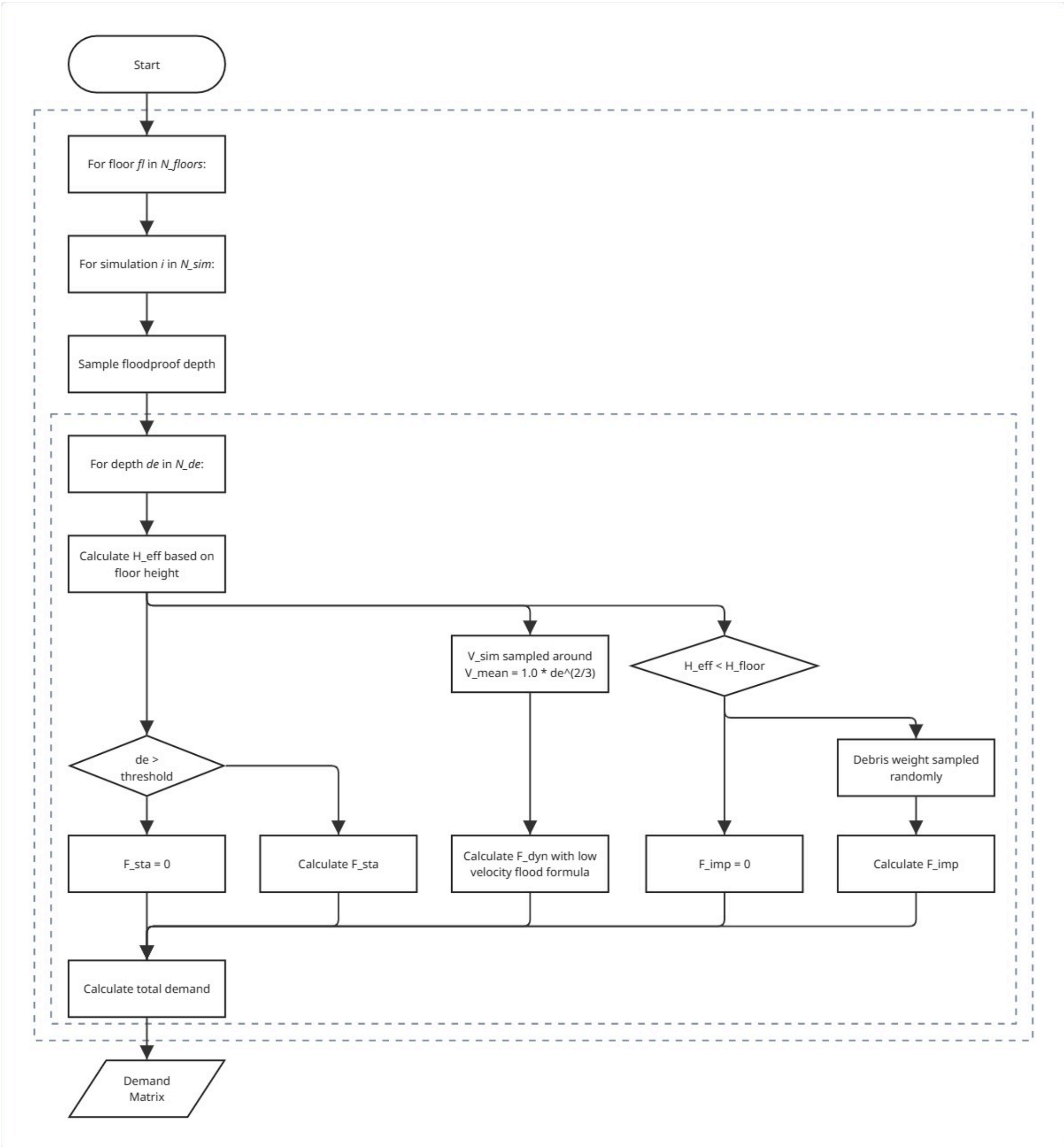


Figure 4.9: Demand generation flowchart (by author).

For structural assessment, the model accounts for partial interior flooding. Once the exterior water depth exceeds the mean failure depth of the exterior door, water is assumed to enter the building and the indoor flood depth will rise at the same rate as the exterior flood depth. The net lateral hydrostatic pressure is therefore decomposed into two components: a uniform block pressure up to the indoor flood depth, and a triangular differential pressure between the indoor and outdoor flood depth (see Figure 3.9). The resultant force and its application height (centroid) are computed as a force-weighted average of the two components.

The hydrodynamic demand is initially calculated with Eq. (3.6), where the drag coefficient is selected from a lookup table based on the facade aspect ratio (see Appendix B), and the flood depth is the effective exterior inundated height of the wall panel, capped at the floor-to-floor height. The main component of the hydrodynamic demand is the flow velocity. In flooding, the flow velocity cannot be considered independent of flood depth. Therefore, for each effective flood depth d_{eff}, the mean flow velocity is estimated from a Manning-type relationship:

$$V_{\text{mean}} = 1.0 \cdot d_{\text{eff}}^{2/3} \tag{4.1}$$

Velocity is sampled from a lognormal distribution based on this mean and a 20% variance, capped at 3 m/s to reflect low speed urban flow conditions (FEMA, 2012). Appendix B elaborates on the relation between flow velocity and flood depth to justify the adoption of this dependency.

To calculate the debris impact forces with Eq. (3.8), the depth, building structure, and blockage coefficients must be determined. These were extracted from the look-up tables provided in the FEMA-259 (FEMA, 2012), included in Appendix B. The debris mass W_d was sampled uniformly to represent any debris between a wooden log and a small vehicle (extracted from table 24 in ASCE 7-22 Supplement 2 (Bennett et al., 2024)). The impact loads are assumed to be distributed over a 1 m effective wall width and are applied at the surface of the floodwater. Since the coefficients for Eq. (3.8) were calibrated for the imperial system, a conversion factor was applied to calibrate the model to the metric system (explained further in Appendix D).

4.2.2 Sampling Resistance and Repair Costs

Non-structural building elements (finishes, MEP, doors, windows, basement tanking, etc.) are assigned flood-depth damage thresholds rather than force capacities, because their failure mode is based on submersion rather than lateral load.

The first floor components (depths 0-3 m) show the earliest and most densely clustered onset of failure. Items at or near the finished floor elevation such as the decking, flooring insulation, baseboard, dishwasher, lower kitchen cabinets, and washing machine, begin failing at near-zero depths, reflecting their vulnerability to even minimal inundation. A second cluster of ground floor components, including drywall (DS1), interior wall finishes, painting, and base electrical outlets, fails shortly after (depths < 0.5m), consistent with their defined mean failure-depths. Mechanically elevated components such as the gas boiler (mean 1.65 m), control box (mean 1.75 m), and ceiling fixtures (mean 2.85 m) show correspondingly later failure onset and tighter curves due to their smaller assigned standard deviations.

Second floor components occupy the 3-6 m depth range, with the floor structure itself (decking, wood flooring, carpet) failing around 3.0-3.5 m. Bedroom contents, bathroom fixtures, and electrical components on this floor follow in a spread between 3.1 and 4.5 m. Ceiling-level components such as light fixtures and the ceiling finish on the first floor extend to approximately 5.85 m, approaching the attic threshold.

Attic components span depths from roughly 6 to 10 m. The roof structure (i.e. trusses, rafters, membrane and sheathing) has the highest and widest depth-resistance range (mean 7.5-8.0 m, SD up to 1.0 m), reflecting uncertainty in the depth at which floodwater reaches and damages roof-level elements.

For the URM facade, the method is more complex for its flood-resistance. Repair costs for the URM are based on Dutch construction cost statistics, ensuring that the simulated costs remain within realistic market limits. The structural resistance of the URM facade against out-of-plane lateral loading is derived using the methodology developed in Section 3.4 with sampled geometry data and reference structural capacity thresholds derived earlier this chapter.

The physical dimensions of the URM facade are not treated as fixed inputs but as uncertain quantities described by distribution functions, reflecting variation within the housing stock. Wall geometry parameters used are the wall height (L), width (B), opening widths (B_{win} and B_{door}), thickness (t), and precompressive stresses (P). These are sampled from a truncated distribution function around mean values. The sampled parameters and their distributional properties are given in Table 4.4. All geometric parameters are stored in a geometry matrix (Figure 4.8), so every simulation index refers to a consistent building geometry throughout the entire analysis. ε_{mat} is sampled lognormally with a variance of 20% and $\varepsilon_{\text{model}}$ is lognormally sampled based on the conservation state (see Table 3.7). Four structural damage states are considered for the URM walls. DS1 is triggered when the flood depth exceeds a stochastically sampled threshold (mean 0.30 m, range 0.10-0.50 m) relative to the finished floor elevation, independent of lateral force. DS2-DS4 are triggered when the applied EUL exceeds the lateral capacity ($q_{\text{DS,i}}$). These DS capacities are stored in the URM structural resistance matrix. Figure C.1 illustrates the data flow.

Table 4.4: Geometry sampling parameters for the Dutch URM terraced house archetype.

Parameter	Mean	Std.	Range
Wall height L	2800 mm	100 mm	2600–3000 mm
Window width B_{win}	2000 mm	200 mm	1600-2400 mm
Door width B_{door}	900 mm	50 mm	800-1000 mm
Facade width B	5100 mm	300 mm	4500–5700 mm
Wall thickness t	220 mm	7.5 mm	205–235 mm
Axial load GF P_{gf}	105 kN/m	5 kN/m	95–115 kN/m
Axial load FF P_{ff}	50 kN/m	2.5 kN/m	45–55 kN/m
Floor elevation L_{FFE} (basement)	0.75 m	0.125 m	0.50–1.00 m
Floor elevation L_{FFE} (no basement)	—	—	0.05–0.10 m

4.2.3 URM Structural Capacity Validation

Kelman & Spence (2003) provide an extensive overview of experimental studies wall panels under an uniform lateral pressure. The experimental results presented there are indicative of the range of expected failure pressures for single leaf URM wall panels, though the results can be somewhat optimistic since walls in residences may actually be stronger than the laboratory suggests, for example due to higher pre-compressive stresses. The results show single leaf wall panels under a uniform lateral pressure would be expected to fail in the approximate range 1-10 kN/m², with the lower end of this range more common than the upper end (Kelman & Spence, 2003).

By running the developed capacity model for a wall panel with thickness of 115mm, a state-dependent fragility curve was created (Figure 4.10). The 50 and 95 percentile capacities have been extruded and are presented in Table 4.5. The pre-compressive stress of 105 kPa is representative for the first floor (i.e. ground floor), while the 50 kPa pre-compressive stress represents the second floor URM structural capacity. The values are similar to the experimental data discussed by Kelman and Spence. The structural capacity is on the higher side, but this can be explained by the fact that the compressive stresses are distributed to the piers, resulting in even higher relative pre-compressive stresses.

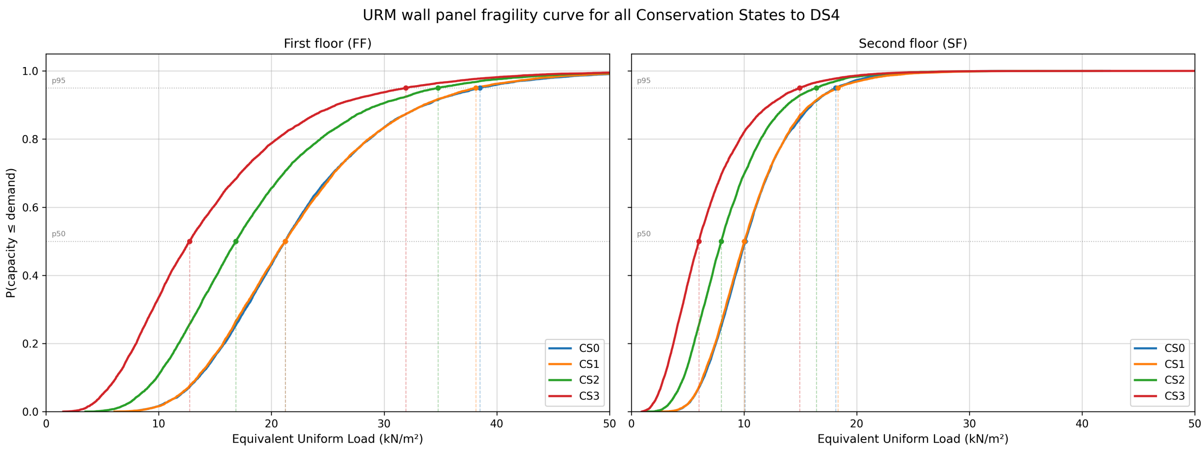


Figure 4.10: State-dependent URM fragility curves for DS4 (by author).

Table 4.5: Model results of single leaf URM wall lateral capacity percentiles per Conservation State for DS4 (collapse) under two loading scenarios.

Pre-compression Mean	CS	DS4	
		p50 (kN/m²)	p95 (kN/m²)
105 kPa	0, 1	5.79	10.58
	2	4.60	9.55
	3	3.71	9.32
50 kPa	0, 1	2.76	4.99
	2	2.18	4.51
	3	1.74	4.37

4.2.4 Downtime Model

Re-occupancy downtime is sampled independently of the depth loop, so that the same draw applies consistently across all flood depths. This preserves within-simulation correlations: a building that is slow to source a contractor remains slow regardless of which flood depth triggers the repair. Total re-occupancy downtime per damage state is the sum of two components:

$$t_{\text{total}} = t_{\text{impeding}} + t_{\text{repair}} \tag{4.2}$$

Impeding factors: Administrative and logistical barriers run in parallel; the longest one governs. For DS 0-2, only clean-up related sequences are considered (flood duration +

contractor or self-managed clean-up). For DS 3-4, additional sequences are added: permitting, financing (pre-arranged, private loan, or insurance), contractor mobilisation, and URM specialist lead time. All individual delays are drawn from lognormal distributions. All impeding delay distributions are taken from REDi for Floods (included in Appendix B).

Drying time: No explicit drying model is included; drying delays are implicitly captured within the clean-up and contractor impeding sequences above. However, this is a strong simplification, since for thick URM walls, high water saturation can persist for many weeks or even months, as surface drying of thick masonry walls is very slow (Matysek & Witkowski, 2026).

Repair actions: Physical repair tasks applied after all impeding delays are drawn from lognormal distributions and summed per Damage State, as listed in Table 4.6. The repair times are extracted from Dutch contractor websites.

Table 4.6: Repair time actions per DS (lognormal parameters).

DS	Repair action(s)	Median (days)	σ
DS0	None	—	—
DS1	None	—	—
DS2	None	—	—
DS3	Repointing + crack repair	4 + 3	0.2
DS4	Rebuild	8	0.2

4.3 Phase III: Vulnerability Evaluation

For each conservation scenario, the saved data from Phase II are loaded. At every flood depth increment, three quantities are computed: which components have failed (cumulatively), what the total repair cost is, and what the recovery downtime is. Results are saved for post-processing. A flowchart for this phase is included in Appendix C.

4.3.1 Damage State Evaluation

At every flood depth increment, each component is tested against its sampled or calculated flood-resistance:

- **Depth-driven components:** fail when the effective interior flood depth exceeds their depth threshold.
- **URM DS1:** fail when the effective exterior flood depth exceeds their depth threshold.
- **URM DS2-DS4:** fail when the total EUL demand exceeds the calculated structural capacity.

Failures accumulate monotonically: once a component fails at a given depth, it remains failed at all higher depths. This reflects the irreversible nature of physical damage: a cracked wall does not recover as the water recedes.

If DS1 is triggered during a flood, but the building was already in CS2 or higher, then the post-hazard damage state will be similar to the conservation state (Eq. (4.3)). A CS3 building that suffers only minor flooding will therefore still be assigned DS3, reflecting its pre-existing structural condition.

$$\text{Post-hazard DS} = \max(\text{flood DS}, \text{CS}) \tag{4.3}$$

Fragility curves are derived directly from the simulation output. For each component and each flood depth increment, the fraction of simulations in which that component has failed is recorded. This empirical failure probability is plotted as a continuous function of flood depth for the depth-driven components, yielding one curve per component.

The URM wall structural capacity is additionally visualised as a cumulative distribution function in the force domain ($\mathrm{kN/m^2}$), showing what lateral demand is required to exceed each percentile of the structural capacity distribution. This allows the demand and structural capacity to be compared directly on a common axis.

4.3.2 Financial Loss

The repair cost for each simulation is the sum of the costs of all failed components. For CS≥1 scenarios it is assumed within the logic from Eq. (4.3) that pre-existing damage must be repaired regardless of whether additional flood damage occurs.

To avoid double-counting when structural collapse occurs, DS4 (full rebuild) overrides the DS1–DS3 partial repair costs: a wall that must be rebuilt subsumes all prior damage. After this override, the total repair cost is normalised by the floor area (m^2) and by the total replacement value of the building to yield a damage ratio factor between 0 and 1. A stochastic Dutch construction cost converts the damage ratio to a monetary loss in euros.

These results are used to create vulnerability curves that show the mean repair cost (or mean damage ratio) as a function of flood depth, with 95% uncertainty bands derived from the 2.5th and 97.5th simulation percentiles. Four curves are produced, one per conservation state, enabling direct comparison of how pre-existing degradation shifts the loss curve toward lower flood depths.

4.3.3 Re-Occupancy Downtime

For each simulation at each depth, the post-hazard DS (computed above) is extracted from the pre-sampled downtime array. Because the downtime was drawn once per simulation at the start, the same contractor mobilisation delay, the same drying time, and the same impeding sequence apply at every depth for a given simulation. The result is a full matrix of downtime (days) over flood depth and simulation, preserving the statistical relationships between simulations.

Cumulative distribution functions of total downtime (weeks) for DS0-4 are plotted as well, independent of flood depth. These show the full spread of recovery times for each structural condition and are derived directly from the pre-sampled downtime arrays.

4.4 Phase IV: Extracting Location Risk

The vulnerability curves produced by Phase III span the full depth range. Phase IV narrows this down to building-specific risk by extracting the simulation output at the two flood depths associated with each case study location.

For CS0-CS3 the script extracts the damage ratio, downtime, and monetary losses from the stored results of Phase III at the specified depths and reads all simulation data for these variables. Based on the full range of simulation data, the 50th and 95th percentile are calculated and written to a csv-file. These percentiles quantify the median outcome and the 95% certainty range respectively. The data is also used to generate box-plots for all three variables.

Additionally, per conservation state, the post-hazard damage state distribution at each target depth is read separately. By plotting this as a bar chart the fraction of simulations assigned to each post-hazard DS based on the conservation state is visualised.

5
Results

5.1 Hazard Demand

Figure 5.1 presents the median equivalent uniform load (EUL) demand decomposed into hydrostatic, hydrodynamic, and debris impact components, alongside the median flow velocity, for both the first floor (FF) and second floor (SF) walls. All demands are expressed in kN/m² after the EUL transformation.

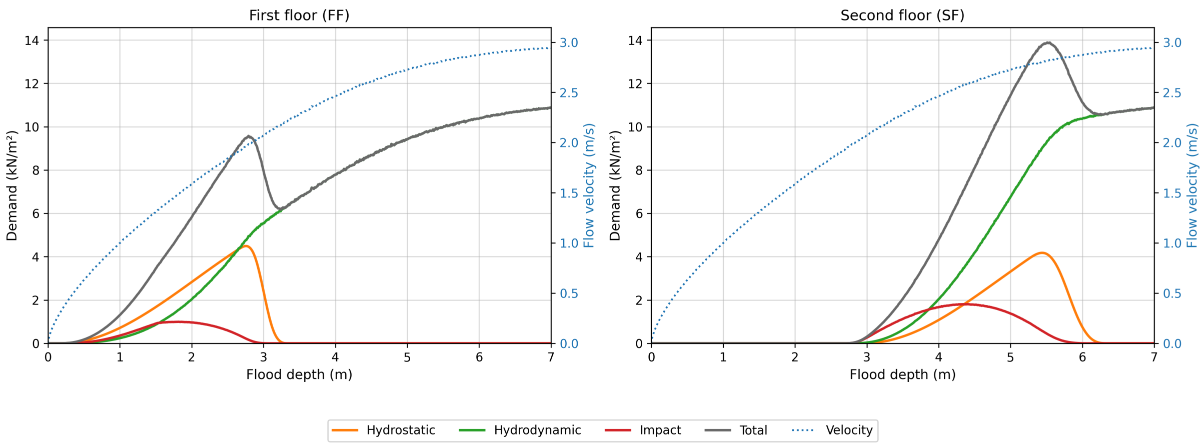


Figure 5.1: Flood demands on the URM, and flow velocity plotted over depth for the first floor and the second floor for the archetype without basement.

On the first floor, the demands start to pick up at approximately 0.3m flood depth. Underneath these flood levels, forces working on the URM walls are negligible. The hydrostatic component rises to a peak of roughly 4.2 kN/m² around 2.7 m before dropping sharply to zero. This behaviour directly reflects the piecewise differential pressure model discussed in Subsection 4.2.1.

The hydrodynamic component follows a smoother monotonic rise from approximately 0.8 m flood depth and continues to grow gradually thereafter as velocity increases. Unlike the hydrostatic force it does not drop off at the ceiling height, since it acts on whatever submerged wall area remains exposed to flow.

The debris impact component is comparatively small, peaking at approximately 1 kN/m² around 1.7 m flood depth before declining. Its bell-shaped profile reflects the interplay between increasing velocity and the height on the wall at which the debris impact has the biggest effect (midway). At very shallow and very deep depths relative to the wall panel the contribution is minimal.

The total demand on the ground floor wall peaks sharply at approximately 9.7 kN/m² around 2.8 m of flood depth before dropping to around 6 kN/m² at 3.2 m as the hydrostatic and debris impact forces zeros out. From there on the demands are solely governed by the hydrodynamic forces.

Demand distributions on the second floor panel (right in Figure 5.1) have similar characteristics, but the hydrodynamic and debris impact forces are higher due to higher mean flow velocity.

5.2 Archetype Vulnerability

5.2.1 Depth-Driven Component Fragilities

To verify the model was working as intended, the data provided by Nofal et al. (2020) was used to replicate their results. Afterwards, the model was run on the new component library (see Appendix A). The computed fragility curves, based on the dataset without and with a basement are displayed in Figure 5.2 and Figure 5.3 respectively.

The figures presents the sampled fragility curves for all components, excluding the URM facade wall panels, across the first floor, second floor, and attic, plotted as failure probability against absolute flood depth. Each curve represents one component's flood depth-resistance distribution sampled from the component dataset, such that the 50th percentile of each curve corresponds to the mean resistance-depth and the spread reflects the component's sampled standard deviation.

When comparing the two figures, it is possible to analyse how the basement influences flood-resistance. The most prominent difference is the addition of a cluster of components in the 0-1 m depth range. These correspond to the basement-level components (e.g. concrete floor, exterior walls, windows, internal insulation, closets). Their fragility curves activate earliest of all components in the inventory, reflecting that the basement sits below the finished floor elevation and largely below ground, and is therefore the first space to be inundated.

The first floor, second floor, and attic clusters are identical between the two figures, as expected. They only manifest at higher inundation levels, due to the increased finished floor elevation.

In terms of implication for the loss model, the basement archetype will produce higher repair costs at shallower flood depths than the no-basement case, since basement components begin failing immediately at any positive inundation depth. This will steepen the onset of the vulnerability curve at low depths and increase total expected losses across all scenarios where the flood depth exceeds the basement floor level.

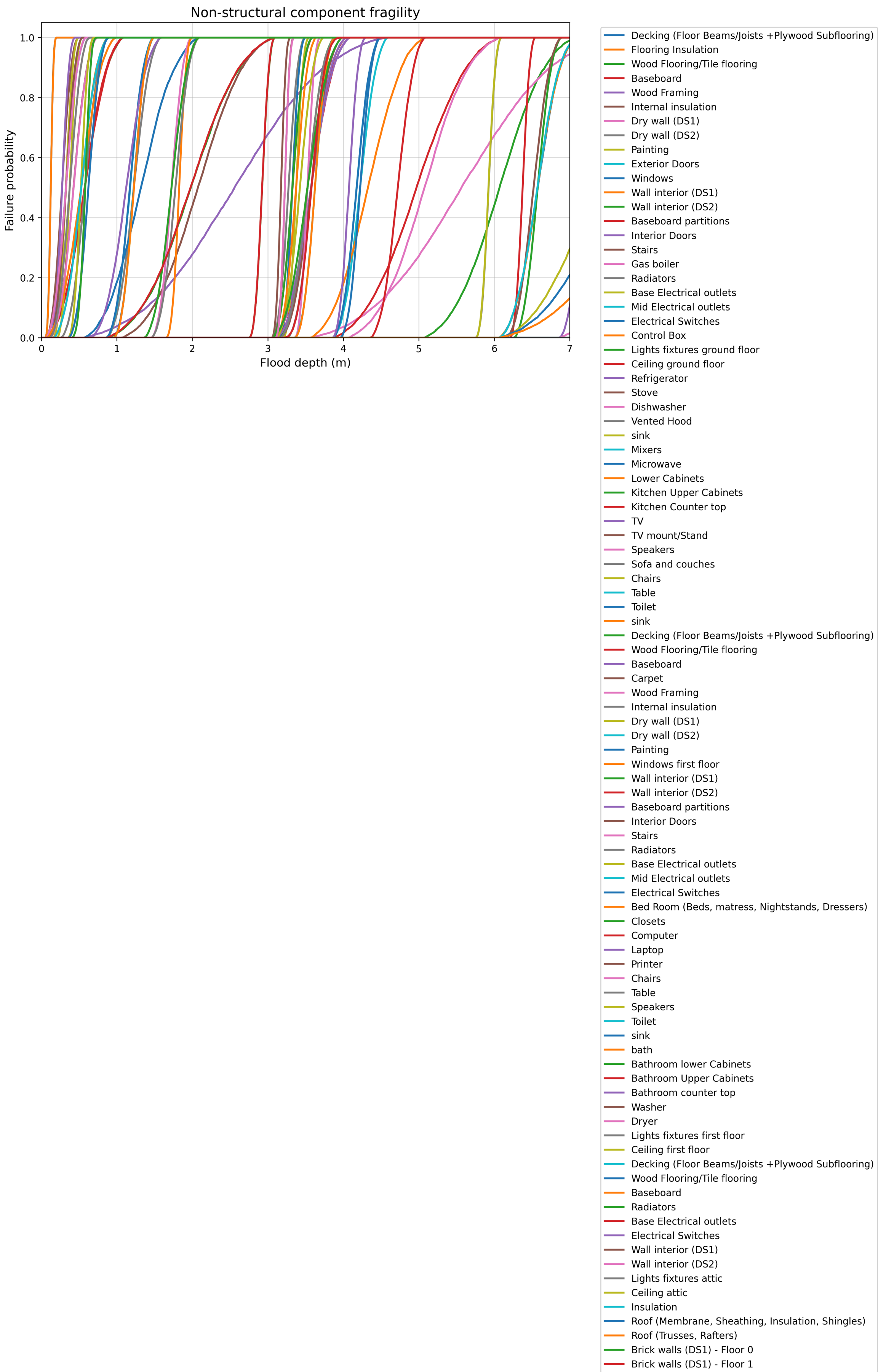


Figure 5.2: Depth-driven components' fragility curves for the archetype without the basement.

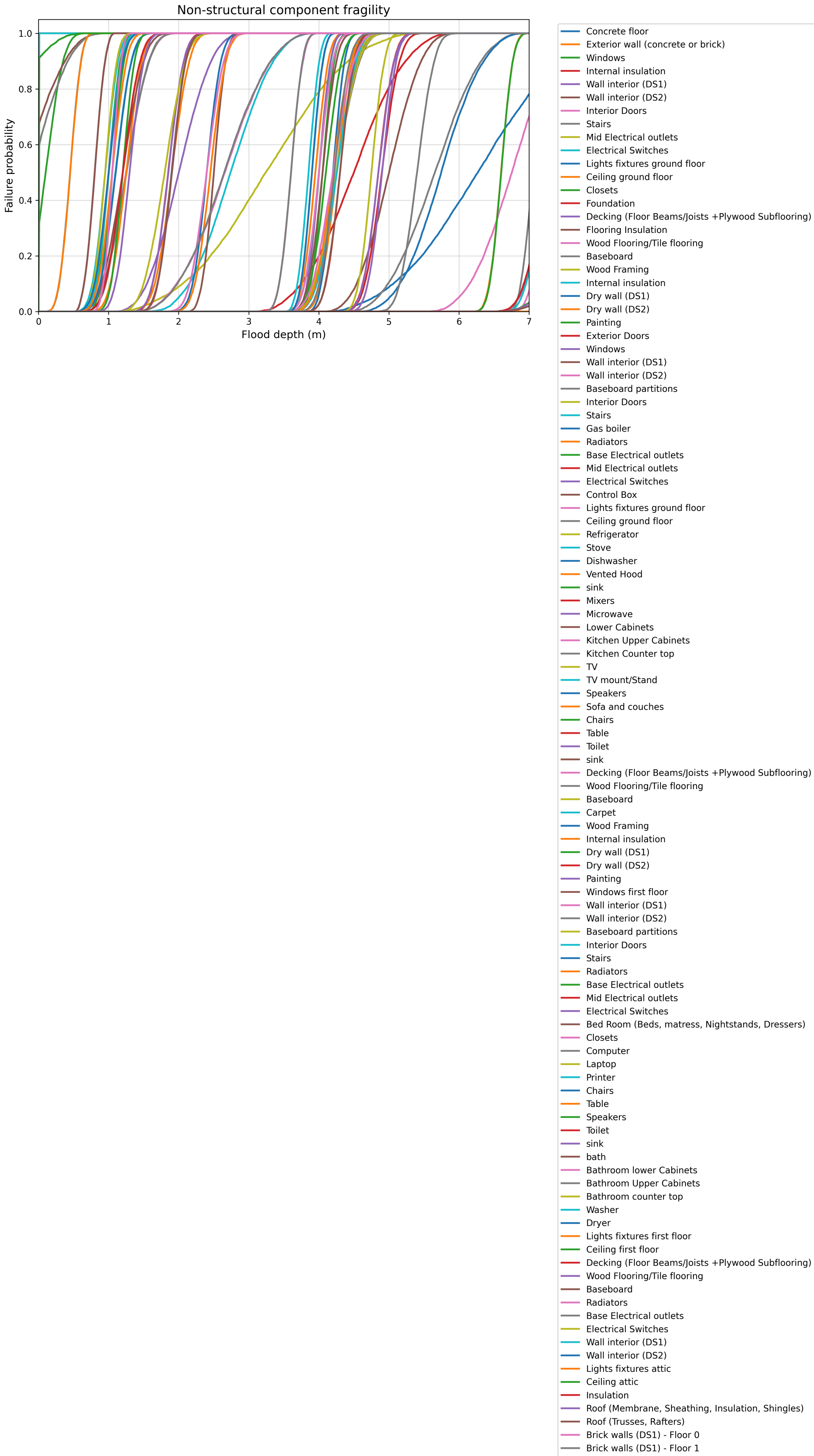


Figure 5.3: Depth-driven components' fragility curves for the archetype with the basement.

5.2.2 URM fragility

Based on the sampled geometry parameters (included in Appendix B), state-dependent fragility curves can be plotted for the URM wall panel. The curves in Figure 5.4 present the probability of DS4 failure given an equivalent uniform load and a conservation state.

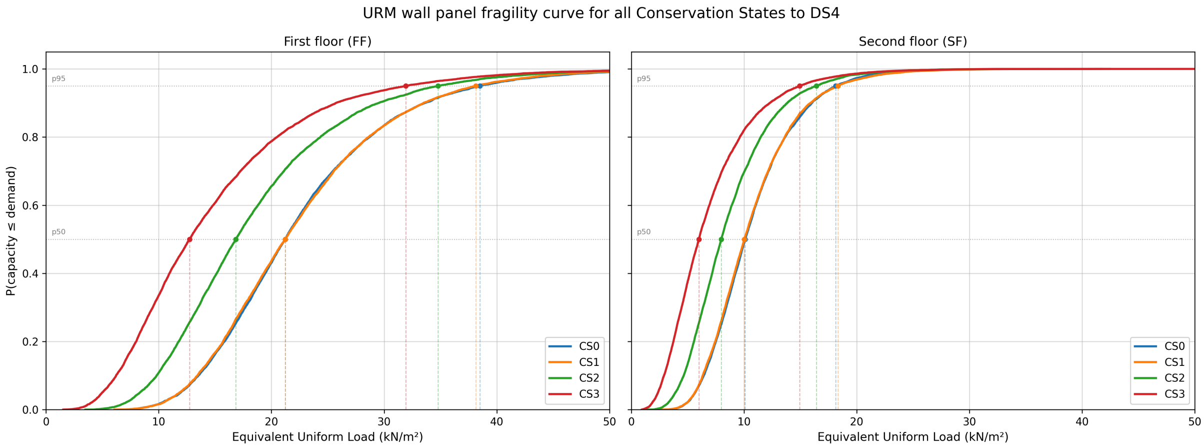


Figure 5.4: URM fragility curves for CS0-3 to DS4.

Extracting the p50 and p95 percentile values from Figure 5.4 gives Table 5.1.

Table 5.1: Model results of double wythe URM wall lateral structural capacity per conservation state for DS4 (collapse). First floor uses 105 kPa pre-compression; second floor uses 50 kPa.

CS	First Floor (105 kPa)		Second Floor (50 kPa)	
	p50 (kN/m²)	p95 (kN/m²)	p50 (kN/m²)	p95 (kN/m²)
0	21.23	38.51	10.10	18.14
1	21.23	38.15	10.03	18.34
2	16.84	34.78	7.98	16.43
3	12.73	31.93	5.99	14.94

The capacities in Table 5.1 are significantly higher than the capacities presented in the table for the validation of the methodology (see Table 4.5), which considered a single leaf wall. For the first floor the threshold demand of a wall panel in CS0 to reach DS4 goes from 5.79 kN/m² to 21.23 kN/m². This is perfectly in line with the quadratic scaling of the structural capacity as discussed in Section 3.3.1.2, since the wall thickness doubles when moving from a single leaf wall to a double wythe wall.

Focussing solely on Table 5.1, the first floor consistently shows roughly double the structural capacity of the second floor across all CS: the ratio increase in the median value (p50) is approximately 2.1 across CS0 through CS3. This is directly attributable to the difference in axial pre-compression and the linear relationship between pre-compression and structural capacity.

CS0 and CS1 produce nearly identical median capacities on both floors, indicating that the transition from pristine to lightly degraded condition has negligible effect on median structural capacity. This is consistent with the model formulation, where CS1 introduces no structural capacity reduction factor. The structural capacity reduction becomes meaningful from CS2 onward. The ground floor median structural capacity shows a 21% reduction for CS2 and a 40% reduction for CS3, relative to CS0.

The spread between p50 and p95 widens progressively with deterioration. On the ground floor, the p95-p50 gap grows from 17.28 kN/m² at CS0 to 19.20 kN/m² at CS3. On the first floor, it grows from 8.04 to 8.95 kN/m². This reflects the increasing model uncertainty assigned to higher conservation states. As the wall condition deteriorates, the simulation samples from a wider structural capacity distribution, producing a heavier upper tail. Importantly, the p95 values decrease much more slowly than the medians across CS, meaning that a deteriorated wall retains a non-negligible probability of exhibiting near-intact structural capacity, which is physically plausible given the heterogeneous nature of masonry degradation in practice.

Next, the 95% variance bounds of the demands in a 2-meter flood scenario where plotted over the state-dependent URM fragility curves of the first floor wall panel to create an understanding of what this structural capacity means for a severe flooding scenario (Figure 5.5). The figure shows that expected demands under the boundary conditions of this study rarely lead to failure of the URM wall. The effect of the conservation state, is very clear. While CS0 and CS1 have close to no chance of failure, the probability increases slightly for CS2 and more severely for CS3. Quantitative analysis of this increase follows in Section 5.4.

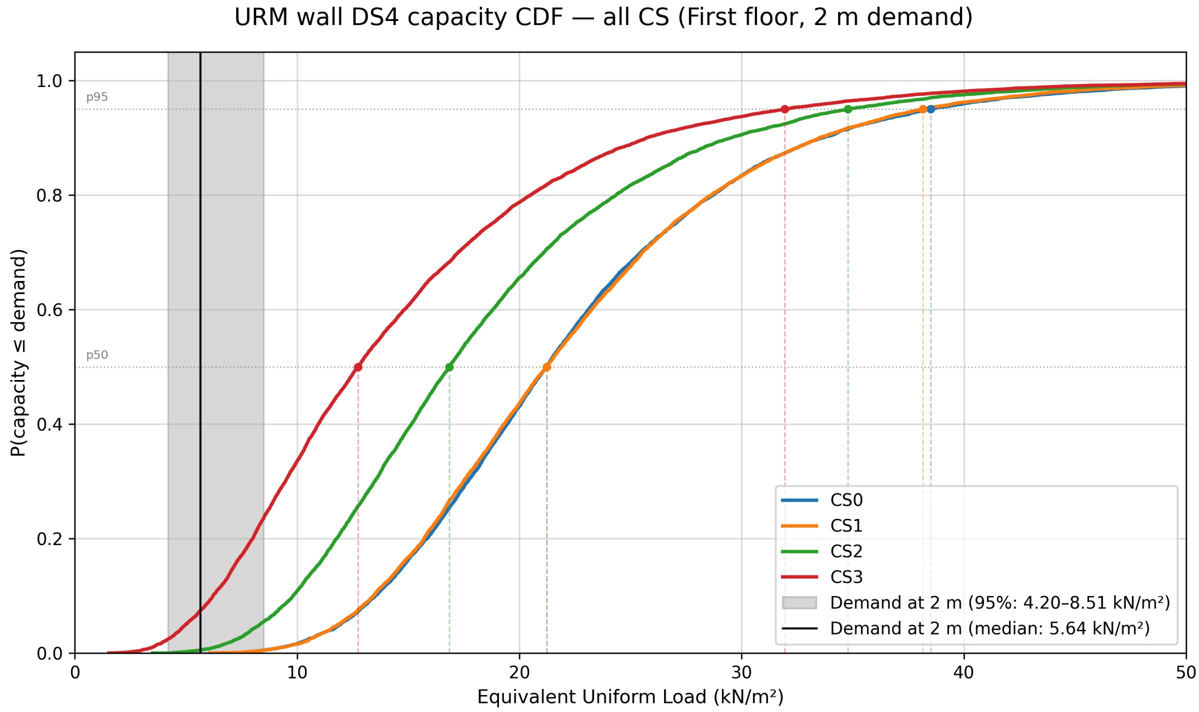


Figure 5.5: URM fragility curves (first floor) for CS0-3 to DS4, showing the 95%-variance and median demands of a flood with a 2m flood depth.

5.2.3 Vulnerability Assessment

5.2.3.1 Repair Costs

Moving on to the vulnerability assessment of the archetype, combining depth-driven components with force-driven components. Figure 5.6 presents mean vulnerability curves with 95% confidence bounds for all four conservation states, plotted against absolute flood depth up to 7 m. The damage ratio expresses total repair cost as a fraction of the total building replacement value.

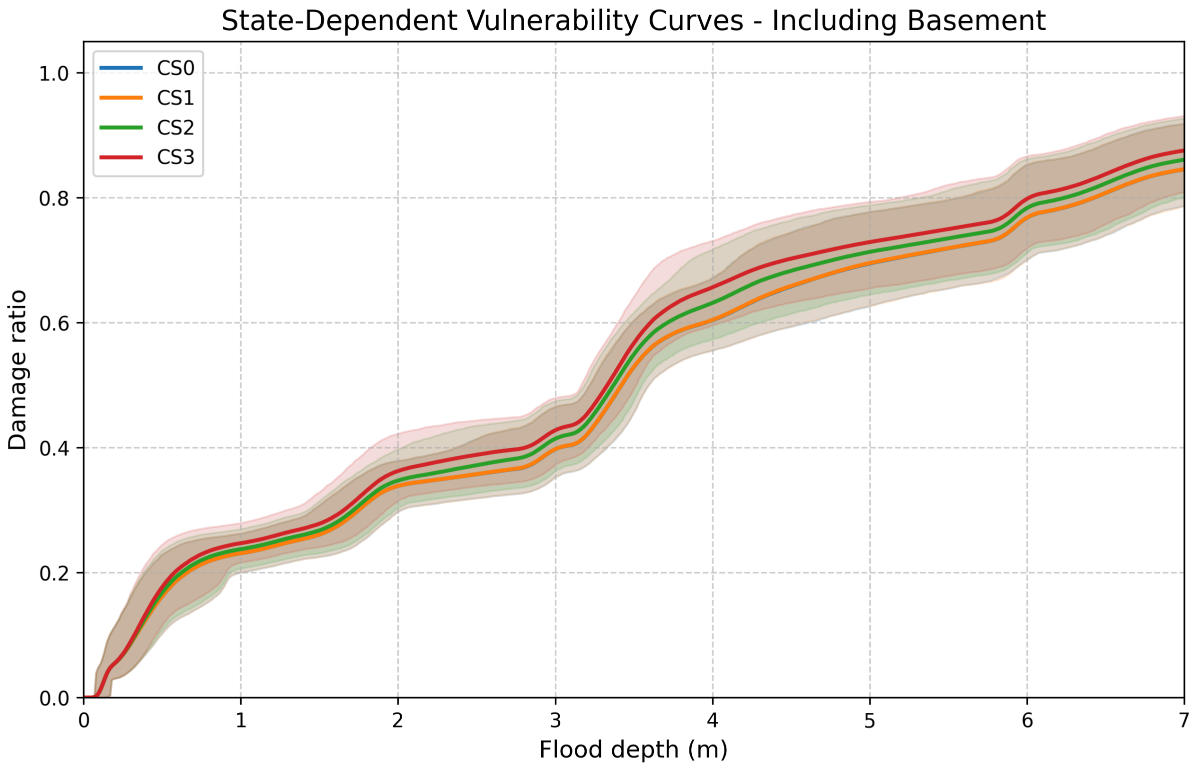


Figure 5.6: Archetype (without basement) conservation state-dependent vulnerability curves with 95% confidence bounds for all conservation states, plotted against absolute flood depth

The curves in Figure 5.6 share a characteristic two-step shape, reflecting the floor-by-floor component failure depths discussed in Subsection 5.2.1. CS0 and CS1 are nearly indistinguishable throughout the full depth range, consistent with the near-identical median capacities reported in the table discussed earlier. Meaningful separation begins at CS2 and is most pronounced for CS3, particularly in the 2-7 m range where the curves diverge by approximately 5-8 percentage points at any given depth.

The separation can be explained by two mechanisms. First, walls in CS2 and CS3 have lower out-of-plane lateral capacity, making structural failure and therefore the associated higher DS repair costs more likely at any given flood depth. Second, and more crucially, the model assumes that walls in CS1-CS3 require repair whenever DS1 is triggered by flood depth exceedance alone, regardless of whether a mechanical demand failure occurs. This means that even at shallow depths where lateral forces are negligible, a deteriorated wall already incurs repair costs simply by being inundated, widening the damage ratio gap relative to CS0 from early depths onward.

Considering that even significant flood depths do not structurally challenge the URM wall (as discussed based on Figure 5.5), it can be said that the CS separation visible in Figure 5.6 at depths at least up until 2 m is driven almost entirely by the depth-triggered repair assumption, not by structural failures. The divergence in damage ratio between CS0 and CS3 is essentially an accounting effect: deteriorated walls are being repaired for flood contact damage regardless of whether the structure was mechanically threatened.

Lastly, The 95% variance bounds are notably wide, spanning roughly ±0.15 around the mean damage ratio at depths above 3 m. This reflects the compounding uncertainty from the component cost distributions. The overlap between the CS bounds across the full depth range, but especially at depths < 0.5m, is substantial. This indicates that while the mean curves are meaningfully separated, individual realisations across conservation states are difficult to distinguish. Therefore, a single flood event on a building of unknown CS would produce loss estimates consistent with multiple scenarios. On top of that, a building in CS3 could therefore receive lower total repair costs then a building in CS0, dependent on the other components costs.

The state-dependent vulnerability curve for the archetype with a basement is presented in Figure 5.7.

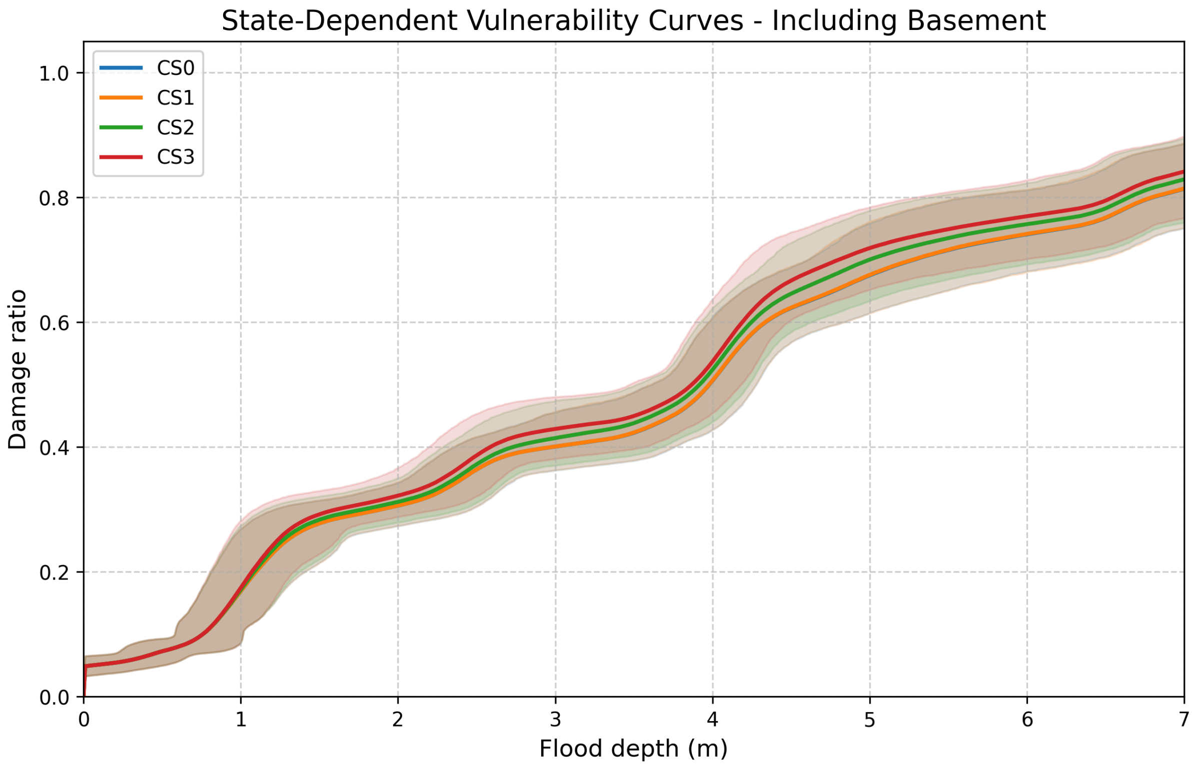


Figure 5.7: Archetype (with basement) conservation state-dependent vulnerability curves with 95% confidence bounds for all conservation states, plotted against absolute flood depth

The most immediately apparent difference is the non-zero damage ratio at just above zero flood depth. This change reflects the basement component inventory which are permanently below the finished floor elevation and therefore trigger repair costs the moment any inundation is recorded, even before floodwater reaches the ground floor. This is due to the assumption that flood water will always find a way into the building.

After the flood reaches the first floor, the curves are similar to Figure 5.6, as illustrated in Figure 5.8. The basement scenario produces a more pessimistic vulnerability function at all depths, but particularly at shallow inundation where the damage ratio is already elevated before any ground floor components are affected. For risk assessment purposes this means that even a minor flood event (i.e. lower than the first floor elevation) generates a non-trivial repair cost in the basement scenario. However, this is independent of the conservation state of the URM.

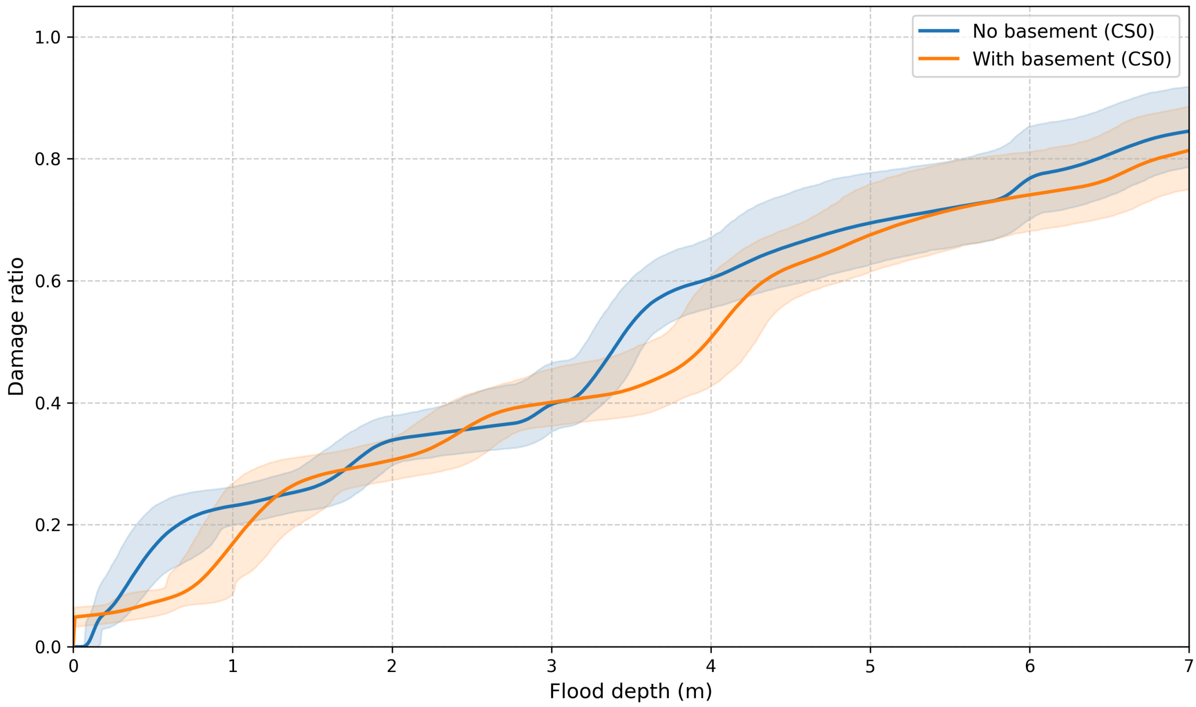


Figure 5.8: Comparison of vulnerability curves for CS0 between the archetype with and without basement.

5.2.3.2 Downtime

Figure 5.9 presents the modelled cumulative distribution of repair downtime per post-hazard damage state. The distributions span roughly three orders of magnitude considering their median (p50), reflecting a non-linear relationship between structural damage level and associated displacement duration.

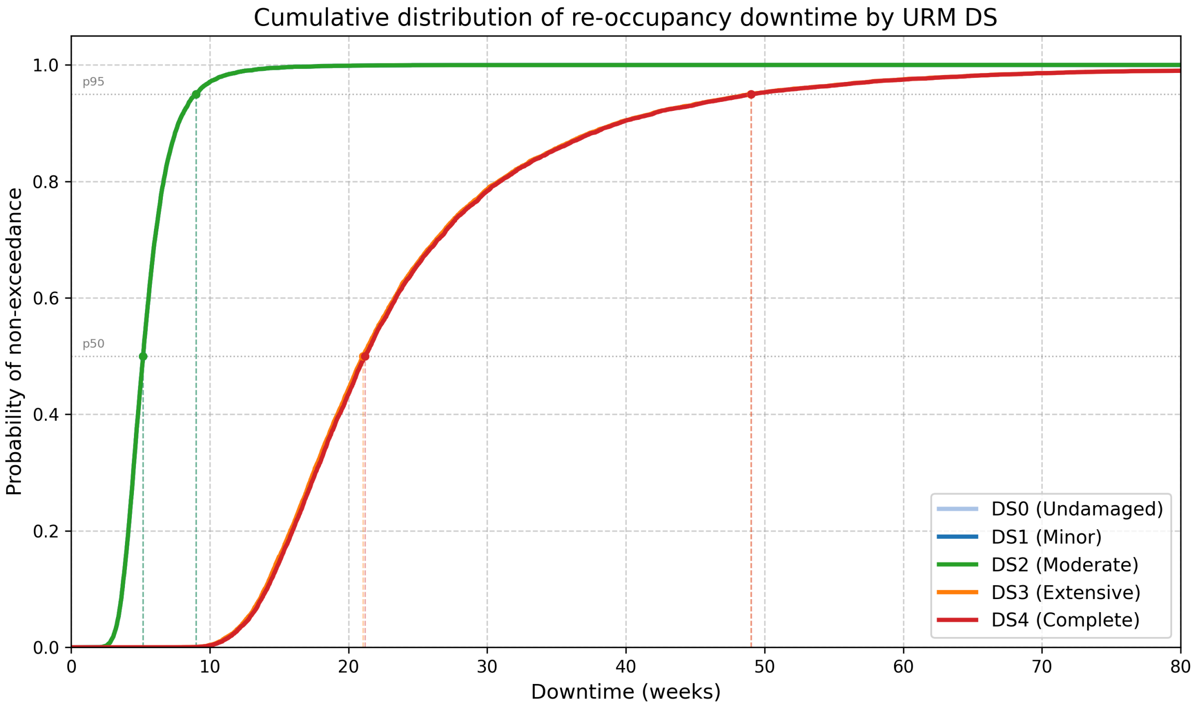


Figure 5.9: Probability of non-exceedance plotted over the re-occupancy downtime, for damage states DS0-4.

DS0, DS1, and DS2 produce identical distributions. The overlap is expected, since the impeding sequences for these damage states share the same recovery path, and the additional repair time for DS1 and DS2 to reach re-occupancy safety(surface cleaning, minor component replacement) is negligible relative to the dominant impeding delays.

DS3 and DS4 also have similar re-occupancy downtime. Within the model's current parameterisation, transitioning from extensive damage to collapse does not substantially extend the median repair timeline. Damage in these states is severe enough to require repair before the building is considered structurally safe. This activates the more extensive impeding sequence, including structural assessment, permit processing, and contractor procurement for major reconstruction — each of which carries substantial uncertainty. As a result, the dispersion of downtime for DS3 and DS4 is broad and heavy-tailed. Downtime can extend to over a year in scenarios where multiple impeding factors compound.

The clear separation between DS0-2 and DS3-4 underscores that the critical threshold for long-term displacement is collapse-level damage: buildings that avoid DS3 are expected to be reoccupied within roughly two months, while those reaching DS3 or DS4 face displacement periods that may exceed a year in adverse cases.

5.3 Mitigation

In this section, the effectiveness of the three mitigation strategies are compared and discussed in general and specifically for both case studies. The quantitative effectiveness of each mitigation strategy is evaluated by comparing the mean damage ratio (DR_0) and mean repair downtime (DT_0, in weeks) of the unmitigated baseline at specified flood depths against three intervention scenarios. Baseline values are presented as absolute numbers; strategy results are expressed as percentage changes relative to the baseline, where negative values indicate a reduction in expected loss or recovery time. Three strategies are compared:

- **Strategy 1** (ΔDR_1, ΔDT_1): Structural restoration — repair of URM walls from the assessed CS back to pristine condition (CS0).
- **Strategy 2** (ΔDR_2, ΔDT_2): Component adaptation — replacement of high-risk non-structural components with flood-resilient alternatives or increased elevation.
- **Strategy 3** (ΔDR_3, ΔDT_3): Combined intervention — both strategies applied simultaneously.

Results are tabulated at 0.5 m depth increments from 0.5 m to 7.0 m. Full results for both archetypes across all Conservation States are provided in Appendix E.

To illustrate how these results can be interpreted, Table 5.2 presents results for the archetype with a basement at CS2. In this scenario, the URM structural capacity reduction due to deterioration is significant and basement-level component exposure amplifies damage at low flood depths. The results show that restoring the URM has a small effect on the total building damage, almost entirely independent of depth, with a maximum reduction in damage ratio of 3.6%. A more effective mitigation measure to reduce repair costs is to strategy 2. However, strategy 2 has close minimal impact on downtime, while within the depth range of 2.5m to 5.0m the effect of URM repair on re-occupancy downtime is significant, with a maximum of 40.7% decrease compared to the unmitigated building.

Table 5.2: Mitigation effectiveness — with-basement archetype, CS2. DR_0 and DT_0 are mean absolute values; ΔDR and ΔDT are percentage changes relative to the baseline.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (weeks)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.0726	5.57	0.0	0.0	−18.9	0.0	−18.9	0.0
1.0	0.1712	5.57	−1.3	0.0	−16.4	0.0	−17.8	0.0
1.5	0.2835	5.57	−2.1	0.0	−18.9	0.0	−21.2	0.0
2.0	0.3121	5.63	−1.9	−1.0	−19.3	0.6	−21.3	−1.0
2.5	0.3723	7.55	−2.2	−23.5	−22.8	1.1	−25.2	−23.4
3.0	0.4143	13.40	−3.3	−40.7	−16.0	−0.9	−19.5	−40.3
3.5	0.4383	18.01	−3.5	−35.3	−14.3	0.1	−18.1	−35.1
4.0	0.5238	18.63	−3.4	−34.2	−10.7	−0.2	−14.3	−34.0
4.5	0.6457	20.13	−3.5	−35.6	−7.2	−0.2	−11.0	−35.0
5.0	0.7004	23.51	−3.6	−19.5	−6.0	−0.2	−9.8	−19.8
5.5	0.7347	24.71	−2.6	−4.7	−5.3	0.0	−8.1	−4.7
6.0	0.7571	24.88	−2.1	−1.1	−4.9	0.0	−7.2	−1.1
6.5	0.7824	24.92	−2.0	−0.7	−4.4	0.0	−6.5	−0.7
7.0	0.8289	24.92	−1.8	−0.7	−3.8	0.0	−5.7	−0.7

Next, Figure 5.10 shows the vulnerability curves for the same archetype at the most severely deteriorated state (CS3), where the visual separation between strategies is largest. At low flood depths, where the URM is not reached by the flood water, only component mitigation is an effective strategy to reduce repair costs. However, as depths increase, URM repair starts to become effective as well. For depths of 4 to 8 meters, the effect of URM repair and component mitigation is similar. At even higher depths, the component mitigation is not effective anymore, and only URM repair is a helpful mitigation strategy. However, these flood depths will likely never occur in Dutch flooding scenarios. For this specific conservation state, the highest damage ratio reduction is reached with a combined strategy between circa 1.3 and 3.8 meters.

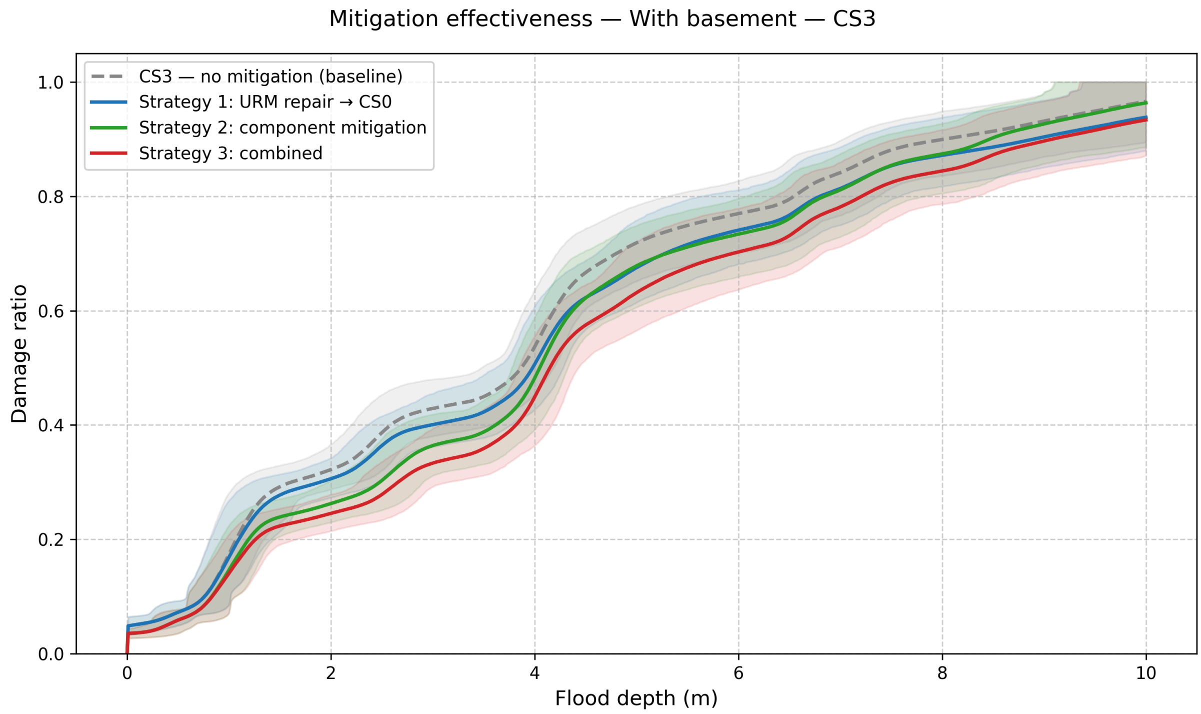


Figure 5.10: Vulnerability curves — with-basement archetype, CS3. The gray dashed line is the unmitigated baseline; colored curves show Strategies 1-3. Shaded bands are 95% confidence intervals.

By analysing the results, effective strategies per conservation state and flood depth can be determined for both archetypes (i.e. with and without basement). These results are presented in Table 5.3 and Table 5.4, where 1 refers to URM repair; 2 refers to component mitigation; 3 refers to a combined approach, since both strategies are individually effective. A strategy was considered effective, if the relative decrease in damage ratio (DR) or downtime (DT) was over 5%. This is an assumption, since the actual effectiveness is dependent on uncertainties and the definition of effectiveness considered by the stake-holder.

Table 5.3: Effective mitigation strategies – no basement.

Depth (m)	CS1		CS2		CS3	
	DR	DT	DR	DT	DR	DT
0.5	2	-	2	-	3	1
1.0	2	-	2	-	3	1
1.5	2	-	2	-	3	1
2.0	2	-	2	1	3	1
2.5	2	-	2	1	3	1
3.0	2	-	2	1	3	1
3.5	2	-	2	1	3	1
4.0	2	-	2	1	3	1
4.5	2	-	2	1	3	1
5.0	2	-	2	-	2	-
5.5	2	-	-	-	-	-
6.0	-	-	-	-	-	-
6.5	-	-	-	-	-	-
7.0	-	-	-	-	-	-

Table 5.4: Effective mitigation strategies – with basement.

Depth (m)	CS1		CS2		CS3	
	DR	DT	DR	DT	DR	DT
0.5	2	-	2	-	2	-
1.0	2	-	2	-	2	1
1.5	2	-	2	-	3	1
2.0	2	-	2	-	3	1
2.5	2	-	2	1	3	1
3.0	2	-	2	1	3	1
3.5	2	-	2	1	3	1
4.0	2	-	2	1	3	1
4.5	2	-	2	1	3	1
5.0	2	-	2	1	3	1
5.5	2	-	2	-	2	1
6.0	2	-	-	-	-	-
6.5	-	-	-	-	-	-
7.0	-	-	-	-	-	-

Analysing the two tables shows that component mitigation is an effective strategy up until very severe flood depths of 5.0 to 6.0 meters, dependent on the conservation state and presence of a basement. URM repair is not effective for CS1, but has a considerable effect in terms of downtime reduction for CS2 and CS3. Notably, in CS2 this is only true for a flood depth between a specified range (2.0-4.5m and 2.5-5.0m for a building without and with a basement respectively), while in CS3 this is true even for low inundation depths. Additionally, in CS3 URM repair positively affects damage accumulation as well. In absence of a basement this is true for flood levels up until 4.5 meters. Otherwise, this rings true between a depth of 1.5-5.0 meters.

5.4 Case Study Risk

In this section the risk data for the two selected case study locations and flood scenarios are presented. For both locations, the losses in terms of repair costs and re-occupancy downtime are calculated at the specified flood depths, dependent on the conservation state. To analyse the results, the losses are presented both in the form of box plots and tables.

After the initial scenario-based flood risk assessment, mitigation strategies are applied as discussed in the previous section. The relative changes in losses are presented in tables as well. For readability, not all figures and tables are included in this section, but they can be found in Section EB.

5.4.1 Location 1: Kamperfoeliestraat 71

Location 1 shows negligible differences in losses for CS0, CS1, and CS2. None of the simulations showed damage progression in the URM, with an exception to a few wall panels in CS3 during the high inundation scenario. This is visible in the slight increase of damage ratio and repair costs (Table 5.5) across the different conservation states, and in the bar chart on the top right of Figure 5.11 which shows the fraction of simulations in which damage state DS1-4 was reached, based on initial condition CS0-3. The small red line on the top of the chart of CS3, shows that a few simulations reached DS4. All other simulations reached DS3. The only notable difference, is the change in downtime for CS3, since the flooding is assumed to trigger repair before re-occupancy can happen.

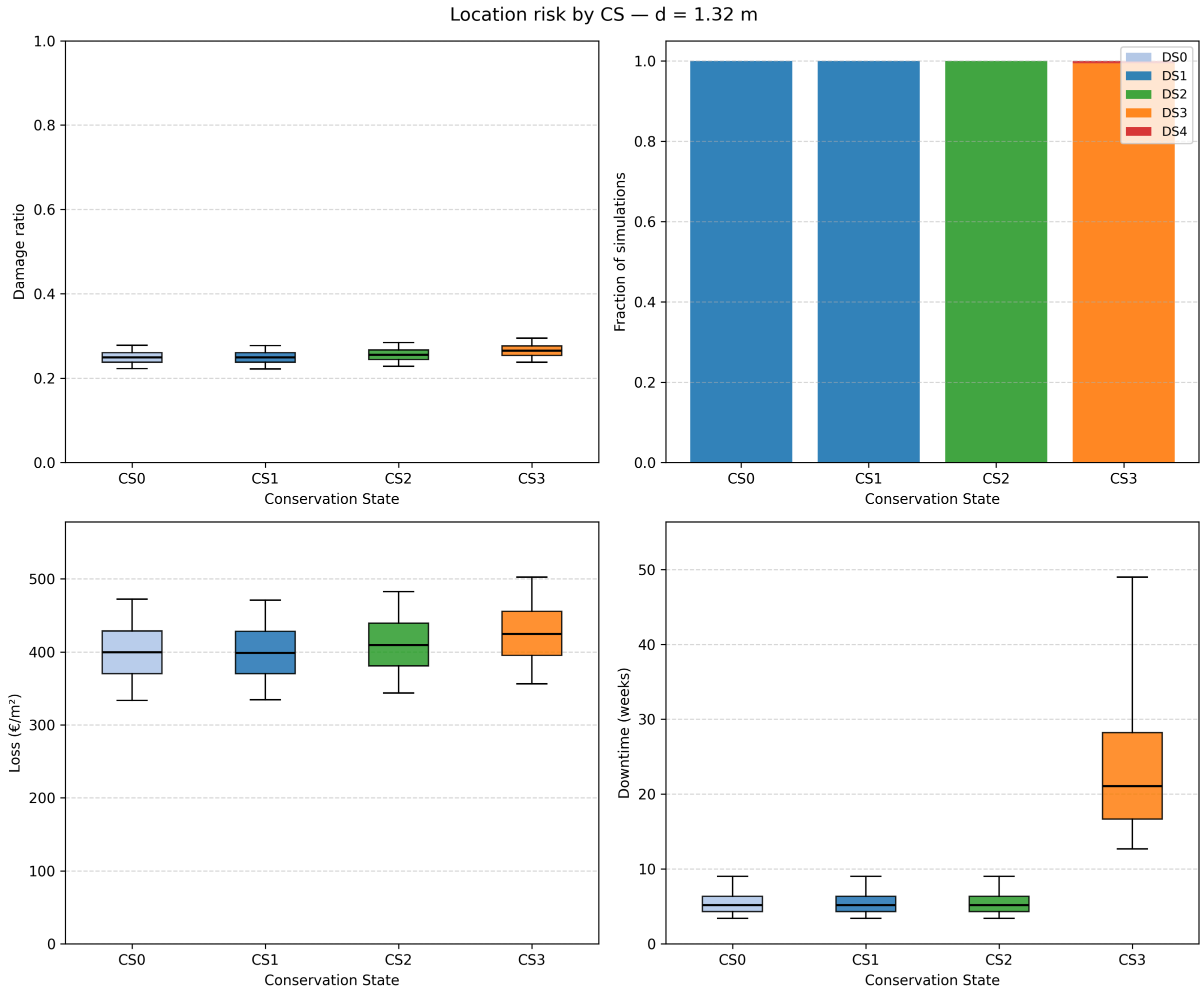


Figure 5.11: Overview of risk metrics at the Kamperfoeliestraat (location 1) for fluvial/coastal flooding (unmitigated).

Table 5.5: State-dependent absolute losses for two flooding scenarios at location 1 (unmitigated). For every variabel two values are given in the form of x(y): median (95th percentile).

CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	**Downtime (weeks)**	**Repair Costs (€/m2)**	**Damage Ratio**	**Downtime (weeks)**	**Repair Costs (€/m2)**
0	0.080 (0.15)	5.2 (9.0)	128 (240)	0.25 (0.28)	5.2 (9.0)	400 (473)
1	0.080 (0.15)	5.2 (9.0)	129 (239)	0.25 (0.28)	5.2 (9.0)	399 (471)
2	0.082 (0.15)	5.2 (9.0)	131 (241)	0.26 (0.28)	5.2 (9.0)	409 (482)
3	0.084 (0.15)	21 (49)	134 (247)	0.27 (0.29)	21 (49)	425 (503)

Reading effective mitigation strategies from Table 5.3 shows that for both scenarios component mitigation is effective for damage ratio and thus repair cost reduction when in CS1-3.

Considering URM repair, in the pluvial scenario with a depth of less than 0.5 meters, repair is only beneficial for a building in CS3, both in terms of downtime and damage ratio. For the severe flooding scenario with a depth between 2.0-2.5 meters, URM repair is effective for reducing re-occupancy downtime in CS2 too as well, but it has little effect on the mean damage ratio. The relative changes in losses are presented in Table 5.6.

Table 5.6: Relative changes in losses for two flooding scenarios at location 1 after applying mitigation strategies. For every variabel two values are given in the form of x(y): median (95th percentile).

Original CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)
1	−13.8 (-29.0)	0.0 (0.0)	−13.8 (-25.5)	−21.6 (-20.1)	0.0 (0.0)	−21.6 (-20.6)
2	−13.7 (-25.3)	0.0 (0.0)	−13.6 (-24.3)	−23.6 (-22.2)	0.0 (0.0)	−23.8 (-22.4)
3	−17.3 (-27.9)	−75.3 (-81.6)	−18.9 (-28.0)	−26.4 (-24.9)	−75.3 (-81.6)	−26.5 (-25.5)

With a floor area of 67m^2, for the low inundation and high inundation scenario this translates to approximately €7281 or €29,929 decrease in repair costs respectively. Both scenario result in 16 weeks less downtime on average.

5.4.2 Location 2: ‘s Gravendijkwal 56

Risk assessment for location 2 (‘s Gravendijkwal 56), show that as a result of the high finished floor due to the presence of the basement the URM conservation state is irrelevant for the pluvial flooding scenario (Table 5.7). On the other hand, the due to the higher water depth in the high inundation scenario, the effect of conservation state is larger than in the first case study location. Figure 5.12 shows in the top-right bar chart how more wall from CS2 and CS3 reach higher damage states during the flooding, compared to the conservation state, indicated by the red fraction. While no simulations from CS0 and CS1 reached DS4, this did happen for a number of simulation in CS2 and CS3, indicating the increased probability of catastrophic failure due to poor maintenance and neglect.

Table 5.7: State-dependent absolute losses for two flooding scenarios at location 2 (unmitigated). For every variabel two values are given in the form of x(y): median (95th percentile).

CS	Pluvial Flooding (0.20m)			Fluvial/Coastal Flooding (2.18m)		
	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)
0	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.36)	5.2 (9.0)	509 (606)
1	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.36)	5.2 (9.0)	509 (605)
2	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.37)	5.2 (9.7)	519 (617)
3	0.055 (0.067)	5.2 (9.0)	87 (111)	0.33 (0.38)	21 (49)	538 (644)

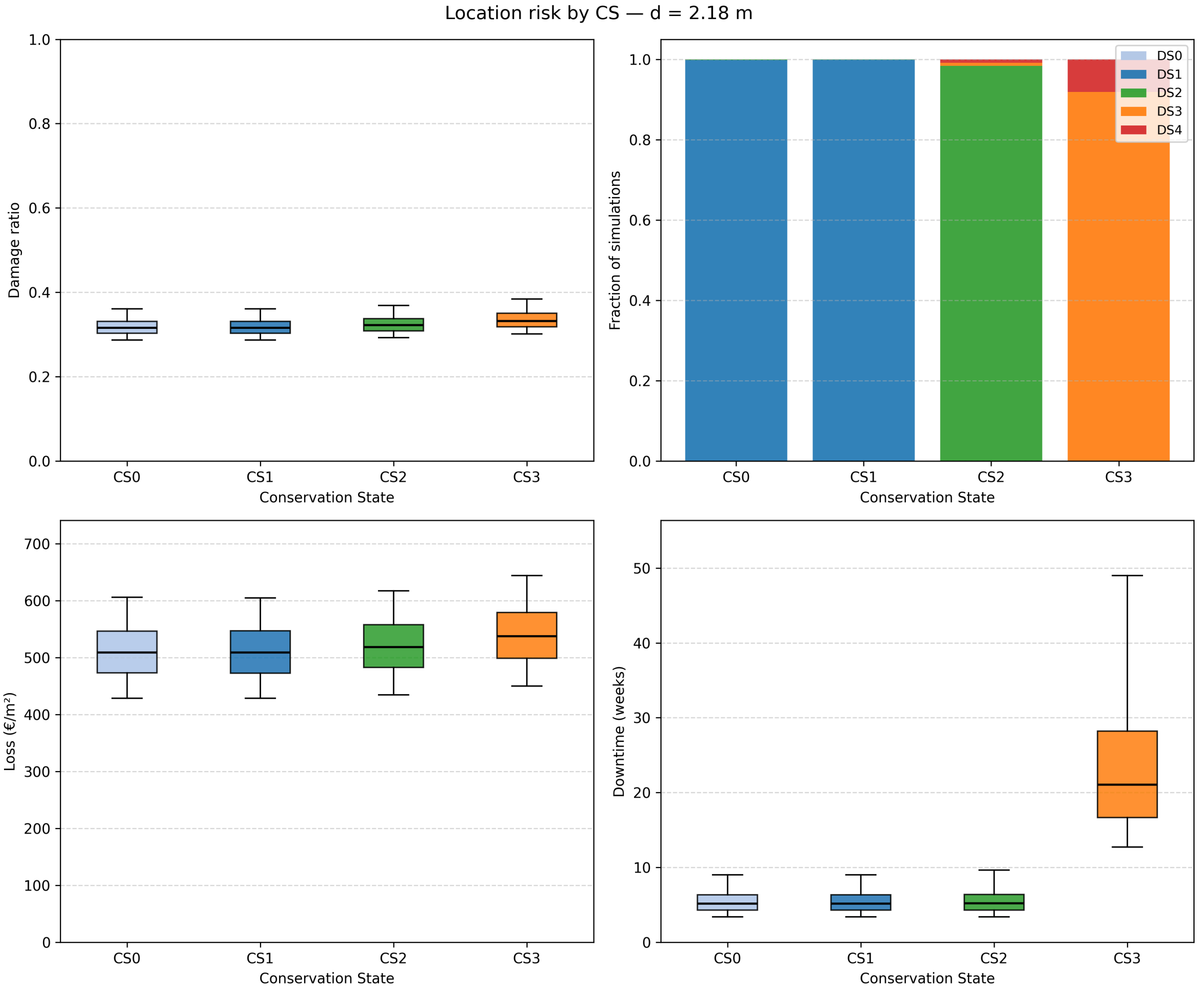


Figure 5.12: Vulnerability curves — with-basement archetype, CS3. The gray dashed line is the unmitigated baseline; colored curves show Strategies 1-3. Shaded bands are 95% confidence intervals.

Reading effective mitigation strategies from Table 5.4 shows that component mitigation is effective for damage ratio and thus repair cost reduction when in both flooding scenarios and all conservation states. URM repair is only beneficial in terms of damage ratio during the severe flooding scenario with a depth between 2.0-2.5 meters and if the building is in CS3, and possibly CS2. In this severe scenario, URM repair is effective for reducing re-occupancy downtime in CS2 and CS3. Applying these mitigation strategies leads to relative decrease in damage ratio, costs, and re-occupancy downtime as presented in Table 5.8.

Table 5.8: Relative changes in losses for two flooding scenarios at location 2 after applying mitigation strategies. For every variabel two values are given in the form of x(y): median (95th percentile).

Original CS	Pluvial Flooding (0.20m)			Fluvial/Coastal Flooding (2.18m)		
	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)
1	−31.0 (-25.1)	0.0 (0.0)	−30.1 (-25.4)	−19.9 (0.36)	0.0 (0.0)	−20.3 (-20.8)
2	−30.9 (-25.1)	0.0 (0.0)	−29.9 (-25.7)	−21.4 (0.37)	−0.4 (-6.7)	−21.7 (-22.5)
3	−30.9 (-25.1)	0.0 (0.0)	−30.3 (-26.0)	−23.9 (0.38)	−75.3 (-81.6)	−24.4 (-25.8)

With a floor area of 327m^2, for the low inundation and high inundation scenario this translates to approximately €19,829 or €130,537 decrease in repair costs respectively. In terms of downtime, the low inundation scenario shows no decrease, but the high inundation scenario leads to 16 weeks less downtime on average if properly mitigated.

6 Discussion & Conclusion

This research investigated the integration of pre-flood structural conservation states into probabilistic FRAs to establish a more accurate, building-specific representation of vulnerability. By developing a force-driven computational engine that accounts for pre-existing structural damage manifestations rather than relying on empirical depth-damage assumptions, this thesis provides answers to the underlying research questions posed in Chapter 1. First, the results are discussed based on their applicability and limitations. Afterwards, the research questions are answered based on the results presented in the previous chapter, and future research directions are mentioned.

6.1 Discussion

6.1.1 Applicability

The methodology and findings developed in this thesis have distinct implications for both the practical execution of regional flood risk management and the academic approach to multiscale structural vulnerability modelling. By establishing a physics-based, state-dependent framework, this work directly addresses the existing gap between large-scale regional FRAs and highly localized structural engineering assessments.

Academically, future computational design and performance-based engineering research can leverage this framework to simulate multi-variable, stochastic flood hazards across urban sectors without requiring computationally exhausting structural models for every individual asset.

Practically, regional FRAs, frequently categorized as mesoscale assessments, currently rely heavily on empirical, aggregated depth-damage frameworks (see also Section 3.1). While these frameworks are effective at identifying broad geographical hazard zones, it is fundamentally flawed in mirroring the physical reality of the built environment. They assume a homogeneous vulnerability across large building typologies, treating all structures within that typology identically regardless of material degradation. The results of this research demonstrate that incorporating the material conservation state significantly alters structural vulnerability thresholds, shifting the predicted damage onset to significantly lower water depths. For regional water boards, municipal planners, heritage conservation authorities, and risk modellers, the implications are manifold:

1. **Targeted Interventions:** Rather than deploying uniform and costly flood mitigation measures across an entire historic district, asset managers can utilize this state-dependent method to prioritize interventions in zones where the heritage fabric exhibits higher vulnerability due to its deterioration state. This enables improved targeted climate adaptation strategies, compared to the current state-of-the-art.
2. **Quantifiable Impact Assessment:** Currently used methodologies are not able to quantify the actual risk and the effectiveness of intervention strategies. While risk labels provide a metric to assess vulnerability, the quantified effect remains uncertain. The developed method allows mitigation strategy based impact assessment that provides quantification metrics.
3. **Enriching Mesoscale Models:** Instead of replacing generic empirical depth-damage curves with damage curves derived from the here developed engineering based methodology for mesoscale risk assessment, the methodology should be used to enrich this assessment method. This way, after a general assessment, the granularity of mesoscale risk assessment can be increased for vulnerable regions by accounting for more building typologies and differences between buildings within the same typology. As a result, policy-makers can avoid the systematic inaccuracy of financial and structural losses in historic buildings where older masonry assets are inherently compromised by environmental ageing.

Consequently, this thesis provides a methodology to bridge the gap between mesoscale and microscale assessments. From an academic and computational perspective, vulnerability modelling suffers from a structural disconnect between scalability and accuracy. Microscale engineering approaches, such as building-specific non-linear finite element modelling, offer high physical fidelity but are computationally expensive and data-intensive, rendering them impractical for regional deployment. This thesis introduces a methodology that successfully bridges this gap, operating as a scalable, high-fidelity link between the mesoscale and microscales. By utilizing a probabilistic framework, this approach proves that microscale physical parameters, such as probabilistic material strengths and localized out-of-plane structural capacities, can be efficiently parameterized and scaled.

6.1.2 Limitations

Multiple simplifications are made on the hazard side of the risk function. These simplifications concern the fluid-structure interaction. First off, the results show that the hydrostatic pressure represents a primary driver of ultimate structural failure. In the developed framework, this pressure head is governed by a static proxy: the mean failure depth of the front door, which triggers internal flooding with the same rise rate as the flood. In reality, the equilibration of internal and external water levels is expected to be a non-linear, time-dependent process. It is dictated by the interaction between the rise rate of the flood hazard and the specific infiltration rate of the building envelope.

Additionally, the dynamic debris actions contain severe epistemic uncertainty. Although dynamic debris actions cause high concentrated point loads capable of overriding steady fluid pressures in extreme cases, the model utilizes discrete FEMA modification coefficients to account for upstream screening and shallow-water dampening. However, this framework represents a highly simplified, first-order approximation. Real-world debris impact forces are stochastically dependent on highly localized, hard to quantify urban parameters. The urban environment dictates expected debris weights or floating orientations for example.

Lastly, only a limited amount of flood actions are included in the demand parameterization. In a dynamic flood hazard scenario, the vertical stabilization provided by gravitational pre-compressive loads is actively compromised by a critical environmental interaction: buoyancy. As the external flood depth rises around the building envelope, the submerged volume of the structure experiences an upward buoyant force perpendicular to the ground plane. This upward lift acts directly against gravity, effectively "lightening" the building's dead load. Consequently, the net vertical pre-compression load on the load-bearing masonry walls decreases progressively as the asymmetrical water level difference between the exterior and interior increases.

On the vulnerability side of the risk function, a primary limitation of this research regards the parameterization of the pre-hazard condition. It relies exclusively on visible crack width as the proxy for the structural conservation state, while the theoretical framework identified a complex matrix of structural damage manifestations. In real-world assets, these damage types rarely occur in isolation. For example, a wall experiencing extensive spalling suffers a reduction in effective thickness, which exponentially penalizes the out-of-plane capacity of the wall. By isolating cracking, the model oversimplifies the complex physical state of severely deteriorated historic facades, and likely overestimates their performance.

Considering the developed methodology for reference structural capacity scaling, only the primary vertical load-bearing masonry piers are evaluated. Horizontal spandrels situated above and below door and window openings were completely omitted from the ultimate limit state bending analysis, serving only as rigid beams to transfer tributary area loads to adjacent members. From a structural mechanics perspective, this introduces a clear limitation. In a real-world out-of-plane bending scenario, spandrels beneath window openings possess minimal vertical pre-compressive stress. Consequently, these elements have lower structural capacity compared to the piers and could show local failure long before the primary load-bearing piers are mechanically threatened. While omitting spandrel collapse ensures a highly efficient, scalable structural capacity model, it means the model likely underestimates the true financial repair costs incurred by localized facade fracturing at intermediate depths.

In the loss assessment, the vulnerability functions demonstrate an immediate, steep onset of financial damage at near-zero flood depths. This behaviour is driven directly by the model assumption that floodwaters eventually penetrate and inundate the building up to the external flood depth. While this provides a conservative lower-bound estimate, it may be physically unrealistic for well-sealed basements, or floods with short durations.

Another major conceptual simplification embedded within the loss model is the mathematical assumption of statistical independence between component failures. The model samples fragility paths for individual non-structural finishes and structural facade elements independently. However, component vulnerability is strictly hierarchical and cascading. If a load-bearing URM facade panel experiences collapse, it is mechanically impossible for the components supported by that wall to remain undamaged. Neglecting these physical dependencies ignores the cascading failure mechanisms that dominate actual structural disasters.

Besides the embedded simplifications within the model, after all results were generated an error was encountered in the input parameters. The results are based on $q_{\mathrm{ref,DS3}} = q_{\mathrm{ref,DS4}} = 1.36\ \mathrm{kN}\ /m^2$, corresponding to a failure depth of 0.95m for a single-wythe wall under solely hydrostatic loading, instead of the accurate $1.61\ \mathrm{kN}\ /m^2$ derived in Subsection 4.1.1. Consequently, the results represent a conservative lower bound for the ultimate failure of the URM. Since URM failure has rarely occurred in the case studies, this change in structural resistance would have negligible effect on the results here. However, it would shift the URM repair mitigation effectiveness to slightly higher depths.

6.2 Conclusions

6.2.1 Main Findings

How can the structural capacity of URM wall panels be incorporated into probabilistic, building-specific flood vulnerability models?

The structural capacity of URM is governed by a set of geometric properties, lateral supports, and pre-compressive stresses. The geometric properties include the wall thickness, height, width-to-height ratio, and the presence and size of openings. The structural capacity scales quadratically with wall thickness, as evidenced by a double-wythe wall panel increasing the mean baseline collapse threshold from 5.79 kN/m^2 to 21.23 kN/m^2, compared to a single-wythe wall panel. Second, following Eurocode 6 mechanics, OOP capacity scales inversely quadratically with the panel height. Furthermore, structural capacity maintains a linear relationship with axial pre-compression, resulting in the first floor (105 kPa pre-compression) consistently exhibiting approximately 2.1 times the median structural capacity of the second floor (50 kPa pre-compression) across all conservation states. An equivalent frame simplification was used to account for openings in the wall panel.

To translate these deterministic physical laws into a scalable probabilistic portfolio-level model, an analytical parameterization methodology was developed that scales reference threshold capacities extracted from a single experimentally validated non-linear finite element model. These reference capacities can be scaled based on a configuration scalar, a structural capacity reduction factor, and a model uncertainty factor to obtain the threshold structural capacity where a wall panel in conservation state (CS) will reach damage state (DS), based on the sampled parameters from simulation *i*:

$$q_{\mathrm{DS,i}}(\mathrm{CS}) = \gamma_i \cdot q_{\mathrm{ref,DS,i}}(\mathrm{CS}) \cdot \varepsilon_{\mathrm{model,i}}(\mathrm{CS}) \tag{6.1}$$

$$= \left(\frac{t_i}{t_{\mathrm{ref}}}\right)^2 \cdot \left(\frac{L_{\mathrm{ref}}}{L_i}\right)^2 \cdot \left(\frac{P_i}{P_{\mathrm{ref}}}\right) \cdot \lambda_{\mathrm{CS}} \cdot q_{\mathrm{ref,DS}} \cdot \varepsilon_{\mathrm{mat,i}} \cdot \varepsilon_{\mathrm{model,i}}(\mathrm{CS}) \tag{6.2}$$

By discretizing the perforated facade strictly into primary vertical load-bearing piers, the framework conservatively isolates structural action into one-way bending panels.

How do pre-existing structural damages alter the structural capacity of URM facade walls when subjected to flood actions?

In the theoretical framework five masonry damage types were identified that affect the OOP structural capacity of URM: cracks, deformation, mortar loss, general material loss, and moisture saturation. As a result of these damages, the URM experiences a loss of tensile, shear, and torsional joint strength, compressive strength, a reduction of pre-compressive stresses, and a reduction of effective wall thickness. The quantification of the residual structural capacity is dependent on the severity, extent, and location of the specific damage types. For the model used to generate the results, only crack width is used to capture pre-existing structural damage manifestations. The results show that pre-existing cracks significantly compromise the out-of-plane lateral load capacity of URM facade walls, leading to a reduction in structural capacity of 20% for cracks with a width of 1-5 mm, and up to 40% for cracks larger than 5 mm. Lightly degraded structures with crack widths of <1 mm have negligible effects on the structural OOP capacity.

How effective are pre-flood structural restoration strategies compared to non-structural component adaptations in reducing repair costs and downtime across different conservation states?

By implementing the literature-derived parameterization methodology into a developed flood risk model, a MCS was run to evaluate the effectiveness of the different strategies. Results show that elevating non-structural components or replacing them with flood proof alternatives acts as an effective financial mitigation tool, reducing the building's aggregate damage ratio up to high inundation depths of 5.0 to 6.0 meters across all conservation states. However, component adaptation achieves zero impact on shortening building re-occupancy timelines.

Repairing deteriorated URM walls back to pristine conditions yields minor benefits for direct economic damage, providing a maximum repair cost reduction of only 3.6%. However, structural restoration is highly effective at mitigating re-occupancy downtime for degraded assets (CS2 and CS3) subjected to intermediate flood depths (2.0 to 5.0 meters), driving an average recovery timeline reduction of up to 40.7%.

To what extent does the integration of pre-flood structural conservation states alter the quantification of building-specific vulnerability within probabilistic FRA?

Ultimately, the integration of pre-flood structural conservation states fundamentally alters the quantification of vulnerability in two ways:

Financial Vulnerability: Integrating conservation states drives a 5-8 percentage point divergence in median damage curves across mid-range flood depths. This divergence is predominantly an accounting effect from early depth-triggered repair assumptions for existing masonry damage rather than mechanical failure.

Societal Vulnerability: Integrating conservation states is critical for assessing re-occupancy downtime, and thus human displacement. Assets that avoid extensive structural damage share identical re-occupancy recovery paths and can be reoccupied within roughly two months. However, crossing into extensive structural damage or collapse triggers systemic, administrative, and procurement delays that can extend displacement duration to over a year.

In conclusion, although for the selected archetype the effect of the conservation state on the financial vulnerability of a building is relatively small, societal vulnerability increases disproportionately between well maintained buildings and poorly conserved ones.

6.2.2 Future Developments

To address the fluid-structure interaction simplifications, future work could focus on the transition from static proxies to transient infiltration modelling that dynamically couples hazard rise rates with building envelope permeability. Also, replacing deterministic debris coefficients with stochastic urban simulations needs attention. Additionally, low-depth vulnerability curves could be refined by incorporating flood duration thresholds and basement sealing efficiencies, while the structural capacity engine could explicitly couple buoyant uplift forces to capture the progressive reduction of stabilizing vertical pre-compression loads. Dependent failure of components can be mitigated by integrating a system-level dependency framework, utilizing tools like Fault Tree Analysis to link component survival directly to the damage state of the component it’s dependent on. However, note that the necessity to remove these simplifications is dependent on how the model is used and whether these simplifications are acceptable in terms of their impact on the results.

A critical recommendation to advance this methodology is the execution of an extensive, multiscale validation. To verify the stability of both the geometric configuration scalar (γ_i) and the state-dependent reduction factors (λ_{CS}), the framework must be applied systematically to a broader sum of wall typologies. Furthermore, the epistemic model uncertainty factors should be calibrated.

Lastly, future work should expand the conservation state vector to multi-variable inputs, mapping concurrent visual damage types to derive a more holistic structural capacity reduction matrix.

7 Reflection

7.1 Process & Approach Evaluation

The development of the state-dependent flood risk assessment methodology required significant deviations from initial project expectations. These deviations represent a shift away from empirical simplifications toward a more rigorous, physics-based, and multi-variable computational framework.

The initial objective of this research included the formulation of an independent damage scale dedicated exclusively to quantifying a building's state of conservation. However, during the exploratory phase, it became evident that constructing an entirely new scale from first principles would introduce a high degree of subjectivity and empirical arbitrariness without extensive empirical testing and non-linear modelling. To maintain methodological rigour within the scope of this master's thesis, an operational compromise was made: the framework adopts and adapts established material degradation metrics currently utilized within seismic vulnerability assessment methodologies. While this limited the development of a novel scale, it provided a significant secondary advantage. Aligning the structural parameters with existing seismic frameworks directly improves the multi-domain integrability of the model.

Additionally, the initial intent was to research the effect of incorporating state-dependent vulnerability utilizing a simplified, empirical approach: mapping failure strictly as a function of inundation depth and applying a static, linear structural capacity reduction factor to account for material degradation. However, as the physical mechanics of unreinforced masonry (URM) under hydrostatic and hydrodynamic loading were modelled, this depth-only assumption proved insufficient, since the failure mechanism of a URM facade is fundamentally non-linear. Consequently, the approach evolved into a more elaborate, force-driven structural analysis. By transitioning to a force-driven failure analysis, the methodology captures the physical reality of building envelope failure far more accurately than standard empirical curves.

Lastly, because of anticipated validation challenges, the initial research design included a broad sensitivity analysis aimed at identifying critical variables to guide future research, rather than delivering a fully functional predictive model. However, the computational framework developed in Python surpassed these baseline expectations. While the current model requires further refinement to expand its usability and extend its applicability across a wider variety of wall geometries and building typologies, it has evolved past a simple sensitivity tool. It stands as a functional, probabilistic simulation engine capable of quantifying state-dependent risk at a structural scale. How well the developed model and methodology truly perform, will be dependent on validation upon translating it to different building and wall typologies.

7.2 Disciplinary Placement

Within the domain of structural engineering, this thesis positions itself as a transition from simplified, deterministic assessments of load-bearing elements to a probabilistic, state-dependent vulnerability framework. Traditional flood risk models rely heavily on idealized structural performance, assuming pristine material properties and static safety factors. By introducing a stochastic model that explicitly accounts for how a building's conservation state changes its vulnerability to hazards over time, this work actively reduces the idealization of structural building performance in flood scenarios.

Consequently, this research positions itself within computational design by utilizing MCS to handle multi-variable, stochastic physical problems. While this thesis does not claim to contribute foundational novelties to the field of computational theory itself, computation serves as the indispensable backbone of the developed risk model. By leveraging vectorized matrix operations, the script executes thousands of simulations in parallel. This computational efficiency reduces the total run time for a comprehensive, state-dependent regional assessment to only one minute, transforming a complex engineering problem into a highly scalable tool.

In heritage conservation, unreinforced masonry structures are recognized as highly vulnerable assets. However, implementing invasive structural retrofits to guarantee absolute safety frequently destroys the underlying architectural and historical value. This thesis addresses this tension by providing an objective, quantitative tool to justify preservation and targeted intervention for historic residential buildings. Introducing a data-driven method to assess risk accurately allows architects and conservationists to preserve heritage by intervening only where strictly necessary—effectively balancing structural safety with historical preservation. Furthermore, while initialized for residential masonry, the underlying logic of the methodology can be translated to different typologies to assess a wider range of heritage assets.

7.3 Methodological Dilemmas & Ethics

Disaster risk assessment methodologies are heavily biased toward quantifiable financial indicators, such as direct economic damage to real estate and infrastructure. However, from a humanistic perspective, the most devastating impacts of flooding are often indirect and intangible losses, including the displacement of communities, psychological trauma, and the irreversible erosion of cultural heritage. While these intangible consequences are arguably the most critical to mitigate, they are difficult to operationalize. Consequently, the risk becomes that financial metrics are the sole proxy for human suffering.

Initially, to preserve the computational scope of this thesis, re-occupancy downtimes were excluded from the research design. However, omitting these parameters felt incomplete, as it decoupled the engineering analysis from an important indicator of societal impact of flooding. To bridge this gap without exceeding scope limitations, a quantifiable framework for re-occupancy downtime was integrated into the final model. By tracking the duration required for structural stabilization, the methodology moves past purely structural or financial failure. It provides a concrete, technical metric that directly correlates with human displacement and societal disruption. While the model remains grounded in physical engineering data, framing the output around recovery timelines ensures that the research directly serves human-centric, climate-adaptive planning rather than purely real estate economics.

References

Aribisala, O. D., Yum, S.-G., Adhikari, M. D., & Song, M.-S. (2022). Flood Damage Assessment: A Review of Microscale Methodologies for Residential Buildings. *Sustainability*, *14*(21), 13817. https://doi.org/10.3390/su142113817

Bennett, J., Reeder, A., Jones, C. P., Perfetto, E., Kennedy, J., Portonova, A., & O'Connor, B. (2024). *Building Designer's Guide to Calculating Flood Loads Using ASCE 7-22 Supplement 2* (2nd edn).

Bianchi, S., Kim, K., Matteoni, M., Koniari, A. M., Koning, K., Costache, R.-D., Ciobotaru, N., Luna Navarro, A., Peng, Z., Ciurlanti, J., Shahriari, H., Mittal, D., Stavridou, E., Silva, A., Petrini, F., Palmieri, M., Pampanin, S., & Overend, M. (2025). *Resilience Readiness Levels for Buildings: Establishing Multi-Hazard Resilience Metrics and Rating Systems*. https://doi.org/10.2139/ssrn.5177406

Brans, E. J. (2012). Uitvoeren van constructief herstel. In *Historisch metselwerk: Instandhouding, herstel en conservering: Historisch metselwerk: Instandhouding, herstel en conservering* (pp. 219–235). WBooks.

Bruijn, K. D., Wagenaar, D., Slager, K., De Bel, M., & Burzel, A. (2015). *Updated and improved method for flood damage assessment : SSM2015 (version 2)* [Technical report]. https://open.rijkswaterstaat.nl/open-overheid/onderzoeksrapporten/@273917/updated-and-improved-method-for-flood/

Cartwright, T. A., Stone (ISCS), I. I. S. C. for, Bourguignon, E., Bromblet, P., Cassar, J. A., Charola, A. E., De Witte, E., Delgado-Rodrigues, J., Fassina, V., Fitzner, B., Fortier, L., Franzen, C., Miguel, J.-M. Garcia de, Hyslop, E., Klingspor-Rotstein, M., Kwiatkowski, D., Krumbein, W. E., Lefèvre, R.-A., Maxwell, I., … Young, D. (2008). *ICOMOS-ISCS: Illustrated glossary on stone deterioration patterns*. https://www.icomos.org/public/publications/monuments_and_sites/15/pdf/Monuments_and_Sites_15_ISCS_Glossary_Stone.pdf

Chang, L.-Z., Rots, J. G., & Esposito, R. (2021). Influence of aspect ratio and pre-compression on force capacity of unreinforced masonry walls in out-of-plane two-way bending. *Engineering Structures*, *249*, 113350. https://doi.org/10.1016/j.engstruct.2021.113350

CIRIA, & Environment Agency. (2003). *Flood Products: Using Flood Protection Products – A Guide for Homeowners* [Technical report].

Climate-alliance, & IWU. (2026,). *TABULA WebTool: EU Building Typology Dashboard*.

De Vent, I. A. E. (2011). *Prototype of a diagnostic decision support tool for Structural damage in masonry* [Doctoral dissertation].

Deakin. (2004). *Stream Discharge Measurement (Stream Gauging)*. Geospatial Science RMIT. https://www.mygeodesy.id.au/documents/Stream.pdf

Demirel, I., Aldemir, A., & Erberik, M. (2012). *A new computational method for the assessment of URM buildings* [Technical report].

DGMR Bouw b.v. (2006). *Referentiewoningen nieuwbouw: Kompas energiebewust wonen en werken* (Technical Report No. 2KPWB0620). https://zoek.officielebekendmakingen.nl/blg-214992.pdf

Dorland, R. van, Beersma, J., Bessembinder, J., Bloemendaal, N., Brink, H. van den, Brotons Blanes, M., Drijfhout, S., Groenland, R., Haarsma, R., Homan, C., Keizer, I., Krikken, F., Le Bars, D., Lenderink, G., Meijgaard, E. van, Meirink, J. F., Overbeek, B., Reerink, T., Selten, F., … Management, W. (2024). *KNMI National Climate Scenarios 2023 for the Netherlands* [Technical report].

Dottori, F., Figueiredo, R., Martina, M. L. V., Molinari, D., & Scorzini, A. R. (2016). INSYDE: a synthetic, probabilistic flood damage model based on explicit cost analysis. *Natural Hazards and Earth System Sciences*, *16*(12), 2577–2591. https://doi.org/10.5194/nhess-16-2577-2016

Drdácký, M. F. (2010). Flood Damage to Historic Buildings and Structures. *Journal of Performance of Constructed Facilities*, *24*(5), 439–445. https://doi.org/10.1061/(ASCE)CF.1943-5509.0000065

Dutch Green Building Council, & Climate Adaptation Services. (2022). *Framework for Climate Adaptive Buildings* [Technical report].

FEMA. (1998). *Evaluation of Earthquake Damaged Concrete and Masonry Wall Buildings: Basic Procedures Manual* (Technical Report No. FEMA P-306). https://nehrpsearch.nist.gov/static/files/NIST/PB2001100763.pdf

FEMA. (2012). *Engineering Principles and Practices for Retrofitting Flood-Prone Residential Structures* (Technical Report No. FEMA P-259, Third Edition).

Ghezelbash, A., Prosperi, A., Sharma, S., D'Altri, A. M., Rots, J. G., & Messali, F. (2026). High-Fidelity Implicit Block-Based Numerical Modeling of Out-of-Plane Behavior in Unreinforced Masonry Walls with Pre-existing Settlement-Induced Damages. In S. Saloustros & K. Beyer (Eds), *Structural Analysis of Historical Constructions: Structural Analysis of Historical Constructions*.

Grahn, T. (2020). *Assessment of residential flood damage functions to guide policy choices* (No. 2020:1). Centre for Societal Risk Research. https://www.kau.se/files/2020-06/CSR%20Report%202020-1%20Assessment%20of%20Residential%20Flood%20Damage%20finalv2_0.pdf

Grünthal, G., R.M.W. Musson, J. Schwarz, & M. Stucchi (Eds). (1998). *European Macroseismic Scale 1998*. Centre Europèen de Géodynamique et de Séismologie.

Hibbeler, R. C. (2012). *Structural Analysis* (8th edn). Prentice Hall.

Hogan, J., Almufti, I., & Ackerson, M. (2023). *REDi: Resilience-Based Design Guidelines for Floods* [Technical report].

Hudson. (1993). *Streamflow*. Silsoe Associates.

ICOMOS. (n.d.). *The Repair and Conservation of Brickwork* (pp. 39–54). https://www.icomos.org/public/publications/earthen2

Intergovernmental Panel on Climate Change. (2014). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the*

Intergovernmental Panel on Climate Change (R. K. Pachauri & L. A. Meyer, Eds) [Technical report].

Intergovernmental Panel on Climate Change. (2023). *Climate Change 2021 – The Physical Science Basis: Working Group I Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* (1st edn). Cambridge University Press. https://doi.org/10.1017/9781009157896

Jansen, L. (2019). *Structural damage to Dutch terraced houses due to flood actions*. https://repository.tudelft.nl/record/uuid:e03eacd3-1db0-4f83-81b9-57b994256c0a

Jansen, L., Korswagen, P. A., Bricker, J. D., Pasterkamp, S., Bruijn, K. M. de, & Jonkman, S. N. (2020). Experimental determination of pressure coefficients for flood loading of walls of Dutch terraced houses. *Engineering Structures*, *216*, 110647. https://doi.org/10.1016/j.engstruct.2020.110647

Kelman, I., & Spence, R. (2003). A Limit Analysis of Unreinforced Masonry Failing Under Flood Water Pressures. *Masonry International*, *16*(2), 51–61. https://www.ilankelman.org/abstracts/kelmanspence2003bi.pdf

Kelman, I., & Spence, R. (2004). An overview of flood actions on buildings. *Engineering Geology*, *73*(3–4), 297–309. https://doi.org/10.1016/j.enggeo.2004.01.010

Khattak, N., Derakhshan, H., Thambiratnam, D. P., Malomo, D., & Perera, N. J. (2023). Modelling the in-plane/out-of-plane interaction of brick and stone masonry structures using Applied Element Method. *Journal of Building Engineering*, *76*, 107175. https://doi.org/10.1016/j.jobe.2023.107175

Kok, M. (2005). *Een waterverzekering in Nederland: mogelijk en wenselijk?* (Technical Report No. PR1038).

Korswagen, P. A. (2024). *Quantifying the Probability of Light Damage to Masonry Structures: An Exploration of Crack Initiation and Progression Due to Seismic Vibrations on Masonry Buildings with Existing Damage* [Doctoral dissertation]. https://doi.org/10.4233/uuid:e56827d2-c821-4547-922e-ea24bd748e68

Korswagen, P. A., & Rots, J. G. (2021). Monitoring and Quantifying Crack-Based Light Damage in Masonry Walls with Digital Image Correlation. In M. Abdel Wahab (Ed.), *Proceedings of 1st International Conference on Structural Damage Modelling and Assessment: Proceedings of 1st International Conference on Structural Damage Modelling and Assessment*. https://doi.org/10.1007/978-981-15-9121-1_1

Korswagen, P. A., Bricker, J. D., Jonkman, S. N., & Kolen, B. (2025). Experimental and numerical analysis of a masonry wall under hydrostatic and debris impact loads. *Frontiers in Built Environment*, *11*. https://doi.org/10.3389/fbuil.2025.1644699

Martens, D. R., & Vermeltfoort, A. T. (2017). Arching effect in laterally loaded URM walls. *Proceedings of the 13th Canadian Masonry Symposium*, 1–12.

Marvi, M. T. (2020). A review of flood damage analysis for a building structure and contents. *Natural Hazards*, *102*(3), 967–995. https://doi.org/10.1007/s11069-020-03941-w

Matysek, P., & Witkowski, M. (2026). Moisture impact on compressive strength and deformability of brick masonry. *Scientific Reports*, *16*, 5147. https://doi.org/10.1038/s41598-025-31595-w

Mens, M., Grave, P. de, Noordhuis, R., & Veggel, W. van. (2024). *Huidige watergerelateerde klimaatrisico's in Nederland* [Technical report].

Merz, B., Kreibich, H., Schwarze, R., & Thieken, A. (2010). Review article "Assessment of economic flood damage". *Natural Hazards and Earth System Sciences*, *10*(8), 1697–1724. https://doi.org/10.5194/nhess-10-1697-2010

Moaiyedfar, Y., & Deniz, D. (2025). Collapse fragility models for unreinforced masonry buildings under flood and flow-type landslide actions. *Journal of Building Engineering*, *101*, 111895. https://doi.org/10.1016/j.jobe.2025.111895

Nadal, N. C., Zapata, R. E., Pagán, I., López, R., & Agudelo, J. (2010). Building Damage due to Riverine and Coastal Floods. *Journal of Water Resources Planning and Management*, *136*(3), 327–336. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000036

Nicolai, R., & Oerlemans, C. (2025). *Assessing Dutch household flood risk and adaptation effectiveness* (Technical Report No. PR5310.10). https://www.hkv.nl/wp-content/uploads/2025/11/Assessing_Dutch_household_flood_risk_and_adaptation_effectiveness.pdf

Nistor, I., Palermo, D., Nouri, Y., Murty, T., & Saatcioglu, M. (2008). *Tsunami-Induced Forces on Structures* [Technical report].

Nofal, O. M., Van De Lindt, J. W., & Do, T. Q. (2020). Multi-variate and single-variable flood fragility and loss approaches for buildings. *Reliability Engineering & System Safety*, *202*, 106971. https://doi.org/10.1016/j.ress.2020.106971

Noor-E-Khuda, S., & Thambiratnam, D. P. (2024). Unreinforced masonry walls under two-way out-of-plane bending: Comparison of FE model predictions with analytical models. *Structures*, *61*, 106126. https://doi.org/10.1016/j.istruc.2024.106126

Nouri, Y., Nistor, I., & Palermo, D. (2010). Experimental Investigation of Tsunami Impact on Free Standing Structures. *Coastal Engineering Journal*, *52*(1), 43–70.

Page, A. W. (1996). Unreinforced masonry structures. *Bulletin of the New Zealand Society for Earthquake Engineering*, *29*(4), 242–255. https://doi.org/10.5459/bnzsee.29.4.242-255

Pieterse, N., Knoop, J., Nabielek, K., Pols, L., & Tennekes, J. (2009). *Overstromingsrisicozonering in Nederland*. Planbureau voor de Leefomgeving. https://www.pbl.nl/sites/default/files/downloads/overstromingsrisicozonering_in_nederland_webpdf.pdf

Planbureau voor de Leefomgeving. (n.d.). *Correctie formulering over overstromingsrisico Nederland in IPCC-rapport*. https://www.pbl.nl/correctie-formulering-over-overstromingsrisico-nederland-in-ipcc-rapport

Poulin, M., Viau, C., Lacroix, D. N., & Doudak, G. (2017). Experimental and analytical investigation of Cross-Laminated timber panels subjected to Out-of-Plane blast loads. *Journal of Structural Engineering*, *144*(2). https://doi.org/10.1061/(asce)st.1943-541x.0001915

Rijksdienst voor Ondernemend Nederland. (2022). *Voorbeeldwoningen bestaande bouw* (Technical Report Nos RVO-231–2022/BR-DUZA). https://www.rvo.nl/sites/default/files/2023-01/brochure-voorbeeldwoningen-bestaande-bouw-2022.pdf

Schröter, K., Kreibich, H., Vogel, K., Riggelsen, C., Scherbaum, F., & Merz, B. (2014). How useful are complex flood damage models?. *Water Resources Research*, *50*(4), 3378–3395. https://doi.org/10.1002/2013WR014396

Sekkaki, M., & Chaaba, A. (2024). Failure analysis of masonry walls in 2D and 3D subjected to combined compression and shear loading. *Civil Engineering and Architecture*, *12*(6), 3944–3961. https://doi.org/10.13189/cea.2024.120614

Shabani, A., Kioumarsi, M., & Zucconi, M. (2021). State of the art of simplified analytical methods for seismic vulnerability assessment of unreinforced masonry buildings. *Engineering Structures*, *239*, 112280. https://doi.org/10.1016/j.engstruct.2021.112280

Slager, K., & Wagenaar, D. (2017). *Standaardmethode 2017 Schade en slachtoffers als gevolg van overstromingen* (Nos 11200580–4; pp. 1–27). Deltares. https://iplo.nl/publish/pages/132789/standaardmethode-2017-schade-en-slachtoffers-als-gevolg-van-overstromingen-definitief.pdf

Stephenson, V., & D'Ayala, D. (2014). A new approach to flood vulnerability assessment for historic buildings in England. *Natural Hazards and Earth System Sciences*, *14*(5), 1035–1048. https://doi.org/10.5194/nhess-14-1035-2014

TNO, TU Delft, KU Leuven, ERM, & Rijksdienst voor het Cultureel Erfgoed. (2023,). *MDCS / Damage Atlas: Monument Diagnosis and Conservation System*. https://mdcs.monumentenkennis.nl/damageatlas

Vaculik, J. (2012). *Unreinforced masonry walls subjected to out-of-plane seismic actions.* [Doctoral dissertation]. https://digital.library.adelaide.edu.au/items/825d4fe3-87a9-4e1f-8576-b534ab2cb94c

Van de Lindt, J. W., & Taggart, M. (2009). Fragility Analysis Methodology for Performance-Based Analysis of Wood-Frame Buildings for Flood. *Natural Hazards Review*, *10*(3), 113–123. https://doi.org/10.1061/(ASCE)1527-6988(2009)10:3(113)

Van de Ven, F., Goosen, H., Gehrels, H., Brolsma, R., Hofland, S., Snep, R., Bosch, P., Pötz, H., Kluck, J., de Bel, M., & Koekoek, A. (2018). *NKWK Klimaatbestendige Stad*. https://klimaatadaptatienederland.nl/publish/pages/205682/nkwk-klimaatbestendige-stad-rapportage-2018-28052019.pdf

Willis, C. R. (2004). *Design of Unreinforced Masonry Walls for Out-of-Plane Loading* [Doctoral dissertation].

A
Depth-Driven Components Data

Component library - without basement - unmitigated

Floor	Category	ID	Component	Min (m)	Max (m)	Mean(m)	SD (m)	Description	Min_UP ($)	Max_UP ($)	UP Mean $/ft2	UP Std $/ft2	No. Units	Min ($)	Max ($)	Mean ($)	SD ($)	Description
First floor	Floor	1	Decking (Floor Beams/Joists +Plywood Subflooring)	0	0.1	0.05	0.025	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		2	Flooring Insulation	0	0.1	0.05	0.025	It is below the decking and between the floor joists	0.5	1.5	1	0.25	434	217	651	434	108.5	$(0.5-1.5) per ft2
		3	Wood Flooring/Tile flooring	0	1	0.5	0.25	It is starting with the FFE	6	11	8.5	1.25	434	2604	4774	3689	542.5	$(6-11) installation and material cost
		4	Baseboard	0	1	0.5	0.25	its starting level are with the FFE	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
	Building Envelope	5	Wood Framing	0.5	4.5	2.5	1	It starts from the wood flooring. Behind brick exterior walls	5	13	9	2	234	1170	3042	2106	468	$(5-13) per square foot
		6	Internal insulation	0	0.4	0.2	0.1	its starting level are with the FFE	0.5	1.5	1	0.25	234	117	351	234	58.5	$(0.5-1.5) per board foot (1ftx1ft) and 1 inch thick of foam
		7	Dry wall (DS1)	0.1	0.4	0.25	0.075	its starting level are with the FFE. Replace bottom 40%	1	3	2	0.5	94	94	282	188	47	$(1-3) The price is based on the square foot of the covered area
		8	Dry wall (DS2)	0.8	1.5	1.15	0.175	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		9	Painting	0.1	0.4	0.25	0.075	its starting level are with the FFE	0.5	3	1.75	0.625	234	117	702	409.5	146.25	$(0.5-3) depends on the model and size
		10	Exterior Doors	0	0.5	0.25	0.125	It starts with the FFE	500	4500	2500	1000	2	1000	9000	5000	2000	$(500-1500) depends on the model and size
		11	Windows	0.5	2	1.25	0.375	It is 75 cm from the FFE	200	400	300	50	6	1200	2400	1800	300	$(200-400) depends on the model and size
	Partitions	12	Wall interior (DS1)	0.1	0.8	0.45	0.175	The same level of the ground outside	2	5	3.5	0.75	270	540	1350	945	202.5	$(2-5) The price is based on the square foot of the covered area
		13	Wall interior (DS2)	0.8	3	1.9	0.55	The same level of the ground outside	4	8	6	1	270	1080	2160	1620	270	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
		14	Baseboard partitions	0	1	0.5	0.25	its starting level are with the FFE	2	6	4	1	58	116	348	232	58	$(676-1758) average home owner expenses /$(2-6) per linear foot
		15	Interior Doors	0	0.5	0.25	0.125	Usually there is a one inch clearance at the abse	300	1000	650	175	3	900	3000	1950	525	$(300-1000) per door including installation
		16	Stairs	1	3	2	0.5	The same level of the ground outside	80	140	110	15	15	1200	2100	1650	225	$(80-140) per step
	Heating & Electrical S	17	Gas boiler	1.4	1.9	1.65	0.125	it depends if it is elevated or baed on the ground	2200	4700	3450	625	1	2200	4700	3450	625	https://zoofy.nl/prijsgidsen/wat-kost-cv-ketel-vervangen/
		18	Radiators	0.2	0.8	0.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	3	150	600	375	112.5	$(3700-7200)depends of the Model and Size
		19	Base Electrical outlets	0.15	0.45	0.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	5	15	200	107.5	46.25	$(3-40) per receptacle outlet
		20	Mid Electrical outlets	0.8	1.4	1.1	0.15	For counter appliences	3	40	21.5	9.25	3	9	120	64.5	27.75	$(3-40) per receptacle outlet
		21	Electrical Switches	0.8	1.4	1.1	0.15	for light control at mid floor height	98	193	145.5	23.75	5	490	965	727.5	118.75	$(98-193) depends on the model and size
		22	Control Box	1.6	1.9	1.75	0.075	usually it is elevated	508	1715	1111.5	301.75	1	508	1715	1111.5	301.75	$(508-1715) depends on the model and size
	Ceiling system	23	Lights fixtures ground floor	2.7	3	2.85	0.075	Ceiling level	200	900	550	175	5	1000	4500	2750	875	$(200-900) per each spot
		24	Ceiling ground floor	2.7	3	2.85	0.075	Ceiling level	1.8	11.9	6.85	2.525	434	781.2	5164.6	2972.9	1095.85	$(1.8-11.9) per square ft
	Kitchen	25	Refrigerator	0.05	0.35	0.2	0.075	Level of the compressor	450	3000	1725	637.5	1	450	3000	1725	637.5	$(450-3000) depends on the model and size
		26	Stove	0.05	0.45	0.25	0.1	Level of the oven insulation	650	2000	1325	337.5	1	650	2000	1325	337.5	$(650-2000) depends on the model and size
		27	Dishwasher	0	0.5	0.25	0.125	It is base controlled	400	700	550	75	1	400	700	550	75	$(400-700) depends on the model and size
		28	Vented Hood	1.4	2	1.7	0.15	Starting level of it	200	2000	1100	450	1	200	2000	1100	450	$(200-2000) depends on the model and size
		29	sink	0.9	1.4	1.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	1	190	900	545	177.5	$(190-900) depends on the model and size
		30	Mixers	0.9	1.4	1.15	0.125	Counter item	30	300	165	67.5	1	30	300	165	67.5	$(30-300) depends on the model and size
		31	Microwave	0.9	1.4	1.15	0.125	Counter item	150	400	275	62.5	1	150	400	275	62.5	$(150-400) depends on the model and size
		32	Lower Cabinets	0	0.9	0.45	0.225	depends on the duration	500	1000	750	125	16	8000	16000	12000	2000	$(500-1000) per legnth of the cabinet ft
		33	Kitchen Upper Cabinets	1.3	2	1.65	0.175	depends on the duration	500	1000	750	125	10	5000	10000	7500	1250	$(500-1000) per legnth of the cabinet ft
		34	Kitchen Counter top	0.8	3	1.9	0.55	It is above the counters	1870	4000	2935	532.5	1	1870	4000	2935	532.5	$(1870-4000) depends on the model and size
	Living room	35	TV	0.6	1.5	1.05	0.225	It will depend on the TV Mount Level	200	1000	600	200	1	200	1000	600	200	$(200-1000) depends on the model and size
		36	TV mount/Stand	0.05	0.65	0.35	0.15	it depends on the material	200	2000	1100	450	1	200	2000	1100	450	$(100-2000) depends on the model and size
		37	Speakers	0.05	0.65	0.35	0.15	Is it elevated on a mount or level on the floor	50	300	175	62.5	2	100	600	350	125	$(50-300) depends on the model and size
		38	Sofa and couches	0.05	0.55	0.3	0.125	It starts from the soft part above the wood frame	1200	5000	3100	950	1	1200	5000	3100	950	$(1200-5000) depends on the model and size
		39	Chairs	0.3	0.6	0.45	0.075	it depends on level of the soft part	500	1800	1150	325	1	500	1800	1150	325	$(500-1800) depends on the model and size
		40	Table	0.1	0.8	0.45	0.175	depends on the duration	200	500	350	75	1	200	500	350	75	$(200-500) depends on the model and size
	Bathroom	41	Toilet	0.3	0.8	0.55	0.125	Depends on the manufacturer but this is mean value	130	500	315	92.5	1	130	500	315	92.5	$(130-500) depends on the model and size
		42	sink	0.9	1.4	1.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	1	190	900	545	177.5	$(190-900) depends on the model and size
Second floor	Floor	43	Decking (Floor Beams/Joists +Plywood Subflooring)	5	7	6	0.5	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		44	Wood Flooring/Tile flooring	3	4	3.5	0.25	It is starting with the FFE	6	11	8.5	1.25	174	1044	1914	1479	217.5	$(6-11) installation and material cost (40% of first floor)
		45	Baseboard	3	4	3.5	0.25	its starting level are with the FFE	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
		46	Carpet	3	3.2	3.1	0.05	its starting level are with the FFE	1	6	3.5	1.25	260	260	1560	910	325	$(1-6) instlation and material cost (60% of first floor)
	Building Envelope	47	Wood Framing	3.5	7.5	5.5	1	It starts from the wood Flooring	5	13	9	2	234	1170	3042	2106	468	$(5-13) per square foot of wall
		48	Internal insulation	3	3.4	3.2	0.1	its starting level are with the FFE	0.5	1.5	1	0.25	234	117	351	234	58.5	$(0.5-1.5) per board foot (1ftx1ft) and 1 inch thick of foam
		49	Dry wall (DS1)	3.1	3.4	3.25	0.075	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		50	Dry wall (DS2)	3.8	4.5	4.15	0.175	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		51	Painting	3.1	3.4	3.25	0.075	its starting level are with the FFE	0.5	3	1.75	0.625	234	117	702	409.5	146.25	$(0.5-3) depends on the model and size
		52	Windows first floor	3.5	5	4.25	0.375	It is 75 cm from the FFE	200	400	300	50	8	1600	3200	2400	400	$(200-400) depends on the model and size
	Partitions	53	Wall interior (DS1)	3.1	3.8	3.45	0.175	The same level of the ground outside	2	5	3.5	0.75	994	1988	4970	3479	745.5	$(2-5) The price is based on the square foot of the covered area
		54	Wall interior (DS2)	3.8	6	4.9	0.55	The same level of the ground outside	2	9	5.5	1.75	994	1988	8946	5467	1739.5	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
		55	Baseboard partitions	3	4	3.5	0.25	its starting level are with the FFE	2	6	4	1	106	212	636	424	106	$(676-1758) average home owner expenses /$(2-6) per linear foot
		56	Interior Doors	3	3.5	3.25	0.125	Usually there is a one inch clearance at the abse	300	1000	650	175	4	1200	4000	2600	700	$(300-1000) per door including installation
		57	Stairs	4	6	5	0.5	The same level of the ground outside	80	140	110	15	15	1200	2100	1650	225	$(80-140) per step
	Heating & Electrical S	58	Radiators	3.2	3.8	3.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	5	250	1000	625	187.5	$(3700-7200)depends of the Model and Size
		59	Base Electrical outlets	3.15	3.45	3.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	5	15	200	107.5	46.25	$(3-40) per receptacle outlet
		60	Mid Electrical outlets	3.8	4.4	4.1	0.15	For counter appliences	3	40	21.5	9.25	6	18	240	129	55.5	$(3-40) per receptacle outlet
		61	Electrical Switches	3.8	4.4	4.1	0.15	for light control at mid floor height	98	193	145.5	23.75	6	588	1158	873	142.5	$(98-193) depends on the model and size
	Bedroom	62	Bed Room (Beds, matress, Nightstands, Dressers)	3.05	3.55	3.3	0.125	It will depend on the frame is it metal or wood	1800	8400	5100	1650	2	3600	16800	10200	3300	$(1800-8400) depends on the model and size
		63	Closets	3	3.5	3.25	0.125	Usually touches the carpet	300	2000	1150	425	2	600	4000	2300	850	$(300-2000) per door including installation
	Workroom	64	Computer	3.05	3.95	3.5	0.225	Desk item	300	3000	1650	675	1	300	3000	1650	675	$(300-3000) depends on the model and size
		65	Laptop	3.05	3.95	3.5	0.225	Desk item	300	3000	1650	675	2	600	6000	3300	1350	$(300-3000) depends on the model and size
		66	Printer	3.1	3.9	3.5	0.2	Desk item	50	200	125	37.5	1	50	200	125	37.5	$(50-200) depends on the model and size
		67	Chairs	3.3	3.6	3.45	0.075	it depends on level of the soft part	500	1800	1150	325	1	500	1800	1150	325	$(500-1800) depends on the model and size
		68	Table	3.1	3.8	3.45	0.175	depends on the duration	200	500	350	75	1	200	500	350	75	$(200-500) depends on the model and size
		69	Speakers	3.05	3.65	3.35	0.15	Is it elevated on a mount or level on the floor	50	300	175	62.5	2	100	600	350	125	$(50-300) depends on the model and size

	Bathroom	70	Toilet	3.3	3.8	3.55	0.125	Depends on the manufacturer but this is mean value	130	500	315	92.5	1	130	500	315	92.5	$(130-500) depends on the model and size
		71	sink	3.9	4.4	4.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	2	380	1800	1090	355	$(190-900) depends on the model and size
		72	bath	3.3	3.8	3.55	0.125	Depends on the manufacturer but this is mean value	300	1000	650	175	2	600	2000	1300	350	$(300-1000) depends on the model and size
		73	Bathroom lower Cabinets	3	3.9	3.45	0.225	depends on the duration	500	1000	750	125	3	1500	3000	2250	375	$(500-1000) per legnth of the cabinet ft
		74	Bathroom Upper Cabinets	4.3	5	4.65	0.175	depends on the duration	500	1000	750	125	1	500	1000	750	125	$(500-1000) per legnth of the cabinet ft
		75	Bathroom counter top	3.8	4.2	4	0.1	It is above the counters	900	1500	1200	150	1	900	1500	1200	150	$(900-1500) depends on the model and size
		76	Washer	3.05	3.25	3.15	0.05	Motor Level	350	1500	925	287.5	1	350	1500	925	287.5	$(350-1500) depends on the model and size
		77	Dryer	3.05	3.25	3.15	0.05	Motor Level	350	1500	925	287.5	1	350	1500	925	287.5	$(350-1500) depends on the model and size
	Ceiling system	78	Lights fixtures first floor	5.7	6	5.85	0.075	Ceiling level	200	900	550	175	6	1200	5400	3300	1050	$(200-900) per each spot
		79	Ceiling first floor	5.7	6	5.85	0.075	Ceiling level	1.8	11.9	6.85	2.525	434	781.2	5164.6	2972.9	1095.85	$(1.8-11.9) per square ft
Attic	Second floor	80	Decking (Floor Beams/Joists +Plywood Subflooring)	8	10	9	0.5	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		81	Wood Flooring/Tile flooring	6	7	6.5	0.25	It is starting with the FFE	6	11	8.5	1.25	434	2604	4774	3689	542.5	$(6-11) installation and material cost
		82	Baseboard	6	7	6.5	0.25	its starting level are with the FFE. Circumference of floor	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
	Heating & Electrical S	83	Radiators	6.2	6.8	6.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	2	100	400	250	75	$(3700-7200)depends of the Model and Size
		84	Base Electrical outlets	6.15	6.45	6.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	4	12	160	86	37	$(3-40) per receptacle outlet
		85	Electrical Switches	6.8	7.4	7.1	0.15	for light control at mid floor height	98	193	145.5	23.75	2	196	386	291	47.5	$(98-193) depends on the model and size
	Partitions	86	Wall interior (DS1)	6.1	6.8	6.45	0.175	The same level of the ground outside	2	5	3.5	0.75	523	1046	2615	1830.5	392.25	$(2-5) The price is based on the square foot of the covered area
		87	Wall interior (DS2)	6.8	9	7.9	0.55	The same level of the ground outside	2	9	5.5	1.75	523	1046	4707	2876.5	915.25	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
	Roof	88	Lights fixtures attic	7.7	8.5	8.1	0.2	Ceiling level	200	900	550	175	2	400	1800	1100	350	$(200-900) per each spot
		89	Ceiling attic	6	8.5	7.25	0.625	Ceiling level	1.8	11.9	6.85	2.525	531	955.8	6318.9	3637.35	1340.78	$(1.8-11.9) per square ft
		90	Insulation	6	7	6.5	0.25	Ceiling level	1.5	3.5	2.5	0.5	531	796.5	1858.5	1327.5	265.5	$(1.5-3.5) per square ft
		91	Roof (Membrane, Sheathing, Insulation, Shingles)	6	9	7.5	0.75	Strating from the attic level	1.5	7	4.25	1.375	434	651	3038	1844.5	596.75	$(1.5-7) per square foot of the building
		92	Roof (Trusses, Rafters)	6	10	8	1	Strating from the attic level	6	9	7.5	0.75	434	2604	3906	3255	325.5	$(6-9) per square foot of the building

Component library - with basement - unmitigated

Floor	Category	ID	Component	Min (m)	Max (m)	Mean(m)	SD (m)	Description	Min_UP ($)	Max_UP ($)	UP Mean $/ft2	UP Std $/ft2	No. Units	Min ($)	Max ($)	Mean ($)	SD ($)	Description
Basement	Floor	1	Concrete floor	-2.7	-2	-2.35	0.175		2	5	3.5	0.75	434	868	2170	1519	325.5	$(2-5) The price is based on the square foot of the covered area
	Envelope	2	Exterior wall (concrete or brick)	-2.7	-1.3	-2	0.35		2	5	3.5	0.75	670	1340	3350	2345	502.5	$(2-5) The price is based on the square foot of the covered area
		3	Windows	-1	-0.3	-0.65	0.175		200	400	300	50	6	1200	2400	1800	300	$(200-400) depends on the model and size
		4	Internal insulation	-2.7	-2.3	-2.5	0.1		0.5	1.5	1	0.25	234	117	351	234	58.5	$(0.5-1.5) per board foot (1ftx1ft) and 1 inch thick of foam
	Partitions	6	Wall interior (DS1)	-2.6	-1.9	-2.25	0.175		2	5	3.5	0.75	135	270	675	472.5	101.25	$(2-5) The price is based on the square foot of the covered area
		7	Wall interior (DS2)	-1.9	0	-0.95	0.475		2	9	5.5	1.75	135	270	1215	742.5	236.25	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
		8	Interior Doors	-2.7	-2.2	-2.45	0.125		300	1000	650	175	2	600	2000	1300	350	$(300-1000) per door including installation
		9	Stairs	-1.7	0	-0.85	0.425		80	140	110	15	15	1200	2100	1650	225	$(80-140) per step
	Heating & Electrical System	10	Mid Electrical outlets	-1.6	-1	-1.3	0.15		3	40	21.5	9.25	2	6	80	43	18.5	$(3-40) per receptacle outlet
		11	Electrical Switches	-1.6	-1	-1.3	0.15		98	193	145.5	23.75	3	294	579	436.5	71.25	$(98-193) depends on the model and size
	Ceiling system	12	Lights fixtures ground floor	-0.4	-0.2	-0.3	0.05		200	900	550	175	3	600	2700	1650	525	$(200-900) per each spot
		13	Ceiling ground floor	-0.4	-0.2	-0.3	0.05		1.8	11.9	6.85	2.525	434	781.2	5164.6	2972.9	1095.85	$(1.8-11.9) per square ft
	Contents	14	Closets	-2.7	-0.3	-1.5	0.6		300	2000	1150	425	2	600	4000	2300	850	$(300-2000) per door including installation
		15	Foundation	2.5	5	3.75	0.625		4	12	8	2	434	1736	5208	3472	868	$(4-12) Including replacement cost per square foot
First floor	Floor	16	Decking (Floor Beams/Joists +Plywood Subflooring)	0	0.1	0.05	0.025	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		17	Flooring Insulation	0	0.1	0.05	0.025	It is below the decking and between the floor joists	0.5	1.5	1	0.25	434	217	651	434	108.5	$(0.5-1.5) per ft2
		18	Wood Flooring/Tile flooring	0	1	0.5	0.25	It is starting with the FFE	6	11	8.5	1.25	434	2604	4774	3689	542.5	$(6-11) installation and material cost
		19	Baseboard	0	1	0.5	0.25	its starting level are with the FFE	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
	Building Envelope	20	Wood Framing	0.5	4.5	2.5	1	It starts from the wood flooring. Behind brick exterior walls	5	13	9	2	234	1170	3042	2106	468	$(5-13) per square foot
		21	Internal insulation	0	0.4	0.2	0.1	its starting level are with the FFE	0.5	1.5	1	0.25	234	117	351	234	58.5	$(0.5-1.5) per board foot (1ftx1ft) and 1 inch thick of foam
		22	Dry wall (DS1)	0.1	0.4	0.25	0.075	its starting level are with the FFE. Replace bottom 40%	1	3	2	0.5	94	94	282	188	47	$(1-3) The price is based on the square foot of the covered area
		23	Dry wall (DS2)	0.8	1.5	1.15	0.175	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		24	Painting	0.1	0.4	0.25	0.075	its starting level are with the FFE	0.5	3	1.75	0.625	234	117	702	409.5	146.25	$(0.5-3) depends on the model and size
		25	Exterior Doors	0	0.5	0.25	0.125	It starts with the FFE	500	4500	2500	1000	2	1000	9000	5000	2000	$(500-1500) depends on the model and size
		26	Windows	0.5	2	1.25	0.375	It is 75 cm from the FFE	200	400	300	50	6	1200	2400	1800	300	$(200-400) depends on the model and size
	Partitions	27	Wall interior (DS1)	0.1	0.8	0.45	0.175	The same level of the ground outside	2	5	3.5	0.75	270	540	1350	945	202.5	$(2-5) The price is based on the square foot of the covered area
		28	Wall interior (DS2)	0.8	3	1.9	0.55	The same level of the ground outside	4	8	6	1	270	1080	2160	1620	270	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
		29	Baseboard partitions	0	1	0.5	0.25	its starting level are with the FFE	2	6	4	1	58	116	348	232	58	$(676-1758) average home owner expenses /$(2-6) per linear foot
		30	Interior Doors	0	0.5	0.25	0.125	Usually there is a one inch clearance at the abse	300	1000	650	175	3	900	3000	1950	525	$(300-1000) per door including installation
		31	Stairs	1	3	2	0.5	The same level of the ground outside	80	140	110	15	15	1200	2100	1650	225	$(80-140) per step
	Heating & Electrical System	32	Gas boiler	1.4	1.9	1.65	0.125	it depends if it is elevated or baed on the ground	2200	4700	3450	625	1	2200	4700	3450	625	https://zoofy.nl/prijsgidsen/wat-kost-cv-ketel-vervangen/
		33	Radiators	0.2	0.8	0.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	3	150	600	375	112.5	$(3700-7200)depends of the Model and Size
		34	Base Electrical outlets	0.15	0.45	0.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	5	15	200	107.5	46.25	$(3-40) per receptacle outlet
		35	Mid Electrical outlets	0.8	1.4	1.1	0.15	For counter appliences	3	40	21.5	9.25	3	9	120	64.5	27.75	$(3-40) per receptacle outlet
		36	Electrical Switches	0.8	1.4	1.1	0.15	for light control at mid floor height	98	193	145.5	23.75	5	490	965	727.5	118.75	$(98-193) depends on the model and size
		37	Control Box	1.6	1.9	1.75	0.075	usually it is elevated	508	1715	1111.5	301.75	1	508	1715	1111.5	301.75	$(508-1715) depends on the model and size
	Ceiling system	38	Lights fixtures ground floor	2.7	3	2.85	0.075	Ceiling level	200	900	550	175	5	1000	4500	2750	875	$(200-900) per each spot
		39	Ceiling ground floor	2.7	3	2.85	0.075	Ceiling level	1.8	11.9	6.85	2.525	434	781.2	5164.6	2972.9	1095.85	$(1.8-11.9) per square ft
	Kitchen	40	Refrigerator	0.05	0.35	0.2	0.075	Level of the compressor	450	3000	1725	637.5	1	450	3000	1725	637.5	$(450-3000) depends on the model and size
		41	Stove	0.05	0.45	0.25	0.1	Level of the oven insulation	650	2000	1325	337.5	1	650	2000	1325	337.5	$(650-2000) depends on the model and size
		42	Dishwasher	0	0.5	0.25	0.125	It is base controlled	400	700	550	75	1	400	700	550	75	$(400-700) depends on the model and size
		43	Vented Hood	1.4	2	1.7	0.15	Starting level of it	200	2000	1100	450	1	200	2000	1100	450	$(200-2000) depends on the model and size
		44	sink	0.9	1.4	1.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	1	190	900	545	177.5	$(190-900) depends on the model and size
		45	Mixers	0.9	1.4	1.15	0.125	Counter item	30	300	165	67.5	1	30	300	165	67.5	$(30-300) depends on the model and size
		46	Microwave	0.9	1.4	1.15	0.125	Counter item	150	400	275	62.5	1	150	400	275	62.5	$(150-400) depends on the model and size
		47	Lower Cabinets	0	0.9	0.45	0.225	depends on the duration	500	1000	750	125	16	8000	16000	12000	2000	$(500-1000) per legnth of the cabinet ft
		48	Kitchen Upper Cabinets	1.3	2	1.65	0.175	depends on the duration	500	1000	750	125	10	5000	10000	7500	1250	$(500-1000) per legnth of the cabinet ft
		49	Kitchen Counter top	0.8	3	1.9	0.55	It is above the counters	1870	4000	2935	532.5	1	1870	4000	2935	532.5	$(1870-4000) depends on the model and size
	Living room	50	TV	0.6	1.5	1.05	0.225	It will depend on the TV Mount Level	200	1000	600	200	1	200	1000	600	200	$(200-1000) depends on the model and size
		51	TV mount/Stand	0.05	0.65	0.35	0.15	it depends on the material	200	2000	1100	450	1	200	2000	1100	450	$(100-2000) depends on the model and size
		52	Speakers	0.05	0.65	0.35	0.15	Is it elevated on a mount or level on the floor	50	300	175	62.5	2	100	600	350	125	$(50-300) depends on the model and size
		53	Sofa and couches	0.05	0.55	0.3	0.125	It starts from the soft part above the wood frame	1200	5000	3100	950	1	1200	5000	3100	950	$(1200-5000) depends on the model and size
		54	Chairs	0.3	0.6	0.45	0.075	it depends on level of the soft part	500	1800	1150	325	1	500	1800	1150	325	$(500-1800) depends on the model and size
		55	Table	0.1	0.8	0.45	0.175	depends on the duration	200	500	350	75	1	200	500	350	75	$(200-500) depends on the model and size
	Bathroom	56	Toilet	0.3	0.8	0.55	0.125	Depends on the manufacturer but this is mean value	130	500	315	92.5	1	130	500	315	92.5	$(130-500) depends on the model and size
		57	sink	0.9	1.4	1.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	1	190	900	545	177.5	$(190-900) depends on the model and size
Second floor	Floor	58	Decking (Floor Beams/Joists +Plywood Subflooring)	5	7	6	0.5	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		59	Wood Flooring/Tile flooring	3	4	3.5	0.25	It is starting with the FFE	6	11	8.5	1.25	174	1044	1914	1479	217.5	$(6-11) installation and material cost (40% of first floor)
		60	Baseboard	3	4	3.5	0.25	its starting level are with the FFE	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
		61	Carpet	3	3.2	3.1	0.05	its starting level are with the FFE	1	6	3.5	1.25	260	260	1560	910	325	$(1-6) instlation and material cost (60% of first floor)
	Building Envelope	62	Wood Framing	3.5	7.5	5.5	1	It starts from the wood Flooring	5	13	9	2	234	1170	3042	2106	468	$(5-13) per square foot of wall
		63	Internal insulation	3	3.4	3.2	0.1	its starting level are with the FFE	0.5	1.5	1	0.25	234	117	351	234	58.5	$(0.5-1.5) per board foot (1ftx1ft) and 1 inch thick of foam
		64	Dry wall (DS1)	3.1	3.4	3.25	0.075	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		65	Dry wall (DS2)	3.8	4.5	4.15	0.175	its starting level are with the FFE	1	3	2	0.5	234	234	702	468	117	$(1-3) The price is based on the square foot of the covered area
		66	Painting	3.1	3.4	3.25	0.075	its starting level are with the FFE	0.5	3	1.75	0.625	234	117	702	409.5	146.25	$(0.5-3) depends on the model and size
		67	Windows first floor	3.5	5	4.25	0.375	It is 75 cm from the FFE	200	400	300	50	8	1600	3200	2400	400	$(200-400) depends on the model and size
	Partitions	68	Wall interior (DS1)	3.1	3.8	3.45	0.175	The same level of the ground outside	2	5	3.5	0.75	994	1988	4970	3479	745.5	$(2-5) The price is based on the square foot of the covered area
		69	Wall interior (DS2)	3.8	6	4.9	0.55	The same level of the ground outside	2	9	5.5	1.75	994	1988	8946	5467	1739.5	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
		70	Baseboard partitions	3	4	3.5	0.25	its starting level are with the FFE	2	6	4	1	106	212	636	424	106	$(676-1758) average home owner expenses /$(2-6) per linear foot

		71	Interior Doors	3	3.5	3.25	0.125	Usually there is a one inch clearance at the abse	300	1000	650	175	4	1200	4000	2600	700	$(300-1000) per door including installation
		72	Stairs	4	6	5	0.5	The same level of the ground outside	80	140	110	15	15	1200	2100	1650	225	$(80-140) per step
	Heating & Electrical System	73	Radiators	3.2	3.8	3.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	5	250	1000	625	187.5	$(3700-7200)depends of the Model and Size
		74	Base Electrical outlets	3.15	3.45	3.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	5	15	200	107.5	46.25	$(3-40) per receptacle outlet
		75	Mid Electrical outlets	3.8	4.4	4.1	0.15	For counter appliences	3	40	21.5	9.25	6	18	240	129	55.5	$(3-40) per receptacle outlet
		76	Electrical Switches	3.8	4.4	4.1	0.15	for light control at mid floor height	98	193	145.5	23.75	6	588	1158	873	142.5	$(98-193) depends on the model and size
	Bedroom	77	Bed Room (Beds, matress, Nightstands, Dressers)	3.05	3.55	3.3	0.125	It will depend on the frame is it metal or wood	1800	8400	5100	1650	2	3600	16800	10200	3300	$(1800-8400) depends on the model and size
		78	Closets	3	3.5	3.25	0.125	Usually touches the carpet	300	2000	1150	425	2	600	4000	2300	850	$(300-2000) per door including installation
	Workroom	79	Computer	3.05	3.95	3.5	0.225	Desk item	300	3000	1650	675	1	300	3000	1650	675	$(300-3000) depends on the model and size
		80	Laptop	3.05	3.95	3.5	0.225	Desk item	300	3000	1650	675	2	600	6000	3300	1350	$(300-3000) depends on the model and size
		81	Printer	3.1	3.9	3.5	0.2	Desk item	50	200	125	37.5	1	50	200	125	37.5	$(50-200) depends on the model and size
		82	Chairs	3.3	3.6	3.45	0.075	it depends on level of the soft part	500	1800	1150	325	1	500	1800	1150	325	$(500-1800) depends on the model and size
		83	Table	3.1	3.8	3.45	0.175	depends on the duration	200	500	350	75	1	200	500	350	75	$(200-500) depends on the model and size
		84	Speakers	3.05	3.65	3.35	0.15	Is it elevated on a mount or level on the floor	50	300	175	62.5	2	100	600	350	125	$(50-300) depends on the model and size
	Bathroom	85	Toilet	3.3	3.8	3.55	0.125	Depends on the manufacturer but this is mean value	130	500	315	92.5	1	130	500	315	92.5	$(130-500) depends on the model and size
		86	sink	3.9	4.4	4.15	0.125	Depends on the manufacturer but this is mean value	190	900	545	177.5	2	380	1800	1090	355	$(190-900) depends on the model and size
		87	bath	3.3	3.8	3.55	0.125	Depends on the manufacturer but this is mean value	300	1000	650	175	2	600	2000	1300	350	$(300-1000) depends on the model and size
		88	Bathroom lower Cabinets	3	3.9	3.45	0.225	depends on the duration	500	1000	750	125	3	1500	3000	2250	375	$(500-1000) per legnth of the cabinet ft
		89	Bathroom Upper Cabinets	4.3	5	4.65	0.175	depends on the duration	500	1000	750	125	1	500	1000	750	125	$(500-1000) per legnth of the cabinet ft
		90	Bathroom counter top	3.8	4.2	4	0.1	It is above the counters	900	1500	1200	150	1	900	1500	1200	150	$(900-1500) depends on the model and size
		91	Washer	3.05	3.25	3.15	0.05	Motor Level	350	1500	925	287.5	1	350	1500	925	287.5	$(350-1500) depends on the model and size
		92	Dryer	3.05	3.25	3.15	0.05	Motor Level	350	1500	925	287.5	1	350	1500	925	287.5	$(350-1500) depends on the model and size
	Ceiling system	93	Lights fixtures first floor	5.7	6	5.85	0.075	Ceiling level	200	900	550	175	6	1200	5400	3300	1050	$(200-900) per each spot
		94	Ceiling first floor	5.7	6	5.85	0.075	Ceiling level	1.8	11.9	6.85	2.525	434	781.2	5164.6	2972.9	1095.85	$(1.8-11.9) per square ft
Attic	Second floor	95	Decking (Floor Beams/Joists +Plywood Subflooring)	8	10	9	0.5	Assuming 23/32 in thickness starting from joists top surface	10	25	17.5	3.75	434	4340	10850	7595	1627.5	$(10-25) per squar ft including floor beams, joists and subflooring
		96	Wood Flooring/Tile flooring	6	7	6.5	0.25	It is starting with the FFE	6	11	8.5	1.25	434	2604	4774	3689	542.5	$(6-11) installation and material cost
		97	Baseboard	6	7	6.5	0.25	its starting level are with the FFE. Circumference of floor	2	6	4	1	86	172	516	344	86	$(676-1758) average home owner expenses /$(2-6) per linear foot
	Heating & Electrical System	98	Radiators	6.2	6.8	6.5	0.15	It is elevated by 45cm from the ground	50	200	125	37.5	2	100	400	250	75	$(3700-7200)depends of the Model and Size
		99	Base Electrical outlets	6.15	6.45	6.3	0.075	base sockets for vaccumes and floor machines	3	40	21.5	9.25	4	12	160	86	37	$(3-40) per receptacle outlet
		100	Electrical Switches	6.8	7.4	7.1	0.15	for light control at mid floor height	98	193	145.5	23.75	2	196	386	291	47.5	$(98-193) depends on the model and size
	Partitions	101	Wall interior (DS1)	6.1	6.8	6.45	0.175	The same level of the ground outside	2	5	3.5	0.75	523	1046	2615	1830.5	392.25	$(2-5) The price is based on the square foot of the covered area
		102	Wall interior (DS2)	6.8	9	7.9	0.55	The same level of the ground outside	2	9	5.5	1.75	523	1046	4707	2876.5	915.25	$(4-8) per square foot, based on Korswagen (2024) and max 50% repointing needed
	Roof	103	Lights fixtures attic	7.7	8.5	8.1	0.2	Ceiling level	200	900	550	175	2	400	1800	1100	350	$(200-900) per each spot
		104	Ceiling attic	6	8.5	7.25	0.625	Ceiling level	1.8	11.9	6.85	2.525	531	955.8	6318.9	3637.35	1340.78	$(1.8-11.9) per square ft
		105	Insulation	6	7	6.5	0.25	Ceiling level	1.5	3.5	2.5	0.5	531	796.5	1858.5	1327.5	265.5	$(1.5-3.5) per square ft
		106	Roof (Membrane, Sheathing, Insulation, Shingles)	6	9	7.5	0.75	Strating from the attic level	1.5	7	4.25	1.375	434	651	3038	1844.5	596.75	$(1.5-7) per square foot of the building
		107	Roof (Trusses, Rafters)	6	10	8	1	Strating from the attic level	6	9	7.5	0.75	434	2604	3906	3255	325.5	$(6-9) per square foot of the building

Table A.1: Changes in mitigated component library compared to the original.

Floor	**Component**	**Changed Parameter**	**Original Data**	**New Data**	**Description**
Basement	Closets	Unit Price (max)	\$2000	\$1000	Move valuable contents to higher floors
Basement	Light switches	Amount of units	3	1	Reduce amount
Basement	Light fixtures	Amount of units	3	2	Reduce amount
Basement	Stairs	Depth resistance (min, max)	\$80-140	\$0	Substitute with flood proof material
Basement	Interior doors	Amount of units	2	1	Remove unnecessary doors
First Floor	Control box	Depth resistance (min, max)	1.6-1.9 m	1.7-2.0 m	Move up slightly
First Floor	Gas boiler	Depth resistance (min, max)	1.4-1.9 m	7.4-7.9 m	Move to the attic
First Floor	Microwave	Depth resistance (min, max)	0.9-1.5 m	1.2-1.8 m	Elevate higher
First Floor	Mixer	Depth resistance (min, max)	0.9-1.5 m	1.2-1.8 m	Elevate higher
First Floor	Lower kitchen cabinets	Unit price (min, max)	\$500-1000	\$0	Substitute with flood proof material
First Floor	All other kitchen components	Depth resistance (min, max)	min, max	min + 0.2m, max + 0.2 m	Move valuable contents to higher floors.
First Floor	Base electrical outlets	Depth resistance (min, max)	0.15-0.45 m	0.8-1.4 m	Move up to mid floor height
First Floor	Speakers	Depth resistance (min, max)	0.05-0.65 m	0.5-1.1m	Place on or in cabinet

B

Model Inputs

Global Inputs:

N_{sim} = 10.000

Depths = [0 – 10.01] steps of 0.01 m

BA Demand Inputs

Static Inputs

Parameter	Symbol	Value	Unit
Gravitational acceleration	g	9.81	m/s²
Flood water density	ρ_w	1025	kg/m³
Flood water specific weight	γ_w	10.057	kN/m³
Ingress lag (max indoor-outdoor differential)	$d_{\text{delta,max}}$	0.25	m
Flow velocity cap	$V_{\max}$	3.0	m/s

The static blockage coefficient and structure coefficient for the calculations of the debris impact forces are based on Table B.1 and Table B.2.

Table B.1: Blockage coefficient C_B by upstream screening condition. (Adopted from FEMA (2012)

Degree of Screening or Sheltering within 100 ft Upstream	C_B
No upstream screening, flow path wider than 30 ft	1.0
Limited upstream screening, flow path 20 ft wide	0.6
Moderate upstream screening, flow path 10 ft wide	0.2
Dense upstream screening, flow path less than 5 ft wide	0.0

Table B.2: Structure coefficient C_{Str} by foundation and structural type. (Adopted from FEMA (2012)

Structure Description	C_{Str}
Timber pile and masonry column supported structures, ≤ 3 stories above grade	0.2
Concrete pile or concrete/steel moment resisting frames, ≤ 3 stories above grade	0.4
Reinforced concrete (incl. insulated concrete) and reinforced masonry foundation walls	0.8

Variable Inputs

Debris mass W_d are uniformly distributed between 453-1088 kg. These weights are derived from table 24 in ASCE 7-22 Supplement 2 (Bennett et al., 2024) and represent a wooden log and a small vehicle. Weights were given in lbs and converted to kg.

The local flow velocity (V_{mean}) is coupled directly to the continuous water depth (d_e) using a simplified variant of the classic Manning open-channel flow equation (Deakin, 2004):

$$V_{\text{mean}} = C \cdot d_e^{\frac{2}{3}} \tag{B.1}$$

Where d_e represents the water depth (m), and C is a lumped topographic coefficient set to $1.0\text{m}^{\frac{1}{3}}/\,\text{s}$.

This mathematical simplification relies on two distinct hydraulic assumptions tailored to urban flood landscapes. First, for wide, shallow overland flow fields—such as pluvial accumulation or riverine breaches sweeping through an urban street grid—the channel width is orders of magnitude greater than the flow depth. Consequently, the wetted perimeter approaches the channel width, causing the hydraulic radius (R_h) to collapse to the inundation depth ($R_h \approx d_e$).

Second, the lumped constant ($C = 1.0$) synthesizes the interaction between the regional energy slope (S) and the surface roughness (n) of the built environment ($C = \sqrt{\frac{S}{n}}$). For the typical flat gradients found in low-lying Dutch urban drainage basins ($S \approx 0.15\%$) paired with standard pavement and building obstruction roughness parameters ($n \approx 0.038$) (derived from Hudson (1993)), the fraction evaluates near unity. This allows the vectorized demand engine to efficiently compute realistic hydrodynamic velocity vectors across the extensive sample space.

Flow velocity for every simulation can then lognormally be sampled with:

- sigma = $\sqrt{\ln(1 + 0.20^2)}$
- median = $\ln(V_{\text{mean}}) - 0.5 * \sigma_v^2$

The drag coefficient used for the hydrodynamic forces and the depth coefficient for the debris impact forces are dynamically extracted from Table B.3 and Table B.4.

Table B.3: Drag coefficient C_d by width-to-flood depth ratio. (Adopted from FEMA (2012)

Width to Flood Depth Ratio (*B*/*d*)	C_d
1–12	1.25
13–20	1.3
21–32	1.4
33–40	1.5
41–80	1.75
81–120	1.8
>120	2.0

Table B.4: Depth coefficient C_D by flood hazard zone and stillwater depth. (Adopted from FEMA (2012)

Flood Depth	C_D
> 5 ft (1.52 m)	1.0
4 ft (1.22 m)	0.75
3 ft (0.91 m)	0.5
2 ft (0.61 m)	0.25
< 1 ft (0.30 m)	0.0

BB URM Capacity Parameters

While URM DS2-4 are triggered by flood forces, URM DS1 is triggered by inundation depth. Since DS1 considers drying and cleaning as repair actions, the URM must be sufficiently inundated to need drying and cleaning. The depth at which this happened was sampled with a truncated normal distribution with bounds at 0.1 m and 0.5 m, a mean of 0.3m and a standard deviation of 0.1 m.

For DS2-4 structural capacity the following parameters are important:

Table B.5: URM geometry parameters sampled in the Monte Carlo Simulation.

Parameter	Distribution	Inputs			
		Min	Max	Mean	Std
L (mm)	Truncated normal	2600	3000	2800	100
B (mm)	Truncated normal	4500	5700	5100	300
B_{door} (mm)	Normal	—	—	900	50
B_{win} (mm)	Normal	—	—	2000	200
t (mm)	Truncated normal	205	235	220	7.5
P_{FF} (kPa)	Truncated normal	95	115	105	5
P_{SF} (kPa)	Truncated normal	45	55	50	2.5
FFE no basement (m)	Uniform	0.05	0.10	—	—
FFE basement (m)	Truncated normal	0.50	1.00	0.75	0.125

Figure B.1 shows the histograms of the sampled vaues for L, B, and t based on the data in Table B.5.

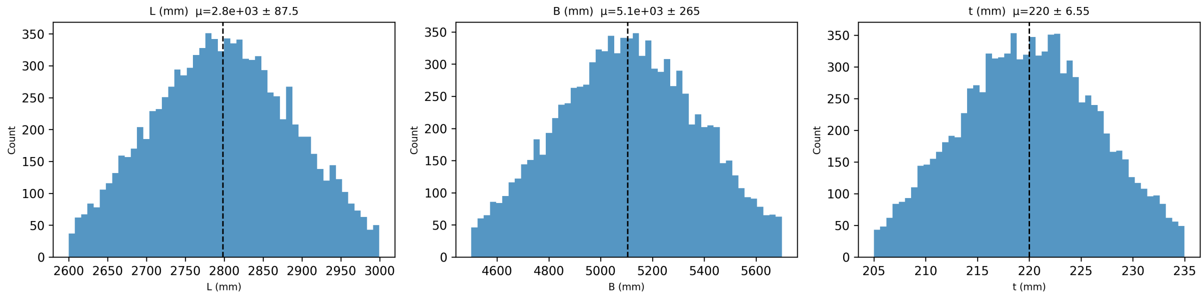


Figure B.1: Histograms of sampled data for L, B, and t

Furthermore:

- ε_{mat} is lognormally distributed around a median of 1 with a dispersion of $\beta_{\text{mat}} = 0.2$
- $\varepsilon_{\text{model}}$ is lognormally distributed around a median of 1 with a dispersion of $\beta_{\text{mat}} = \{\text{CS0}: 0.25, \text{CS1}: 0.25, \text{CS2}: 0.35, \text{CS3}: 0.50\}$

Precompression calculations

The self-weight of the URM wall per meter is $q_{\text{wall}} = \gamma \times t \times L$, where γ is the specific weight of URM, t is the wall thickness, and L is the wall height. Based on a floor height of 3 meters, the pre-compressive stress at the middle of the first floor and second floor can be calculated for $L = 4.5 \wedge 1.5$ m respectively. The specific weight of URM was taken as $\gamma = 18.54$ kN/m³ (mean), based on a mass density of 1890 kg/m³. Applying a coefficient of variation of 3%,giving a 95% confidence interval of $\gamma \in [17.45, 19.63]\text{kN/m}^3$

This interal is used to set up the lower and upper bound for the precompression range. The mass density and coefficient of variation are derived from literature (Khattak et al., 2023).

Table B.6: $q_{\text{wall,line}}$ calculations based on the lower and upper bound for the first floor (FF) and second floor (SF).

Bound	γ (kN/m³)	t (m)	L (m)	$q_{\text{wall,line}}$ (kN/m)
Lower FF	17.45	0.21	4.5	16.49
Upper FF	19.63	0.25	4.5	22.08
Lower SF	17.45	0.21	1.5	5.50
Upper SF	19.63	0.25	1.5	7.36

The roof is a typical roof with wooden trusses and clay tiles, with an assumed self-weight of $q_{\text{roof}} \in [0.8, 1.2]$ kN/m². Heavy clay tiles (e.g. traditional Mediterranean pan tiles) can push toward the upper end (1.2 kN/m²), while lighter flat clay tiles sit closer to 0.8 kN/m². Based on the archetype data, a tributary area of 25 m² is carried over a mean wall length of 5.1 m, giving a line load of:

$$q_{\text{roof,line}} = \frac{A_{\text{trib}} \times q_{\text{roof}}}{L_{\text{wall}}} \tag{B.2}$$

Table B.7: $q_{\mathrm{roof,line}}$ calculations based on the lower and upper bound.

Bound	q_{roof} (kN/m²)	$q_{\mathrm{roof,line}}$ (kN/m)
Lower	0.8	25 × 0.8 / 5.1 = 3.92
Upper	1.2	25 × 1.2 / 5.1 = 5.88

The total vertical stress at mid wall height is:

$$P = \frac{W_{\mathrm{wall}} + q_{\mathrm{roof,line}}}{t} \tag{B.3}$$

Giving:

Table B.8: P calculations based on the lower and upper bound for the first floor (FF) and second floor (SF).

Bound	t (m)	Total load (kN/m)	P (kPa)
Lower FF	0.21	16.49 + 3.92 = 20.41	**97.19**
Upper FF	0.25	22.08 + 5.88 = 27.96	**111.84**
Lower SF	0.21	5.50 + 3.92 = 9.42	**44.86**
Upper SF	0.25	7.36 + 5.88 = 13.24	**52.96**

To account for simplifications in the calculations, the precompression range applied in the analyses is therefore 95-115 kPa for the first floor, and 45-55 kPa for the second floor. The lower bound corresponds to a relatively light wall and light roofing, and the upper bound to a heavier wall with heavy roofing.

BC Loss Parameters

URM repair costs:

	Lower bound	Upper bound	min_euro/m²	max_euro/m²	min_dollar/m²	max_dollar/m²	min_dollar/ft²	max_dollar/ft²
DS2	20%	80%	20	80	23	93	2	9
Variable	repoint median	repoint median						
DS3	10	25%	46	100	54	116	5	11
Variable	x bricks replacement	rebuild median						
DS4	100%	100%	280	700	326	814	30	76
Variable	rebuild min	rebuild max						

	min (€)	max (€)	median (€)
repoint	60	190	100
replace	50	120	100
rebuild	280	700	400

euro to dollar average 2025: 1.1626

m² to ft²: 10.76391042

exterior wall area per floor 21.725 m²

Figure B.2: Derivation of repair costs per DS. Note that prices must be converted to dollars per square feet to be in line with the data provided by Nofal et al. (2020).

The prices for the lower and upper bound of the total building replacement costs are based on the key figures provided by the Dutch government for: A) new terraced houses, and B) an extension to a residential dwelling respectively. The prices are given per square meter gross floor area. The costs are sampled from a truncated normal distribution between €1318 and €1891. (See website).

Table B.9 lists the impeding delay parameters.

Table B.9: Impeding factor delay distributions (lognormal parameters). (Hogan et al., 2023)

Delay	Median	σ	Unit
Flood duration	2	0.5	days
Cleanup (contracted, limited budget)	3	0.5	weeks
Cleanup (uncontracted, limited budget)	1	0.3	months
Cleanup (contracted, unlimited budget)	2	0.5	days
Cleanup (uncontracted, unlimited budget)	12	0.8	days
Permitting (contracted)	7	0.4	weeks
Permitting (uncontracted)	12	0.5	weeks
Financing (pre-arranged)	1	0.5	weeks
Financing (private loans)	15	0.7	weeks
Financing (insurance)	6	0.5	weeks
Contractor mobilisation (contracted)	1.5	0.5	weeks
Contractor mobilisation (uncontracted)	6	0.5	weeks
URM specialist lead time	5	0.5	weeks

C
Framework

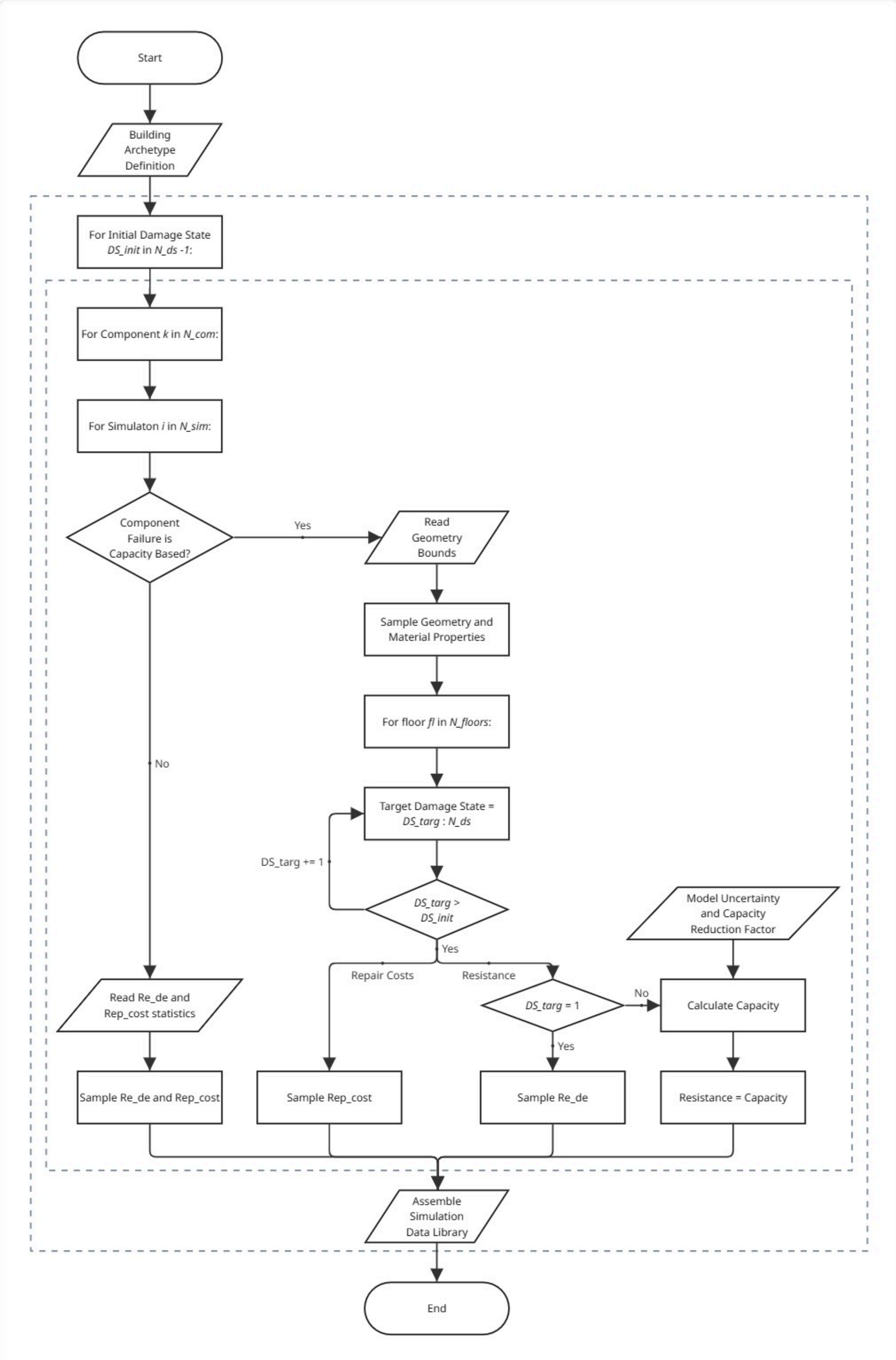


Figure C.1: Simulation data assembly. Phase II

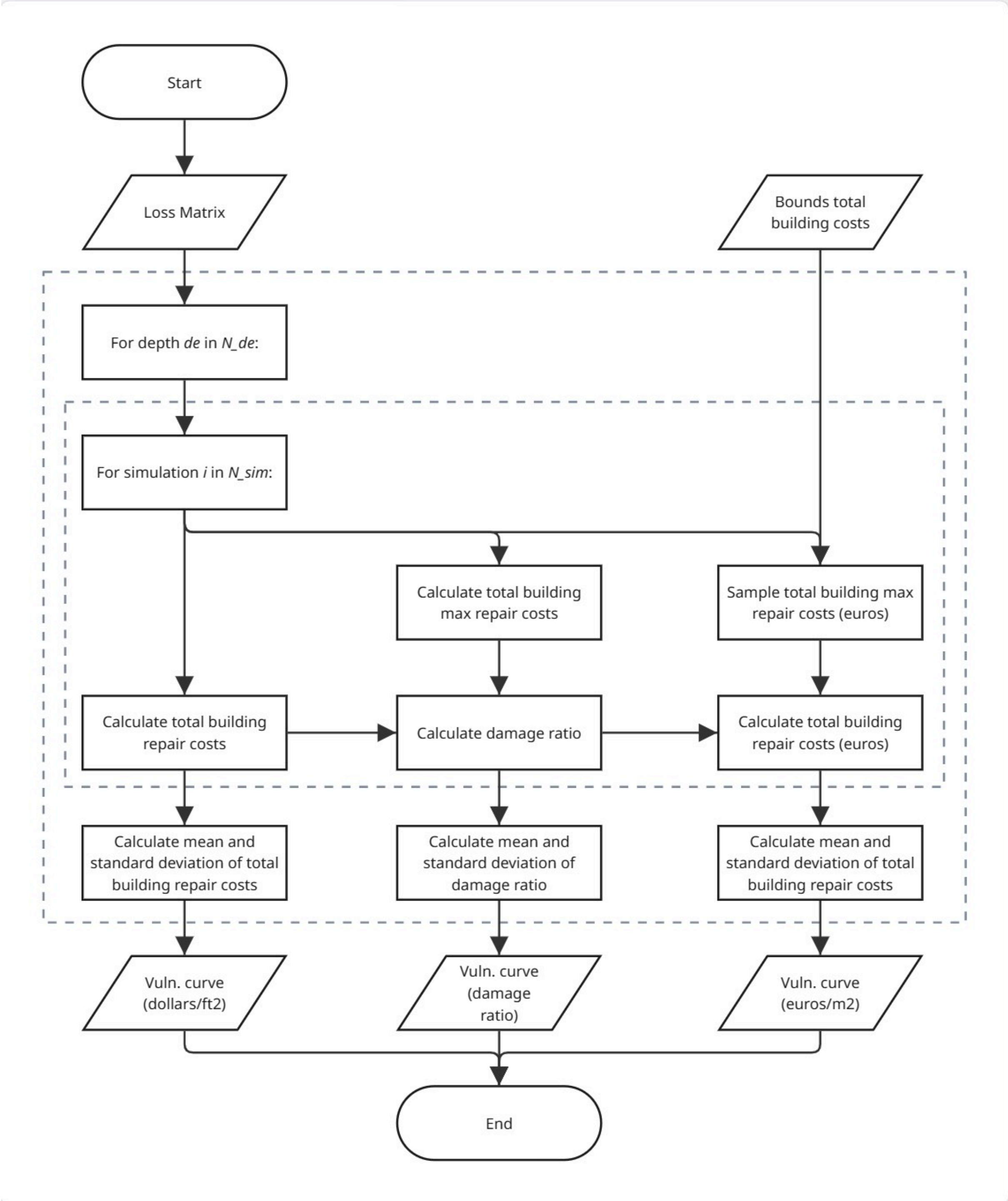


Figure C.2: Vulnerability and fragility curves generation. Phase III

D

Model Python Code

````markdown
# Monte Carlo Flood Risk Model — Dutch URM Terraced House

Probabilistic flood loss and downtime model for an unreinforced masonry (URM)
terraced house archetype, based on Dutch housing stock data (RVO/Tabula/DGMR).
Damage is evaluated across four Conservation State (CS) scenarios representing
pre-existing wall degradation.

The pipeline is split into two scripts that communicate only via `.pkl` files,
so data generation and post-processing can be run independently.

## Quick start

**Step 1 — Generate data (run once):**
```
python initialize.py
```
A settings window opens. Fill in parameters and click **GENERATE DATA**.
All sampled arrays are saved to a timestamped folder `results_DDMM_HHMM/`.

**Step 2 — Post-process (run as many times as needed):**
```
python postprocess.py
```
Browse to the results folder from Step 1, choose scenarios and plot options,
then click **PROCESS**. No re-sampling is performed; all randomness is locked
in the pkl files.

**Step 3 — Location risk (optional):**
```
python extract_location_risk.py
```
GUI to select a geographic location and results folder. Extracts p50/p95
statistics for damage ratio, loss (€/m²), and downtime at location-specific
flood depths, and saves box-plot figures and a CSV summary.

## Required input files

| File | Purpose |
|---|---|
| `components_noURM.csv` | Non-URM component library (no basement) |
| `components_noURM_basement.csv` | Non-URM component library (with basement) |
| `components_noURM_mitigated.csv` | Mitigated component library (no basement) |
````

| `components_noURM_basement_mitigated.csv` | Mitigated component library (with basement) |

CSV column layout: cols 4–7 are (lower, upper, mean, std) for flood depth resistance; cols 14–17 are (lower, upper, mean, std) for repair cost ($/unit).

Pkl interface between the two scripts

`initialize.py` writes the following files to `results_DDMM_HHMM/`.
`postprocess.py` reads them; plots go to a `plots/` sub-folder.

File	Written by	Read by	Contents
`run_config.pkl`	initialize	postprocess	N_SIM, MAX_DEPTH, INCREMENTS, FLOORAREA, SEED
`master_table.pkl`	initialize	postprocess	Sampled geometry (L, B, t, P_FF, P_SF, FFE)
`library_URM_floor{n}.pkl`	initialize	postprocess	URM resistance + cost arrays per floor
`library_components.pkl`	initialize	initialize	Non-URM component arrays (if AllComponents)
`simData_cs{0-3}.pkl`	initialize	postprocess	Assembled component library per CS scenario
`downtime_model.pkl`	initialize	postprocess	Pre-sampled downtime (days) per DS × N_sim
`demandMatrix.pkl`	initialize	postprocess	EUL demand arrays (first + second floor)
`lossMatrix_cs{cs}.pkl`	postprocess	postprocess / extract	Repair costs per depth × simulation
`downtimeMatrix_cs{cs}.pkl`	postprocess	postprocess / extract	Downtime per depth × simulation
`dsMatrix_cs{cs}.pkl`	postprocess	extract	DS distribution fractions per depth
`damageMatrix_cs{cs}.pkl`	postprocess	postprocess	Component failure probabilities per depth
`euroMatrix_cs{cs}.pkl`	postprocess	extract	Loss (€/m^2) per depth × simulation

Pipeline overview

initialize.py

```
InitGUI
    │
    ▼
np.random.seed(SEED)  ← geometry RNG seeded here
    │
    ▼  (saves master_table.pkl + run_config.pkl)
ComponentLibraryBuilder.build_all()
    ├── URM_library(floor=0)       → library_URM_floor0.pkl
    ├── URM_library(floor=1)       → library_URM_floor1.pkl
    ├── comp_nobrick_library()     → library_components.pkl   [if AllComponents]
    └── assemble_lib(cs=0..3)      → simData_cs{0..3}.pkl
    │
    ▼
DowntimeModel(seed=SEED)._build() → downtime_model.pkl
    │
    ▼
DemandGenerator.generate()        → demandMatrix.pkl
                                    demand_velocity.png
```

```

### postprocess.py

```
PostProcessGUI (browse to results folder; pick scenarios + plot options)
    │
    ▼ (loads run_config.pkl, master_table.pkl)
FloodRiskEvaluator.__init__()
    loads demandMatrix.pkl
    loads downtime_model.pkl
    │
    ▼ (repeated for each selected CS scenario)
FloodRiskEvaluator.compute_scenario_loss(cs)   [if recompute = True]
    └── lossMatrix_cs{cs}.pkl
        downtimeMatrix_cs{cs}.pkl
        dsMatrix_cs{cs}.pkl
        damageMatrix_cs{cs}.pkl
    │
    ▼
FloodRiskEvaluator.plot_lossCurves(cs)
    └── plots/compFragCurves.png
        plots/urmCapacityCDF_cs{cs}.png
        plots/lossCurve_dollar_cs{cs}.png
        plots/damageRatioCurve_cs{cs}.png
        plots/lossCurve_Euro_cs{cs}.png   → also writes euroMatrix_cs{cs}.pkl
    │
    ▼
FloodRiskEvaluator.plot_damageRatio_allCS()
    └── plots/damageRatioCurve_allCS.png
    │
    ▼
FloodRiskEvaluator.plot_downtime_cdfs()
    └── plots/downtime_CDF_by_DS.png
    │
    ▼
FloodRiskEvaluator.plot_urmCapacity_allCS()
    └── plots/urmCapacityCDF_DS{tag}_allCS.png
        plots/urmCapacityCDF_DS{tag}_allCS_withDemand.png
```

## Settings — initialize.py

| Setting | Default | Description |
|---|---|---|
| Evaluate All Components | True | Include non-URM components from CSV |
| Include Basement | True | Affects FFE distribution and CSV path |
| Mitigated | False | Use flood-proofed component CSV variant |
| Number of Simulations | 10 000 | Monte Carlo sample size |
| Max Flood Depth (m) | 10.0 | Upper end of depth sweep |
| Depth Increments (m) | 0.01 | Resolution of depth sweep |
| Max Plot Depth (m) | 7.0 | X-axis limit on demand summary plot |
| Floor Area (m$^2$) | 100 | Used to normalise costs to \$/m$^2$ |
| Random Seed | 42 | Seeds both numpy global RNG and DowntimeModel RNG |

## Settings — postprocess.py
```

Setting	Default	Description
Results folder	—	Path to a `results_DDMM_HHMM/` folder from initialize
Scenarios to process	CS 0–3	Which Conservation State scenarios to evaluate
Re-compute scenario losses	True	Uncheck to re-plot from existing pkl without re-running
Min plot depth (m)	0.0	X-axis lower limit on output plots
Max plot depth (m)	7.0	X-axis upper limit on output plots
Component Fragility Curves	True	Plot P(fail) vs. depth per component
Damage Ratio	True	Plot repair cost / replacement cost vs. depth
Losses in Dollars ($\$/m^2$)	True	Plot mean repair cost in USD
Losses in Euros (€/m^2)	True	Plot mean repair cost in EUR

Settings — extract_location_risk.py

Setting	Description
Location	`kamperfoeliestraat` (depths 0.30, 1.32 m) or `sGravendijkwal` (depths 0.20, 2.18 m)
Results folder	Path to a processed `results_DDMM_HHMM/` folder

Outputs are saved to `{results_folder}/{location}/`:
- `location_risk.csv` — p50/p95 of DR, downtime (weeks), and loss (€/m^2) per CS and depth
- `location_risk_charts_d{depth}m.png` — 2×2 box-plot figure per target depth

Conservation State (CS) scenarios

CS	Description	Capacity reduction	Model uncertainty (β)
0	Undamaged	1.00	0.25
1	Minor pre-damage	1.00	0.25
2	Moderate pre-damage	0.80	0.35
3	Extensive pre-damage	0.60	0.50

Structural model (URM walls)

Out-of-plane lateral capacity is derived from yield-line theory for a
two-way spanning simply-supported panel:

```
q = q_ref × (t/t_ref)² × (L_ref/L)² × (P/P_ref) × mat_factor × eps
```

- `q_ref` — reference EUL capacity per target DS in kN/m^2 (calibrated at t=110 mm, L=2700 mm, P=100 kN/m)
- `mat_factor` — lognormal material variability (COV = 20%)
- `eps` — lognormal model uncertainty

Reference capacities: DS2 = 1.16 kN/m^2, DS3 = DS4 = 1.36 kN/m^2.

The facade is divided into three piers by one door (≈900 mm) and one window
(≈2000 mm). The weakest pier governs.

Demand model

Three load types are computed as Equivalent Uniform Loads (EUL, kN/m^2) for
direct comparison with the wall capacity:

Hydrostatic — piecewise differential pressure:
- Indoor water level lags outside by 0.5 m (ingress threshold)
- Net pressure = uniform indoor block (F_uni) + triangular differential (F_tri)
- Centroid computed as force-weighted average of both components
- Active for both first and second floor walls once flood depth exceeds their respective floor levels

Hydrodynamic — `0.5 × Cd × V² × H_eff`
- `Cd` from B/H aspect ratio lookup (FEMA P-55)
- Flow velocity lognormal with mean = d^(2/3), COV = 20%, capped at 3 m/s

Impact — depth-dependent debris coefficient × (W × V)
- Debris mass W sampled uniformly between 453 and 1 088 kg (FEMA P-55)
- Applied over 1 m effective width; only active within panel height

All forces are converted to EUL via bending-moment equivalence:
`w_eq = 8 × M_max / Z²` where `M_max = P × y × (Z−y) / Z`.

Downtime model

Total downtime = max(impeding sequences) + repair time.
Drying is implicitly captured within the cleanup impeding sequences.

Post-hazard DS = max(flood DS, baseline DS), where baseline DS = max(CS, 1)
whenever water reaches the wall (cleaning is always required).

Impeding factor sequences (maximum of parallel paths governs):

DS	Sequences
0–2	flood duration + cleanup (contracted/uncontracted × limited/unlimited access)
3–4	above + permitting + financing + contractor mobilisation + URM lead time

Repair actions:

DS	Actions
0–2	none
3	repointing + crack repair
4	rebuild

All downtime samples are pre-drawn once in `initialize.py` and stored
in `downtime_model.pkl` (columns = DS 0–4, rows = simulations).
`postprocess.py` loads this file as a DataFrame and indexes it directly.

Reproducibility

The two scripts are connected entirely through the pkl files in the results
folder. To reproduce a run exactly:

1. Keep the entire `results_DDMM_HHMM/` folder (including `run_config.pkl`
   and `master_table.pkl`).
2. Run `postprocess.py` pointing at that folder — no re-sampling occurs.

The global numpy RNG is seeded with `SEED` before geometry sampling.

```
`DowntimeModel` uses a separate `np.random.default_rng(seed=SEED)` so its
samples are independent of the geometry RNG sequence.
"""
initialize_v0.py — Data generation pipeline for the Dutch URM flood risk model.

Generates and saves all Monte Carlo samples to a timestamped results folder.
Run this once; post-process the saved pkl files with postprocess_v0.py.

Classes
-------
  InitGUI               — tkinter window for generation settings.
  ComponentLibraryBuilder — stochastic URM and non-URM resistance/cost libraries.
  DowntimeModel           — pre-samples impeding-factor + drying + repair downtime.
  DemandGenerator         — hydrostatic / hydrodynamic / impact demand matrices.

Output folder structure (results_DDMM_HHMM/)
---------------------------------------------
  run_config.pkl          simulation parameters (loaded by postprocess_v0)
  master_table.pkl        sampled geometry (L, B, t, P_FF, P_SF, FFE)
  library_URM_floor{n}.pkl  URM resistance + cost arrays per floor
  library_components.pkl  non-URM component arrays (if AllComponents)
  simData_ds{0-3}.pkl     assembled component library per Conservation State
  downtime_model.pkl      pre-sampled downtime arrays (DS 0-4 x N_sim)
  demandMatrix.pkl        EUL demand arrays (first + second floor)
  demand_velocity.png     summary plot of mean demands vs. flood depth
"""

import tkinter as tk
from tkinter import ttk, messagebox

# ==============================================================================
# CLASS 1: SETTINGS GUI
# ==============================================================================
class InitGUI:
    """Collects generation settings; self.settings is None if user cancels."""

    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Flood Risk — Data Initialisation")
        self.settings = None

        self.defaults = {
            "AllComponents": True,
            "basement":      True,
            "N_SIM":         10000,
            "MAX_DEPTH":     10.0,
            "INCREMENTS":    0.01,
            "MAX_PLOT_DEPTH":7.0,
            "FLOORAREA":     100,
            "SEED":          42,
        }

        self.all_comp_var  = tk.BooleanVar(value=self.defaults["AllComponents"])
        self.basement_var  = tk.BooleanVar(value=self.defaults["basement"])
        self.mitigated_var = tk.BooleanVar(value=False)
        self.n_sim_var     = tk.StringVar(value=str(self.defaults["N_SIM"]))
        self.max_depth_var = tk.StringVar(value=str(self.defaults["MAX_DEPTH"]))
```

```python
        self.inc_var       = tk.StringVar(value=str(self.defaults["INCREMENTS"]))
        self.plot_var      = tk.StringVar(value=str(self.defaults["MAX_PLOT_DEPTH"]))
        self.area_var      = tk.StringVar(value=str(self.defaults["FLOORAREA"]))
        self.seed_var      = tk.StringVar(value=str(self.defaults["SEED"]))

        self._build_ui()
        self.root.mainloop()

    def _build_ui(self):
        f = ttk.Frame(self.root, padding="15")
        f.grid(row=0, column=0, sticky="nsew")

        ttk.Checkbutton(f, text="Evaluate All Components (URM + non-URM)",
variable=self.all_comp_var).grid(row=0, column=0, columnspan=2, sticky=tk.W)
        ttk.Checkbutton(f, text="Include Basement",
variable=self.basement_var ).grid(row=1, column=0, columnspan=2, sticky=tk.W)
        ttk.Checkbutton(f, text="Mitigated (flood-proofed components)",
variable=self.mitigated_var).grid(row=2, column=0, columnspan=2, sticky=tk.W)

        rows = [
            ("Number of Simulations",   self.n_sim_var),
            ("Max Flood Depth (m)",     self.max_depth_var),
            ("Depth Increments (m)",    self.inc_var),
            ("Max Plot Depth (m)",      self.plot_var),
            ("Floor Area (m²)",         self.area_var),
            ("Random Seed",             self.seed_var),
        ]
        for i, (label, var) in enumerate(rows, start=3):
            ttk.Label(f, text=label).grid(row=i, column=0, sticky=tk.W, pady=2)
            ttk.Entry(f, textvariable=var, width=12).grid(row=i, column=1, pady=2)

        ttk.Button(f, text="GENERATE DATA", command=self._on_run).grid(
            row=len(rows)+3, column=0, columnspan=2, pady=15)

    def _on_run(self):
        try:
            self.settings = {
                "AllComponents":  self.all_comp_var.get(),
                "basement":       self.basement_var.get(),
                "mitigated":      self.mitigated_var.get(),
                "N_SIM":          int(self.n_sim_var.get()),
                "MAX_DEPTH":      float(self.max_depth_var.get()),
                "INCREMENTS":     float(self.inc_var.get()),
                "MAX_PLOT_DEPTH": float(self.plot_var.get()),
                "FLOORAREA":      float(self.area_var.get()),
                "SEED":           int(self.seed_var.get()),
            }
            self.root.destroy()
        except ValueError:
            messagebox.showerror("Input Error", "Please check that all numerical fields are
valid.")

# ==============================================================================
# CLASS 2: COMPONENT LIBRARY BUILDER
# ==============================================================================
class ComponentLibraryBuilder:
    """
```

```python
    Generates N_sim Monte Carlo samples of flood resistance and repair cost for
    every building component.

    Non-URM components are read from a CSV lookup table.
    URM wall capacity is derived from out-of-plane yield-line theory, scaled by
    a Conservation State (CS0–CS3) capacity-reduction factor.
    """

    def __init__(self, n_sim, L_samples, B_samples, t_samples,
                 P_samples_ff, P_samples_sf, floors, run_dir):
        self.N_sim        = n_sim
        self.L_samples    = L_samples       # floor-to-floor height, mm
        self.B_samples    = B_samples       # facade width, mm
        self.t_samples    = t_samples       # wall thickness, mm
        self.P_samples_ff = P_samples_ff   # axial load first floor (ground level), kN/m
        self.P_samples_sf = P_samples_sf   # axial load second floor, kN/m
        self.floors       = floors
        self.run_dir      = run_dir
        print("Initializing Component Library Builder...")

    # --------------------------------------------------------------------------
    # SAMPLING HELPERS
    # --------------------------------------------------------------------------

    def _sample_truncnorm(self, bounds):
        """Sample N_sim values from a truncated normal (lower, upper, mean, std)."""
        lower, upper, mean, std = bounds
        a, b = (lower - mean) / std, (upper - mean) / std
        return truncnorm(a, b, loc=mean, scale=std).rvs(size=self.N_sim)

    def _sample_lognormal_from_bounds(self, bounds):
        """
        Lognormal whose ±2σ percentiles land at the given lower/upper bounds.
        Returns zeros if the bounds are degenerate (component already failed).
        """
        lower, upper = max(bounds[0], 1e-6), bounds[1]
        if lower >= upper:
            return np.zeros(self.N_sim)
        mu    = (np.log(lower) + np.log(upper)) / 2.0
        sigma = (np.log(upper) - np.log(lower)) / 4.0
        return np.random.lognormal(mu, sigma, size=self.N_sim)

    # --------------------------------------------------------------------------
    # NON-URM COMPONENTS
    # --------------------------------------------------------------------------

    def comp_nobrick_library(self, csv_path):
        """
        Build depth-resistance and repair-cost libraries for non-URM components
        from a CSV.  Columns 4–7: resistance bounds; columns 14–17: cost bounds.
        """
        if not os.path.exists(csv_path):
            print(f"Warning: {csv_path} not found.")
            return
        df = pd.read_csv(csv_path)
        vulndata = pd.DataFrame(index=df["Component"],
                                columns=["Depth Resistance", "Repair Costs"])
```

```python
    for k in range(df.shape[0]):
        vulndata.iloc[k, 0] = self._sample_truncnorm(df.iloc[k, 4:8].tolist())
        vulndata.iloc[k, 1] = self._sample_truncnorm(df.iloc[k, 14:18].tolist())
    vulndata.to_pickle(os.path.join(self.run_dir, "library_components.pkl"))

# --------------------------------------------------------------------------
# URM WALL CAPACITY
# --------------------------------------------------------------------------

def URM_resistance(self, cs, target_ds, floor):
    """
    N_sim samples of out-of-plane lateral capacity (kPa EUL) for a URM wall
    transitioning from cs to target_ds.

    Scaling law (yield-line theory):
        q = q_ref x (t/t_ref)² x (L_ref/L)² x (P/P_ref) x mat x eps
    where mat = lognormal material variability (COV 20%) and
          eps = lognormal model uncertainty (β increases with CS).
    Weakest pier governs.
    """
    t_ref, L_ref, P_ref = 100.0, 2700.0, 50.0

    q_ref_map            = {2: 1.162, 3: 1.359, 4: 1.359}  # EUL kN/m² at Z_ref=2.7 m
    cap_reduction_map    = {0: 1.0,  1: 1.0,  2: 0.8,  3: 0.6}
    model_uncertainty_map = {0: 0.25, 1: 0.25, 2: 0.35, 3: 0.50}

    q_ref_target      = q_ref_map[target_ds]
    cap_reduction     = cap_reduction_map[cs]
    model_uncertainty = model_uncertainty_map[cs]

    sigma_mat         = np.sqrt(np.log(1 + 0.20**2))
    mat_uncertainty   = lognorm.rvs(s=sigma_mat, scale=1.0, size=self.N_sim)
    q_ref_sim_target = cap_reduction * mat_uncertainty * q_ref_target
    eps_samples       = lognorm.rvs(s=model_uncertainty, scale=1.0, size=self.N_sim)

    B_door = np.random.normal(loc=900.0,  scale=50.0,  size=self.N_sim)
    B_win  = np.random.normal(loc=2000.0, scale=200.0, size=self.N_sim)

    # Clip openings so they never overflow the total wall width
    scale_factor = np.minimum(1.0, (self.B_samples - 1500.0) / (B_door + B_win))
    B_door_eff  = B_door * scale_factor
    B_win_eff   = B_win  * scale_factor
    B_remaining = self.B_samples - (B_door_eff + B_win_eff)

    weights = np.random.uniform(1.0, 3.0, size=(3, self.N_sim))
    B_piers = B_remaining * (weights / weights.sum(axis=0))

    P = self.P_samples_ff if floor == 0 else self.P_samples_sf
    P_pier1 = P * (B_piers[0] + 0.5 * B_door_eff) / B_piers[0]
    P_pier2 = P * (B_piers[1] + 0.5 * B_door_eff + 0.5 * B_win_eff) / B_piers[1]
    P_pier3 = P * (B_piers[2] + 0.5 * B_win_eff) / B_piers[2]

    base = lambda P_: (self.t_samples/t_ref)**2 * (L_ref/self.L_samples)**2 * (P_/P_ref)
    scalar_sys = np.minimum.reduce([base(P_pier1), base(P_pier2), base(P_pier3)])

    return q_ref_sim_target * scalar_sys * eps_samples
```

```python
def URM_library(self, floor):
    """
    Build the URM component library for one floor and save to pkl.
    DS1 uses a depth-threshold model; DS2–DS4 use physics-based capacity.
    Upper floors get a floor-height offset on the depth threshold.
    """
    URM_ds1 = [0.1, 0.5, 0.3, 0.1]  # (lower, upper, mean, std) in metres
    URM_repair_costs = {
        1: [468,  1170,  819,  175.5],
        2: [468,  2106, 1287,  409.5],
        3: [1170, 2574, 1872,  351  ],
        4: [7020, 17784, 12402, 2691 ],
    }

    rows = []
    for init_ds in range(4):
        for targ_ds in range(max(1, init_ds + 1), 5):
            name = f"Brick walls (DS{targ_ds}) | init={init_ds}"
            res  = (self._sample_truncnorm(URM_ds1) if targ_ds == 1
                    else self.URM_resistance(init_ds, targ_ds, floor))
            if floor != 0 and targ_ds == 1:
                res = res + (self.L_samples / 1000) * floor
            cost = self._sample_truncnorm(URM_repair_costs[targ_ds])
            rows.append(pd.DataFrame(
                {"Depth Resistance": [res], "Repair Costs": [cost]},
                index=[name]))

    pd.concat(rows).to_pickle(
        os.path.join(self.run_dir, f"library_URM_floor{floor}.pkl"))

def assemble_lib(self, cs, all_components=False):
    """
    Collect URM rows for cs across all floors; optionally prepend
    the non-URM component library.  Saves simData_cs{cs}.pkl.
    """
    dataframes = []
    if all_components:
        dataframes.append(
            pd.read_pickle(os.path.join(self.run_dir, "library_components.pkl")))

    tag = f"| init={cs}"
    for fl in self.floors:
        path = os.path.join(self.run_dir, f"library_URM_floor{fl}.pkl")
        if os.path.exists(path):
            df = pd.read_pickle(path)
            sub = df[df.index.str.contains(tag, regex=False)].copy()
            sub.index = sub.index.str.replace(f" {tag}", f" – Floor {fl}", regex=False)
            dataframes.append(sub)

    pd.concat(dataframes).to_pickle(
        os.path.join(self.run_dir, f"simData_cs{cs}.pkl"))

def build_all(self, all_components=False, csv_path="components_noURM.csv"):
    """Build all library pkl files for a fresh run."""
    print("Building component libraries...")
    for fl in self.floors:
        self.URM_library(fl)
```

```python
        if all_components:
            self.comp_nobrick_library(csv_path)
        for scenario in range(4):
            self.assemble_lib(scenario, all_components=all_components)
        print("Component libraries complete.")

# ==============================================================================
# CLASS 3: DOWNTIME MODEL
# ==============================================================================
class DowntimeModel:
    """
    Pre-samples N_sim total downtime values per post-hazard damage state (DS 0-4).

    Total downtime = max(impeding sequences) + repair time.
    Drying is assumed to be inherently included in the cleanup impeding sequences.

    Impeding factors (max over parallel paths)
    ------------------------------------------
    DS 0-2  cleanup sequences only
    DS 3-4  + permit / finance / contractor / URM lead-time sequences

    Repair actions
    --------------
    DS 0-1  none
    DS 2    repointing
    DS 3    repointing + crack repair
    DS 4    rebuild

    All times are stored in days.  Call save(run_dir) to persist.
    """

    _UNIT_TO_DAYS = {"day": 1, "week": 7, "month": 30}

    _DELAYS = {
        "flood_dur":            (2,    0.5, "day"),
        "cleanup_contr_unl":    (2,    0.5, "day"),
        "cleanup_contr_lim":    (12,   0.8, "day"),
        "cleanup_uncontr_unl": (3,    0.5, "week"),
        "cleanup_uncontr_lim": (1,    0.3, "month"),
        "permit_contr":         (7,    0.4, "week"),
        "permit_uncontr":       (12,   0.5, "week"),
        "finance_prearr":       (1,    0.5, "week"),
        "finance_privloan":     (15,   0.7, "week"),
        "finance_insurance":    (6,    0.5, "week"),
        "contractor_contr":     (1.5,  0.5, "week"),
        "contractor_uncontr":  (6,    0.5, "week"),
        "URM_longlead":                  (5,    0.5, "week"),
    }

    _REPAIR_TIMES = {
        "repointing":   (4, 0.2, "day"),
        "crack_repair": (3, 0.2, "day"),
        "rebuild":      (8, 0.2, "day"),
    }

    _REPAIR_ACTIONS = {0: [], 1: [], 2: [],
                       3: ["repointing", "crack_repair"], 4: ["rebuild"]}
```

```
    _SEQUENCES_LOW = {
        "seq1_a": ["flood_dur", "cleanup_contr_unl"],
        "seq1_b": ["flood_dur", "cleanup_contr_lim"],
        "seq1_c": ["flood_dur", "cleanup_uncontr_unl"],
        "seq1_d": ["flood_dur", "cleanup_uncontr_lim"],
    }
    _SEQUENCES_ALL = {
        **_SEQUENCES_LOW,
        "seq2_a": ["flood_dur", "permit_contr"],
        "seq2_b": ["flood_dur", "permit_uncontr"],
        "seq3_a": ["flood_dur", "finance_prearr"],
        "seq3_b": ["flood_dur", "finance_privloan"],
        "seq3_c": ["flood_dur", "finance_insurance"],
        "seq4_a": ["flood_dur", "contractor_contr",   "URM_longlead"],
        "seq4_b": ["flood_dur", "contractor_uncontr",  "URM_longlead"],
    }

    def __init__(self, n_sim, seed=42):
        self.n_sim = n_sim
        self.rng   = np.random.default_rng(seed)
        self._build()

    def _sample_delay(self, name, source):
        median, sigma, unit = source[name]
        return self.rng.lognormal(np.log(median), sigma, size=self.n_sim) *
self._UNIT_TO_DAYS[unit]

    def _sum_sequence(self, steps):
        return sum(self._sample_delay(s, self._DELAYS) for s in steps)

    def _sum_repair(self, ds):
        actions = self._REPAIR_ACTIONS[ds]
        if not actions:
            return np.zeros(self.n_sim)
        return sum(self._sample_delay(a, self._REPAIR_TIMES) for a in actions)

    def _build(self):
        """Sample everything once; store in self.downtime (DS → days array)."""
        imp_low  = np.stack([self._sum_sequence(s) for s in
self._SEQUENCES_LOW.values()]).max(axis=0)
        imp_high = np.stack([self._sum_sequence(s) for s in
self._SEQUENCES_ALL.values()]).max(axis=0)

        IMP = {0: imp_low, 1: imp_low, 2: imp_low, 3: imp_high, 4: imp_high}

        self.downtime = {ds: IMP[ds] + self._sum_repair(ds)
                         for ds in self._REPAIR_ACTIONS}

    def save(self, run_dir):
        """Persist downtime arrays to downtime_model.pkl (columns = DS 0–4)."""
        pd.DataFrame(self.downtime).to_pickle(
            os.path.join(run_dir, "downtime_model.pkl"))

    def print_summary(self):
        for ds, dt in self.downtime.items():
            p50, p90, p95 = np.percentile(dt, [50, 90, 95])
```

```python
            print(f"DS{ds}  mean={dt.mean():6.1f} d  P50={p50:6.1f} d  "
                  f"P90={p90:6.1f} d  P95={p95:6.1f} d")

# ==============================================================================
# CLASS 4: DEMAND GENERATOR
# ==============================================================================
class DemandGenerator:
    """
    Computes hydrostatic, hydrodynamic, and impact EUL demands for every
    flood depth x simulation combination and saves them to demandMatrix.pkl.

    All three load types are converted to an Equivalent Uniform Load (EUL)
    via bending-moment equivalence:
        w_eq = 8 · P · y · (Z-y) / Z³
    so they can be directly compared with the uniform-capacity q_ref.
    """

    def __init__(self, n_sim, L_samples, B_samples, FFE, depths, run_dir):
        self.N_sim     = n_sim
        self.L_samples = L_samples   # floor-to-floor height, mm
        self.B_samples = B_samples   # facade width, mm
        self.ffe       = FFE         # finished floor elevation, m
        self.depths    = depths      # flood depths to evaluate, m
        self.run_dir   = run_dir

    def generate(self, max_plot_depth=None):
        """
        Generate demand matrices for first and second floors, save to pkl,
        and produce a mean-demand summary plot.

        Hydrostatic – ground floor only, active for ffe < d <= ffe + Z + d_delta_max:
          f_sta_stable (uniform, below indoor level) + f_sta_tri (triangular above);
          both zero outside that range, no trapezoidal overshoot above wall.
        Hydrodynamic – 0.5 x Cd x V² x d_dyn; Cd from B/d aspect-ratio lookup (FEMA P-55).
        Impact       – depth-dependent debris coefficient x (W x V), 1 m width.
        """
        print("Generating demand matrices...")
        demands = {'first':  np.zeros((len(self.depths), self.N_sim)),
                   'second': np.zeros((len(self.depths), self.N_sim))}

        Z           = self.L_samples / 1000     # floor-to-floor height, m (per simulation)
        gamma_w     = 1025 * 9.81 / 1000         # Density of flood water in kg/m3 x g
converted to kN ==> kN/m³
        d_delta_max = 0.25                       # indoor-outdoor ingress lag, m

        C_drag    = [1.25, 1.3, 1.4, 1.5, 1.75, 1.8, 2.0]
        h_meters  = np.array([0.3048, 0.6096, 0.9144, 1.2192, 1.524])
        cd_values = np.array([0.0,    0.25,   0.5,    0.75,   1.0  ])

        mean_sta    = np.zeros(len(self.depths))
        mean_dyn    = np.zeros(len(self.depths))
        mean_imp    = np.zeros(len(self.depths))
        mean_sta_sf = np.zeros(len(self.depths))
        mean_dyn_sf = np.zeros(len(self.depths))
        mean_imp_sf = np.zeros(len(self.depths))
        mean_v      = np.zeros(len(self.depths))
```

```
        def eul(P, y, Z):
            """Convert a force P at height y on a wall of height Z to EUL."""
            M_max = (P * y * (Z - y)) / Z
            return (8 * M_max) / (Z**2)

        for i, d in enumerate(self.depths):
            if d == 0:
                continue

            ratios = (self.B_samples / 1000) / d
            C_drag_conds = [ratios <= 12, (ratios > 12) & (ratios <= 20),
                         (ratios > 20) & (ratios <= 32), (ratios > 32) & (ratios <= 40),
                         (ratios > 40) & (ratios <= 80), (ratios > 80) & (ratios <= 120),
                         ratios > 120]
            C_drag_array = np.select(C_drag_conds, C_drag, default=1.25)

            # Flow velocity: lognormal around a Manning-type mean, capped at 3 m/s
            V_mean  = 1.0 * (d ** (2/3))
            sigma_v = np.sqrt(np.log(1 + 0.20**2))
            mu_v    = np.log(V_mean) - 0.5 * sigma_v**2
            V_sim   = np.minimum(3.0, np.maximum(0, np.random.lognormal(mu_v, sigma_v,
self.N_sim)))

            # Debris impact — FEMA 259: F = W[lbf]·V[ft/s]·Cd·Cb·Cstr [lbf]
            # SI inputs (kg, m/s) → kN: multiply by 2.205 × 3.281 / 224.8 ≈ 0.0322
            C_depth = np.interp(d, h_meters, cd_values)
            C_b     = 0.2                                          # blockage coefficient
            C_str   = 0.2                                          # structure type
(masonry)
            W_sim   = np.random.uniform(453, 1088, size=self.N_sim)     # debris mass, kg
            F_imp   = W_sim * V_sim * C_depth * C_b * C_str * 0.0322  # kN/m

            d_out_ff = np.maximum(0.0, d - self.ffe)         # flood depth above FF floor
level
            d_out_sf = np.maximum(0.0, d - Z - self.ffe)     # flood depth above SF floor
level

            # Hydrostatic — first floor, active for 0 < d_out_ff <= Z + d_delta_max
            # Case 1 (d_out_ff <= d_delta_max): f_sta_stable = 0, f_sta_tri triangular only
            # Case 2 (d_out_ff >  d_delta_max): both components; d_delta_ff limited by
remaining wall
            d_in_ff    = np.minimum(Z, np.maximum(0.0, d_out_ff - d_delta_max))
            d_delta_ff = np.minimum(d_out_ff - d_in_ff, Z - d_in_ff)

            f_sta_stable = gamma_w * d_delta_ff * d_in_ff
            y_sta_stable = d_in_ff / 2.0

            f_sta_tri = 0.5 * gamma_w * d_delta_ff**2
            y_sta_tri = d_in_ff + d_delta_ff / 3.0

            f_sta  = f_sta_stable + f_sta_tri
            safe_f = np.where(f_sta > 0, f_sta, 1.0)
            y_sta  = np.where(f_sta > 0,
                              (f_sta_stable * y_sta_stable + f_sta_tri * y_sta_tri) / safe_f,
                              0.0)

            # Hydrostatic — second floor, same model applied to d_out_sf / d_in_sf
```

```
            d_in_sf    = np.minimum(Z, np.maximum(0.0, d_out_sf - d_delta_max))
            d_delta_sf = np.minimum(d_out_sf - d_in_sf, Z - d_in_sf)

            f_sta_stable_sf = gamma_w * d_delta_sf * d_in_sf
            y_sta_stable_sf = d_in_sf / 2.0

            f_sta_tri_sf = 0.5 * gamma_w * d_delta_sf**2
            y_sta_tri_sf = d_in_sf + d_delta_sf / 3.0

            f_sta_sf  = f_sta_stable_sf + f_sta_tri_sf
            safe_f_sf = np.where(f_sta_sf > 0, f_sta_sf, 1.0)
            y_sta_sf  = np.where(f_sta_sf > 0,
                                 (f_sta_stable_sf * y_sta_stable_sf + f_sta_tri_sf *
y_sta_tri_sf) / safe_f_sf,
                                 0.0)

            # Hydrodynamic & impact — first floor (ff) and second floor (sf)
            d_dyn_ff = np.minimum(Z, d_out_ff);  y_dyn_ff = d_dyn_ff / 2.0
            d_dyn_sf = np.minimum(Z, d_out_sf);  y_dyn_sf = d_dyn_sf / 2.0
            y_imp_ff = np.minimum(Z, d_out_ff)
            y_imp_sf = np.minimum(Z, d_out_sf)

            f_dyn_ff = 0.5 * C_drag_array * V_sim**2 * d_dyn_ff
            f_imp_ff = np.where((d_out_ff > 0) & (d_out_ff <= Z), F_imp, 0)

            demands['first'][i, :] = (eul(f_sta,    y_sta,    Z) +
                                      eul(f_dyn_ff, y_dyn_ff, Z) +
                                      eul(f_imp_ff, y_imp_ff, Z))

            f_dyn_sf = 0.5 * C_drag_array * V_sim**2 * d_dyn_sf
            f_imp_sf = np.where((d_out_sf > 0) & (d_out_sf <= Z), F_imp, 0)

            demands['second'][i, :] = (eul(f_sta_sf, y_sta_sf, Z) +
                                       eul(f_dyn_sf, y_dyn_sf, Z) +
                                       eul(f_imp_sf, y_imp_sf, Z))

            mean_sta[i]    = np.mean(eul(f_sta,    y_sta,    Z))
            mean_dyn[i]    = np.mean(eul(f_dyn_ff, y_dyn_ff, Z))
            mean_imp[i]    = np.mean(eul(f_imp_ff, y_imp_ff, Z))
            mean_sta_sf[i] = np.mean(eul(f_sta_sf, y_sta_sf, Z))
            mean_dyn_sf[i] = np.mean(eul(f_dyn_sf, y_dyn_sf, Z))
            mean_imp_sf[i] = np.mean(eul(f_imp_sf, y_imp_sf, Z))
            mean_v[i]      = np.mean(V_sim)

        # Summary plot — two panels (first floor / second floor), shared colour scheme
        COL = {'sta': 'C1', 'dyn': 'C2', 'imp': 'C3', 'tot': 'dimgray', 'vel': 'C0'}

        fig, (ax_ff, ax_sf) = plt.subplots(1, 2, figsize=(14, 5))

        # First floor
        ax_ff.set_title('First floor (FF)', fontsize=12)
        ax_ff.set_xlabel('Flood depth (m)', fontsize=11)
        ax_ff.set_ylabel('Demand (kN/m²)', fontsize=11)
        h_sta, = ax_ff.plot(self.depths, mean_sta,                       lw=2,
color=COL['sta'], label='Hydrostatic')
        h_dyn, = ax_ff.plot(self.depths, mean_dyn,                       lw=2,
color=COL['dyn'], label='Hydrodynamic')
```

```
        h_imp, = ax_ff.plot(self.depths, mean_imp,                       lw=2,
color=COL['imp'], label='Impact')
        h_tot, = ax_ff.plot(self.depths, mean_sta + mean_dyn + mean_imp, lw=2,
color=COL['tot'], label='Total')
        ax_ff.grid(True, alpha=0.5)
        ax2_ff = ax_ff.twinx()
        ax2_ff.set_ylabel('Flow velocity (m/s)', color=COL['vel'], fontsize=11)
        h_vel, = ax2_ff.plot(self.depths, mean_v, lw=1.5, color=COL['vel'], ls=':',
label='Velocity')
        ax2_ff.tick_params(axis='y', labelcolor=COL['vel'])
        if max_plot_depth is not None:
            ax_ff.set_xlim(0, max_plot_depth)

        # Second floor
        ax_sf.set_title('Second floor (SF)', fontsize=12)
        ax_sf.set_xlabel('Flood depth (m)', fontsize=11)
        ax_sf.set_ylabel('Demand (kN/m²)', fontsize=11)
        ax_sf.plot(self.depths, mean_sta_sf,                              lw=2,
color=COL['sta'])
        ax_sf.plot(self.depths, mean_dyn_sf,                              lw=2,
color=COL['dyn'])
        ax_sf.plot(self.depths, mean_imp_sf,                              lw=2,
color=COL['imp'])
        ax_sf.plot(self.depths, mean_sta_sf + mean_dyn_sf + mean_imp_sf, lw=2,
color=COL['tot'])
        ax_sf.grid(True, alpha=0.5)
        ax2_sf = ax_sf.twinx()
        ax2_sf.set_ylabel('Flow velocity (m/s)', color=COL['vel'], fontsize=11)
        ax2_sf.plot(self.depths, mean_v, lw=1.5, color=COL['vel'], ls=':')
        ax2_sf.tick_params(axis='y', labelcolor=COL['vel'])
        if max_plot_depth is not None:
            ax_sf.set_xlim(0, max_plot_depth)

        # Equalise demand and velocity scales across both panels
        demand_ymax = max(np.max(mean_sta + mean_dyn + mean_imp),
                          np.max(mean_sta_sf + mean_dyn_sf + mean_imp_sf)) * 1.05
        vel_ymax    = np.max(mean_v) * 1.05
        ax_ff.set_ylim(0, demand_ymax)
        ax_sf.set_ylim(0, demand_ymax)
        ax2_ff.set_ylim(0, vel_ymax)
        ax2_sf.set_ylim(0, vel_ymax)

        # Shared legend below both panels; bbox_inches='tight' ensures it is not clipped
        fig.legend(handles=[h_sta, h_dyn, h_imp, h_tot, h_vel],
                   loc='lower center', ncol=5, bbox_to_anchor=(0.5, -0.04), fontsize=10)
        fig.tight_layout()
        fig.subplots_adjust(bottom=0.20)
        plt.savefig(os.path.join(self.run_dir, "demand_velocity.png"), dpi=300,
bbox_inches='tight')
        plt.close()

        pd.to_pickle(demands, os.path.join(self.run_dir, "demandMatrix.pkl"))
        print("Demand matrices saved.")

# ===============================================================================
# DIAGNOSTICS
# ===============================================================================
```

```python
def run_diagnostics(run_dir, depths):
    """
    Post-generation diagnostic: prints demand statistics and capacity-vs-demand
    overlap for all URM DS2-DS4 components across CS 0-3.
    Warns if P(fail) is 0 or 1 across all depths (unit mismatch / calibration error).
    """
    import numpy as np
    import pandas as pd
    import os

    dm = pd.read_pickle(os.path.join(run_dir, "demandMatrix.pkl"))

    # Representative depths: ankle / knee / chest / head level
    target_m  = [0.5, 1.0, 2.0, 3.0]
    rep_pairs = [(td, np.argmin(np.abs(depths - td)))
                 for td in target_m if td <= depths.max()]

    print("\n" + "="*72)
    print("DIAGNOSTICS — DEMAND STATISTICS (first floor)")
    print("="*72)
    print(f"  {'Depth (m)':<12} {'Mean':>9} {'p5':>9} {'p95':>9}  (kN/m²)")
    print("  " + "-"*46)
    zero_max = np.max(np.abs(dm['first'][0, :]))
    print(f"  {'0.00':<12} {'0.000':>9} {'0.000':>9} {'0.000':>9}"
          f"  [demand@depth=0 max={zero_max:.2e}]")
    for td, idx in rep_pairs:
        row = dm['first'][idx, :]
        print(f"  {td:<12.2f} {row.mean():>9.4f} {np.percentile(row, 5):>9.4f}"
              f" {np.percentile(row, 95):>9.4f}")

    print()
    FLOOR_KEY = {"Floor 0": "first", "Floor 1": "second"}

    for cs in range(4):
        path = os.path.join(run_dir, f"simData_cs{cs}.pkl")
        if not os.path.exists(path):
            print(f"  [SKIP] simData_cs{cs}.pkl not found")
            continue
        sd  = pd.read_pickle(path)
        urm = sd[sd.index.str.contains(r"Brick walls \(DS[2-4]\)", regex=True)]
        if urm.empty:
            continue

        print(f"CS {cs}  ({len(urm)} URM DS2-4 components)")
        hdr = (f"  {'Component':<38} {'Cap μ':>7} {'σ':>7} {'p5':>7} {'p95':>7}  "
               + "  ".join(f"d={td:.1f}" for td, _ in rep_pairs))
        print(hdr)
        print("  " + "-"*80)

        for name in urm.index:
            cap   = np.asarray(sd.loc[name, "Depth Resistance"], dtype=float)
            floor = "Floor 0" if "Floor 0" in name else "Floor 1"
            dk    = FLOOR_KEY[floor]

            p_reps = [np.mean(dm[dk][idx, :] > cap) for _, idx in rep_pairs]

            # Full-range warning check (strided for speed)
```

```
            stride = max(1, len(depths) // 200)
            p_all  = [np.mean(dm[dk][i, :] > cap) for i in range(0, len(depths), stride)]
            warn = ""
            if max(p_all) == 0.0:
                warn = "  *** WARNING: P(fail)=0 everywhere — check units ***"
            elif min(p_all) == 1.0:
                warn = "  *** WARNING: P(fail)=1 everywhere — check calibration ***"

            short     = name[:38]
            pfail_str = "  ".join(f"{p:>6.3f}" for p in p_reps)
            print(f"  {short:<38} {cap.mean():>7.3f} {cap.std():>7.3f}"
                  f" {np.percentile(cap,5):>7.3f} {np.percentile(cap,95):>7.3f}"
                  f"  {pfail_str}{warn}")
        print()

    print("="*72 + "\n")

# ==============================================================================
# MAIN — DATA GENERATION PIPELINE
# ==============================================================================
if __name__ == "__main__":
    gui = InitGUI()
    if not gui.settings:
        print("Cancelled.")
        exit()

    print("Loading libraries...")
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import truncnorm, lognorm
    import pandas as pd
    import os
    import datetime

    cfg = gui.settings

    # ── Create timestamped output folder ──────────────────────────────────────
    timestamp = datetime.datetime.now().strftime("%d%m_%H%M")
    RUN_DIR   = f"results_{timestamp}"
    os.makedirs(RUN_DIR, exist_ok=True)
    print(f"Output folder: {RUN_DIR}")

    # ── Seed the global RNG for reproducibility ───────────────────────────────
    np.random.seed(cfg["SEED"])

    # ── Geometry sampling (dimensions in mm, loads in kN/m) ──────────────────
    # Sources: RVO/Tabula Dutch terraced house archetype, DGMR structural survey
    N_SIM   = cfg["N_SIM"]
    FLOORS  = [0, 1]
    DEPTHS  = np.arange(0, cfg["MAX_DEPTH"] + cfg["INCREMENTS"], cfg["INCREMENTS"])

    MASTER_L    = truncnorm(a=(2600-2800)/100, b=(3000-2800)/100, loc=2800,
scale=100).rvs(N_SIM)
    MASTER_B    = truncnorm(a=(4500-5100)/300, b=(5700-5100)/300, loc=5100,
scale=300).rvs(N_SIM)
    MASTER_T    = truncnorm(a=(205-220)/7.5,   b=(235-220)/7.5,   loc=220,
scale=7.5).rvs(N_SIM)
```

```
    # MASTER_T  = truncnorm(a=(105-115)/5,     b=(125-115)/5,     loc=115,
scale=5  ).rvs(N_SIM)  # single-wythe
    MASTER_P_FF = truncnorm(a=(95-105)/5,      b=(115-105)/5,     loc=105,
scale=5 ).rvs(N_SIM)
    MASTER_P_SF = truncnorm(a=(45-50)/2.5,     b=(55-50)/2.5,     loc=50,
scale=2.5 ).rvs(N_SIM)

    if cfg["basement"]:
        MASTER_FFE = truncnorm(a=(0.5-0.75)/0.125, b=(1-0.75)/0.125,
                               loc=0.75, scale=0.125).rvs(N_SIM)
        CSV = ("components_incl-basement_mitigated.csv" if cfg["mitigated"]
               else "components_incl-basement_original.csv")
    else:
        MASTER_FFE = np.random.uniform(0.05, 0.10, size=N_SIM)
        CSV        = ("components_no-basement_mitigated.csv" if cfg["mitigated"]
                      else "components_no-basement_original.csv")

    # Save geometry for reproducibility and postprocess loading
    pd.DataFrame({"L": MASTER_L, "B": MASTER_B, "t": MASTER_T,
                  "P_FF": MASTER_P_FF, "P_SF": MASTER_P_SF, "FFE": MASTER_FFE},
                 index=pd.RangeIndex(N_SIM, name="sim_ID")
                 ).to_pickle(os.path.join(RUN_DIR, "master_table.pkl"))

    # Save run config so postprocess can reconstruct DEPTHS and other params
    pd.to_pickle({
        "N_SIM":      N_SIM,
        "MAX_DEPTH":  cfg["MAX_DEPTH"],
        "INCREMENTS": cfg["INCREMENTS"],
        "FLOORAREA":  cfg["FLOORAREA"],
        "SEED":       cfg["SEED"],
    }, os.path.join(RUN_DIR, "run_config.pkl"))

    # ── Geometry histograms ──────────────────────────────────────────────
    fig_geo, axes_geo = plt.subplots(2, 3, figsize=(14, 7))
    _geo_vars = [
        (MASTER_L,    "L (mm)"),
        (MASTER_B,    "B (mm)"),
        (MASTER_T,    "t (mm)"),
        (MASTER_P_FF, "P_FF (kN/m)"),
        (MASTER_P_SF, "P_SF (kN/m)"),
        (MASTER_FFE,  "FFE (m)"),
    ]
    for ax, (data, label) in zip(axes_geo.flat, _geo_vars):
        ax.hist(data, bins=50, edgecolor="none", alpha=0.75)
        ax.axvline(data.mean(), color="k", linestyle="--", linewidth=1.2)
        ax.set_title(f"{label}  μ={data.mean():.3g} ± {data.std():.3g}", fontsize=9)
        ax.set_xlabel(label, fontsize=8)
        ax.set_ylabel("Count", fontsize=8)
    fig_geo.tight_layout()
    plt.savefig(os.path.join(RUN_DIR, "geometry_histograms.png"), dpi=300,
bbox_inches="tight")
    plt.close(fig_geo)
    print("Geometry histograms saved.")

    # ── Step 1: Component libraries ──────────────────────────────────────
    builder = ComponentLibraryBuilder(
        N_SIM, MASTER_L, MASTER_B, MASTER_T,
```

```
        MASTER_P_FF, MASTER_P_SF, FLOORS, RUN_DIR)
    builder.build_all(all_components=cfg["AllComponents"], csv_path=CSV)

    # ── Step 2: Downtime model ────────────────────────────────────────────
    print("Building downtime model...")
    dt_model = DowntimeModel(n_sim=N_SIM, seed=cfg["SEED"])
    dt_model.save(RUN_DIR)
    dt_model.print_summary()

    # ── Step 3: Demand matrices ───────────────────────────────────────────
    DemandGenerator(N_SIM, MASTER_L, MASTER_B, MASTER_FFE, DEPTHS, RUN_DIR
                    ).generate(max_plot_depth=cfg["MAX_PLOT_DEPTH"])

    run_diagnostics(RUN_DIR, DEPTHS)

    print(f"\n=== DATA GENERATION COMPLETE — results in {RUN_DIR}/ ===")
"""
postprocess_v0.py — Post-processing pipeline for the Dutch URM flood risk model.

Loads pkl files produced by initialize_v0.py, evaluates component failures and
losses per Conservation State scenario, and generates vulnerability / downtime
plots.  No sampling is performed here; all randomness is locked in the pkl files.

Usage
-----
  1. Run initialize_v0.py to generate a results folder.
  2. Run this script, enter the results folder path in the GUI, choose scenarios
     and plot options, then click PROCESS.

Classes
-------
  PostProcessGUI    — tkinter window for post-processing settings.
  FloodRiskEvaluator — loads pkl data, evaluates losses, produces plots.
"""

import tkinter as tk
from tkinter import ttk, filedialog, messagebox

# ==============================================================================
# CLASS 1: POST-PROCESSING GUI
# ==============================================================================
class PostProcessGUI:
    """Collects post-processing settings; self.settings is None if user cancels."""

    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Flood Risk — Post-Processing")
        self.settings = None

        self.run_dir_var       = tk.StringVar(value="")
        self.recompute_var     = tk.BooleanVar(value=True)
        self.plot_depth_min_var= tk.StringVar(value="0.0")
        self.plot_depth_max_var= tk.StringVar(value="7.0")

        self.scenario_vars = {ds: tk.BooleanVar(value=True) for ds in range(4)}
        self.plot_vars = {
            "Component Fragility Curves": tk.BooleanVar(value=True),
```

```
            "Damage Ratio":               tk.BooleanVar(value=True),
            "Losses in Dollars ($/m²)":   tk.BooleanVar(value=True),
            "Losses in Euros (€/m²)":     tk.BooleanVar(value=True),
        }

        self._build_ui()
        self.root.mainloop()

    def _build_ui(self):
        f = ttk.Frame(self.root, padding="15")
        f.grid(row=0, column=0, sticky="nsew")
        row = 0

        # Run directory
        ttk.Label(f, text="Results folder:", font=('Helvetica', 10, 'bold')).grid(
            row=row, column=0, sticky=tk.W, pady=(0, 2)); row += 1
        dir_frame = ttk.Frame(f)
        dir_frame.grid(row=row, column=0, columnspan=2, sticky="ew"); row += 1
        ttk.Entry(dir_frame, textvariable=self.run_dir_var, width=30).pack(side=tk.LEFT)
        ttk.Button(dir_frame, text="Browse…", command=self._browse).pack(side=tk.LEFT, padx=5)

        ttk.Separator(f, orient="horizontal").grid(row=row, column=0, columnspan=2,
                                                    sticky="ew", pady=8); row += 1

        # Scenarios
        ttk.Label(f, text="Scenarios to process:", font=('Helvetica', 10, 'bold')).grid(
            row=row, column=0, sticky=tk.W); row += 1
        for ds, var in self.scenario_vars.items():
            ttk.Checkbutton(f, text=f"CS {ds}", variable=var).grid(
                row=row, column=0, sticky=tk.W); row += 1

        ttk.Separator(f, orient="horizontal").grid(row=row, column=0, columnspan=2,
                                                    sticky="ew", pady=8); row += 1

        # Options
        ttk.Checkbutton(f, text="Re-compute scenario losses (uncheck to re-plot only)",
                        variable=self.recompute_var).grid(row=row, column=0,
                        columnspan=2, sticky=tk.W); row += 1

        ttk.Label(f, text="Min plot depth (m):").grid(row=row, column=0, sticky=tk.W, pady=4)
        ttk.Entry(f, textvariable=self.plot_depth_min_var, width=8).grid(row=row, column=1,
                                                                          sticky=tk.W); row +=
1
        ttk.Label(f, text="Max plot depth (m):").grid(row=row, column=0, sticky=tk.W, pady=4)
        ttk.Entry(f, textvariable=self.plot_depth_max_var, width=8).grid(row=row, column=1,
                                                                       sticky=tk.W); row += 1

        ttk.Separator(f, orient="horizontal").grid(row=row, column=0, columnspan=2,
                                                    sticky="ew", pady=8); row += 1

        # Plot options
        ttk.Label(f, text="Plot options:", font=('Helvetica', 10, 'bold')).grid(
            row=row, column=0, sticky=tk.W); row += 1
        for label, var in self.plot_vars.items():
            ttk.Checkbutton(f, text=label, variable=var).grid(
                row=row, column=0, sticky=tk.W); row += 1
```

```python
        ttk.Separator(f, orient="horizontal").grid(row=row, column=0, columnspan=2,
                                                    sticky="ew", pady=8); row += 1
        ttk.Button(f, text="PROCESS", command=self._on_run).grid(
            row=row, column=0, columnspan=2, pady=5)

    def _browse(self):
        folder = filedialog.askdirectory(title="Select results folder")
        if folder:
            self.run_dir_var.set(folder)

    def _on_run(self):
        run_dir = self.run_dir_var.get().strip()
        if not run_dir:
            messagebox.showerror("Missing input", "Please select a results folder.")
            return
        try:
            min_plot_depth = float(self.plot_depth_min_var.get())
            max_plot_depth = float(self.plot_depth_max_var.get())
        except ValueError:
            messagebox.showerror("Input Error", "Plot depth values must be numbers.")
            return

        self.settings = {
            "run_dir":       run_dir,
            "scenarios":     [ds for ds, v in self.scenario_vars.items() if v.get()],
            "recompute":     self.recompute_var.get(),
            "min_plot_depth":min_plot_depth,
            "max_plot_depth":max_plot_depth,
            "comp_cdf":      self.plot_vars["Component Fragility Curves"].get(),
            "ratio":         self.plot_vars["Damage Ratio"].get(),
            "dollars":       self.plot_vars["Losses in Dollars ($/m²)"].get(),
            "euros":         self.plot_vars["Losses in Euros (€/m²)"].get(),
        }
        self.root.destroy()

# ==============================================================================
# CLASS 2: FLOOD RISK EVALUATOR
# ==============================================================================
class FloodRiskEvaluator:
    """
    Evaluates flood risk from pre-generated pkl files.

    On construction, loads demandMatrix.pkl and downtime_model.pkl from run_dir.
    No sampling is performed — all arrays are deterministic given the pkl files.

    Workflow per scenario (compute_scenario_loss):
      1. Load simData_ds{cs}.pkl (component resistance + repair costs)
      2. For each depth: compare demand vs. capacity; accumulate failures
      3. Sum repair costs; apply DS4 collapse override
      4. Derive post-hazard DS; look up pre-sampled downtime from pkl
      5. Save lossMatrix, downtimeMatrix, damageMatrix pkl files
    """

    def __init__(self, n_sim, FFE, depths, area, run_dir, res_dir):
        self.N_sim    = n_sim
        self.ffe      = FFE       # finished floor elevation, m, shape (N_sim,)
        self.depths   = depths
```

```python
        self.area    = area      # floor area m², for $/m² normalisation
        self.run_dir = run_dir
        self.res_dir = res_dir

        self.demand_matrices = pd.read_pickle(os.path.join(run_dir, "demandMatrix.pkl"))
        # downtime_df: columns are DS integers 0–4, rows are simulations
        self.downtime_df = pd.read_pickle(os.path.join(run_dir, "downtime_model.pkl"))

        print("Flood Risk Evaluator ready.")

    # ------------------------------------------------------------------------
    # HELPERS
    # ------------------------------------------------------------------------

    def _sample_truncnorm(self, bounds):
        lower, upper, mean, std = bounds
        a, b = (lower - mean) / std, (upper - mean) / std
        return truncnorm(a, b, loc=mean, scale=std).rvs(size=self.N_sim)

    def _get_downtime(self, post_hazard_ds):
        """
        Return (N_sim,) downtime array (days) by indexing each simulation's
        pre-sampled distribution for its own post-hazard DS.
        """
        result = np.zeros(self.N_sim)
        for ds in range(5):
            mask = (post_hazard_ds == ds)
            if np.any(mask):
                result[mask] = self.downtime_df[ds].values[mask]
        return result

    # ------------------------------------------------------------------------
    # SCENARIO LOSS COMPUTATION
    # ------------------------------------------------------------------------

    def compute_scenario_loss(self, cs):
        """
        Evaluate losses and downtime for one Conservation State scenario and
        save lossMatrix, downtimeMatrix, damageMatrix pkl files to run_dir.
        """
        print(f"  > Computing losses for CS {cs}...")

        simdata     = pd.read_pickle(os.path.join(self.run_dir, f"simData_cs{cs}.pkl"))
        res_matrix  = np.stack(simdata.iloc[:, 0].values)   # (n_comps, N_sim)
        cost_matrix = np.stack(simdata.iloc[:, 1].values)

        demand_ff = self.demand_matrices['first']
        demand_sf = self.demand_matrices['second']

        n_comps             = res_matrix.shape[0]
        cumulative_failures = np.zeros_like(res_matrix, dtype=bool)
        f_probs_matrix      = np.zeros((n_comps, len(self.depths)))
        loss_results        = {}
        downtime_results    = {}
        ds_dist_results     = {}  # depth -> fraction of sims per post-hazard DS

        # Component type masks
```

```
        is_ds1to3  = simdata.index.str.contains(r'Brick walls \(DS[1-3]\)')
        is_ds4     = simdata.index.str.contains(r'Brick walls \(DS4\)')
        is_urm     = is_ds1to3 | is_ds4
        is_ds1     = simdata.index.str.contains(r'Brick walls \(DS1\)', regex=True)
        is_ff_urm_depth = is_urm & is_ds1 & simdata.index.str.contains(r'Floor 0')
        is_sf_urm_depth = is_urm & is_ds1 & simdata.index.str.contains(r'Floor 1')
        is_ff_urm_force = is_urm & ~is_ds1 & simdata.index.str.contains(r'Floor 0')
        is_sf_urm_force = is_urm & ~is_ds1 & simdata.index.str.contains(r'Floor 1')
        is_non_urm = ~is_urm

        urm_indices_by_ds = {
            ds: np.where(simdata.index.str.contains(fr'Brick walls \(DS{ds}\)'))[0]
            for ds in [1, 2, 3, 4]
        }

        # ── Pre-compute sunk costs from floor library files ───────────────────
        # For CS ≥ 1, DS1..CS rows are filtered out of simData by assemble_lib
        # (pre-existing damage). We read them from the floor library to add their
        # repair costs when water first contacts the wall (depth trigger).
        # cumulative_failures is never modified — only the loss total is affected.
        sunk_data = {}  # fl -> {'depth_thr': ndarray, 'cost': ndarray, 'done': bool array}
        if cs >= 1:
            for fl in (0, 1):
                lib_path = os.path.join(self.run_dir, f"library_URM_floor{fl}.pkl")
                if not os.path.exists(lib_path):
                    continue
                lib = pd.read_pickle(lib_path)
                ds1_match = lib[lib.index == "Brick walls (DS1) | init=0"]
                if len(ds1_match) == 0:
                    continue
                depth_thr  = ds1_match.iloc[0, 0]          # (N_sim,) depth threshold in m
                total_cost = np.zeros(self.N_sim)
                for pre_ds in range(1, cs + 1):
                    row_key   = f"Brick walls (DS{pre_ds}) | init={pre_ds - 1}"
                    row_match = lib[lib.index == row_key]
                    if len(row_match) > 0:
                        total_cost += row_match.iloc[0, 1]  # (N_sim,) cost samples
                if total_cost.sum() > 0:
                    sunk_data[fl] = {
                        'depth_thr': depth_thr,
                        'cost':      total_cost,
                        'done':      np.zeros(self.N_sim, dtype=bool),
                    }
        cumulative_sunk = np.zeros(self.N_sim)

        for i, de in enumerate(self.depths):
            # ── Demand vs. capacity ───────────────────────────────────────────
            eff_depth = de - self.ffe if de > 0 else np.full(self.N_sim, -9999.0)

            cur = np.zeros_like(res_matrix, dtype=bool)
            if np.any(is_non_urm):      cur[is_non_urm,      :] = eff_depth >
res_matrix[is_non_urm, :]
            if np.any(is_ff_urm_depth): cur[is_ff_urm_depth, :] = eff_depth >
res_matrix[is_ff_urm_depth, :]
            if np.any(is_sf_urm_depth): cur[is_sf_urm_depth, :] = eff_depth >
res_matrix[is_sf_urm_depth, :]
            if np.any(is_ff_urm_force): cur[is_ff_urm_force, :] = demand_ff[i, :] >
```

```python
res_matrix[is_ff_urm_force, :]
            if np.any(is_sf_urm_force): cur[is_sf_urm_force, :] = demand_sf[i, :] >
res_matrix[is_sf_urm_force, :]

            cumulative_failures |= cur
            f_probs_matrix[:, i] = cumulative_failures.mean(axis=1)

            # ── Sunk cost: add pre-existing DS costs on first water contact ──
            for sd in sunk_data.values():
                newly = (eff_depth > sd['depth_thr']) & ~sd['done']
                if newly.any():
                    sd['done']       |= newly
                    cumulative_sunk  += sd['cost'] * newly

            # ── Repair costs ──────────────────────────────────────────────
            costs = cumulative_failures * cost_matrix

            # DS4 collapse override: zero out DS1–DS3 costs for collapsed sims
            sims_collapsed = np.any(cumulative_failures[is_ds4, :], axis=0)
            if np.any(is_ds1to3) and np.any(sims_collapsed):
                costs[np.ix_(np.where(is_ds1to3)[0], np.where(sims_collapsed)[0])] = 0

            # Sunk cost is also subsumed by DS4 replacement
            effective_sunk = cumulative_sunk.copy()
            effective_sunk[sims_collapsed] = 0

            loss_results[de] = np.sum(costs, axis=0) + effective_sunk

            # ── Post-hazard damage state ──────────────────────────────────
            water_reached = (de > self.ffe)

            flood_ds = np.zeros(self.N_sim, dtype=int)
            for ds in [4, 3, 2, 1]:
                idx = urm_indices_by_ds[ds]
                if len(idx) > 0:
                    hit  = np.any(cumulative_failures[idx, :], axis=0)
                    mask = hit & (flood_ds == 0)
                    flood_ds[mask] = ds

            post_hazard_ds = np.zeros(self.N_sim, dtype=int)
            if np.any(water_reached):
                baseline = max(cs, 1)
                post_hazard_ds[water_reached] = np.maximum(
                    baseline, flood_ds[water_reached])

            downtime_results[de] = self._get_downtime(post_hazard_ds)
            ds_dist_results[de]  = {ds: (post_hazard_ds == ds).sum() / self.N_sim
                                    for ds in range(5)}

        # ── Save results ──────────────────────────────────────────────────
        pd.DataFrame(loss_results).to_pickle(
            os.path.join(self.run_dir, f"lossMatrix_cs{cs}.pkl"))
        pd.DataFrame(downtime_results).to_pickle(
            os.path.join(self.run_dir, f"downtimeMatrix_cs{cs}.pkl"))
        # DS distribution: rows = DS 0-4, columns = depths, values = fraction of N_sim
        pd.DataFrame(ds_dist_results).to_pickle(
            os.path.join(self.run_dir, f"dsMatrix_cs{cs}.pkl"))
```

```python
        simdata.insert(2, "f_prob", list(f_probs_matrix))
        simdata.to_pickle(
            os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl"))

    # ---------------------------------------------------------------------------
    # HELPER — MAXIMUM REPLACEMENT COST
    # ---------------------------------------------------------------------------

    def calculate_max_costs(self, cs):
        """
        Maximum per-simulation replacement cost: sum of all component costs
        excluding DS1-DS3 URM rows (subsumed by DS4 rebuild; would double-count).
        """
        simdata     = pd.read_pickle(os.path.join(self.run_dir, f"simData_cs{cs}.pkl"))
        cost_matrix = np.stack(simdata.iloc[:, 1].values)
        is_ds1to3   = simdata.index.str.contains(r'Brick walls \(DS[1-3]\)')
        return np.sum(cost_matrix[~is_ds1to3, :], axis=0)

    # ---------------------------------------------------------------------------
    # PLOTTING
    # ---------------------------------------------------------------------------

    def compute_global_ylims(self, scenarios):
        """Return (dollar_ymax, euro_ymax) as shared upper bounds across all CS scenarios."""
        euro_bounds = (1318, 1891, (1318+1891)/2, (1891-1318)/4)
        dollar_max = 0.0
        euro_max   = 0.0
        for cs in scenarios:
            df_loss   = pd.read_pickle(os.path.join(self.run_dir, f"lossMatrix_cs{cs}.pkl"))
            max_costs = self.calculate_max_costs(cs)
            ratios    = df_loss.div(max_costs, axis=0)
            price     = ratios.mul(self._sample_truncnorm(euro_bounds), axis=0)
            dollar_max = max(dollar_max, (df_loss.quantile(0.975) / self.area).max())
            euro_max   = max(euro_max, price.quantile(0.975).max())
        return dollar_max * 1.05, euro_max * 1.05

    def plot_comp_fragility(self, cs, min_plot_depth=0, max_plot_depth=None):
        """
        Plot non-structural component fragility curves (failure probability vs. depth).
        Non-URM components are CS-independent, so this only needs to be called once.
        """
        df_fail = pd.read_pickle(os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl"))
        is_non_urm = ~df_fail.index.str.contains(r'Brick walls \(DS[2-4]\)', regex=True)
        non_urm_idx = np.where(is_non_urm)[0]
        if len(non_urm_idx) == 0:
            return
        plt.figure(figsize=(10, 6))
        for k in non_urm_idx:
            plt.plot(self.depths, df_fail.iloc[k, 2], lw=2, label=df_fail.index[k])
        plt.title('Non-structural component fragility', fontsize=14)
        plt.xlabel('Flood depth (m)', fontsize=12)
        plt.ylabel('Failure probability', fontsize=12)
        plt.ylim(0, 1.05)
        plt.grid(True, alpha=0.5)
        plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
        if max_plot_depth:
            plt.xlim(min_plot_depth, max_plot_depth)
```

```python
    plt.savefig(os.path.join(self.res_dir, "compFragCurves.png"),
                dpi=300, bbox_inches='tight')
    plt.close()
    print('  Saved: compFragCurves.png')

def plot_lossCurves(self, cs, comp_cdf=True, ratio=True,
                    dollars=True, euros=True,
                    max_plot_depth=None, min_plot_depth=0,
                    dollar_ylim=None, euro_ylim=None):
    """
    Plot vulnerability and downtime curves for one CS scenario.
    Saves PNG files to res_dir.
    """
    print(f"  > Plotting curves for CS {cs}...")
    df_loss = pd.read_pickle(os.path.join(self.run_dir, f"lossMatrix_cs{cs}.pkl"))
    df_fail = pd.read_pickle(os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl"))
    df_down = pd.read_pickle(os.path.join(self.run_dir, f"downtimeMatrix_cs{cs}.pkl"))

    xlim = (min_plot_depth, max_plot_depth) if max_plot_depth else None

    is_urm_force = df_fail.index.str.contains(r'Brick walls \(DS[2-4]\)', regex=True)

    if comp_cdf:
        # --- URM walls (DS2–DS4): empirical CDF of capacity samples in demand space ---
        # x-axis is lateral pressure (kN/m² EUL), so the curve shows what demand is
        # required to exceed each percentile of the capacity distribution.
        urm_force_idx = np.where(is_urm_force)[0]
        if len(urm_force_idx) > 0:
            # Fixed color and label per damage state — same in both floor panels
            DS_COL = {'DS2': 'C0', 'DS4': 'C2'}
            DS_LBL = {'DS2': 'DS2 (Moderate)', 'DS4': 'DS3 and DS4'}

            fig, (ax_ff, ax_sf) = plt.subplots(1, 2, figsize=(16, 6), sharey=True)
            fig.suptitle(f'URM Fragility Curve — CS{cs}', fontsize=14)

            for ax in (ax_ff, ax_sf):
                for p_ref, lbl in [(0.50, 'p50'), (0.95, 'p95')]:
                    ax.axhline(p_ref, color='grey', ls=':', lw=0.8, alpha=0.6)
                    ax.text(0.01, p_ref + 0.01, lbl,
                            transform=ax.get_yaxis_transform(),
                            fontsize=7, color='grey', va='bottom')

            for k in urm_force_idx:
                name = df_fail.index[k]
                cap  = np.sort(df_fail.iloc[k, 0])
                cdf  = np.arange(1, len(cap) + 1) / len(cap)

                if   '(DS2)' in name: ds_key = 'DS2'
                elif '(DS3)' in name: continue
                else:                 ds_key = 'DS4'
                color = DS_COL[ds_key]

                if 'Floor 0' in name:
                    ax_ff.plot(cap, cdf, lw=2, color=color, label=DS_LBL[ds_key])
                    _add_percentile_markers(ax_ff, cap, cdf, color, xlim_cap=15)
                else:
                    ax_sf.plot(cap, cdf, lw=2, color=color, label=DS_LBL[ds_key])
```

```
                _add_percentile_markers(ax_sf, cap, cdf, color, xlim_cap=15)

            for ax, title in ((ax_ff, 'First floor (FF)'), (ax_sf, 'Second floor (SF)')):
                ax.set_title(title, fontsize=12)
                ax.set_xlabel('Lateral capacity – EUL (kN/m²)', fontsize=11)
                ax.set_xlim(0, 15)
                ax.set_ylim(0, 1.05)
                ax.grid(True, alpha=0.5)
                ax.legend(loc='lower right', fontsize=10)
            ax_ff.set_ylabel('P(capacity ≤ demand)', fontsize=11)

            fig.tight_layout()
            plt.savefig(os.path.join(self.res_dir, f"urmCapacityCDF_cs{cs}.png"),
                        dpi=300, bbox_inches='tight')
            plt.close(fig)

    if dollars:
        means = df_loss.mean() / self.area
        lb    = np.maximum(0, df_loss.quantile(0.025)) / self.area
        ub    = df_loss.quantile(0.975) / self.area
        plt.figure(figsize=(10, 6))
        plt.plot(df_loss.columns, means, color='blue', lw=2, label='Mean repair cost')
        plt.fill_between(df_loss.columns, lb, ub, color='blue', alpha=0.2, label='95%
bounds')
        plt.title('Repair costs vs. flood depth', fontsize=14)
        plt.xlabel('Flood depth (m)', fontsize=12)
        plt.ylabel('Repair cost ($/m²)', fontsize=12)
        plt.ylim(0, dollar_ylim) if dollar_ylim else plt.ylim(bottom=0)
        plt.grid(True, ls='--', alpha=0.6)
        plt.legend(loc='upper left')
        if xlim: plt.xlim(*xlim)
        plt.savefig(os.path.join(self.res_dir, f"lossCurve_dollar_cs{cs}.png"),
                    dpi=300, bbox_inches='tight')
        plt.close()

    max_costs = self.calculate_max_costs(cs)
    ratios    = df_loss.div(max_costs, axis=0)

    if ratio:
        means = ratios.mean()
        lb    = np.maximum(0, ratios.quantile(0.025))
        ub    = ratios.quantile(0.975)
        plt.figure(figsize=(10, 6))
        plt.plot(ratios.columns, means, color='red', lw=2, label='Mean damage ratio')
        plt.fill_between(ratios.columns, lb, ub, color='red', alpha=0.2, label='95%
bounds')
        plt.title('Building damage ratio vs. flood depth', fontsize=14)
        plt.xlabel('Flood depth (m)', fontsize=12)
        plt.ylabel('Damage ratio', fontsize=12)
        plt.ylim(0, 1.05)
        plt.grid(True, ls='--', alpha=0.6)
        plt.legend(loc='upper left')
        if xlim: plt.xlim(*xlim)
        plt.savefig(os.path.join(self.res_dir, f"damageRatioCurve_cs{cs}.png"),
                    dpi=300, bbox_inches='tight')
        plt.close()
```

```python
        if euros:
            # Convert damage ratio to € via stochastic Dutch replacement cost
            # (1318–1891 €/m², BDA building cost statistics)
            euro_bounds = (1318, 1891, (1318+1891)/2, (1891-1318)/4)
            price = ratios.mul(self._sample_truncnorm(euro_bounds), axis=0)
            price.to_pickle(os.path.join(self.run_dir, f"euroMatrix_cs{cs}.pkl"))
            means = price.mean(axis=0)
            lb    = np.maximum(0, price.quantile(0.025))
            ub    = price.quantile(0.975)
            plt.figure(figsize=(10, 6))
            plt.plot(price.columns, means, color='red', lw=2, label='Mean repair cost')
            plt.fill_between(price.columns, lb, ub, color='red', alpha=0.2, label='95%
bounds')
            plt.title(f'Repair costs (€) vs. flood depth – CS{cs}', fontsize=14)
            plt.xlabel('Flood depth (m)', fontsize=12)
            plt.ylabel('Repair cost (€)', fontsize=12)
            plt.ylim(0, euro_ylim) if euro_ylim else plt.ylim(bottom=0)
            plt.grid(True, ls='--', alpha=0.6)
            plt.legend(loc='upper left')
            if xlim: plt.xlim(*xlim)
            plt.savefig(os.path.join(self.res_dir, f"lossCurve_Euro_cs{cs}.png"),
                        dpi=300, bbox_inches='tight')
            plt.close()


    def plot_damage_ratio_allCS(self, scenarios, xlim=None):
        """
        Overlay mean damage ratio curves (+ 95 % band) for all CS scenarios
        in a single plot, using consistent colors per CS.
        Saves damageRatioCurve_allCS.png to res_dir.
        """
        CS_COL = {0: 'C0', 1: 'C1', 2: 'C2', 3: 'C3'}
        CS_LBL = {cs: f'CS{cs}' for cs in range(4)}

        plt.figure(figsize=(10, 6))
        for cs in scenarios:
            df_loss   = pd.read_pickle(os.path.join(self.run_dir, f"lossMatrix_cs{cs}.pkl"))
            max_costs = self.calculate_max_costs(cs)
            ratios    = df_loss.div(max_costs, axis=0)
            means = ratios.mean()
            lb    = np.maximum(0, ratios.quantile(0.025))
            ub    = ratios.quantile(0.975)
            col   = CS_COL[cs]
            plt.plot(ratios.columns, means, color=col, lw=2, label=CS_LBL[cs])
            plt.fill_between(ratios.columns, lb, ub, color=col, alpha=0.15)
        plt.title(f'State-Dependent Vulnerability Curves – Including Basement', fontsize=14)
        plt.xlabel('Flood depth (m)', fontsize=12)
        plt.ylabel('Damage ratio', fontsize=12)
        plt.ylim(0, 1.05)
        plt.grid(True, ls='--', alpha=0.6)
        plt.legend(loc='upper left')
        if xlim:
            plt.xlim(*xlim)
        plt.savefig(os.path.join(self.res_dir, "damageRatioCurve_allCS.png"),
                    dpi=300, bbox_inches='tight')
        plt.close()
        print('  Saved: damageRatioCurve_allCS.png')
```

```python
    def plot_downtime_cdfs(self):
        """
        CDF of total downtime (weeks) per DS, read directly from
        downtime_model.pkl — independent of flood depth and scenario.
        """
        colors = {0:'#aec7e8', 1:'#1f77b4', 2:'#2ca02c', 3:'#ff7f0e', 4:'#d62728'}
        labels = {0:'DS0 (Undamaged)', 1:'DS1 (Minor)', 2:'DS2 (Moderate)',
                  3:'DS3 (Extensive)', 4:'DS4 (Complete)'}

        global_max = max(self.downtime_df[ds].max() for ds in range(5)) / 7

        fig, ax = plt.subplots(figsize=(10, 6))
        for p_ref, lbl in [(0.50, 'p50'), (0.95, 'p95')]:
            ax.axhline(p_ref, color='grey', ls=':', lw=0.8, alpha=0.6)
            ax.text(0.01, p_ref + 0.01, lbl, transform=ax.get_yaxis_transform(),
                    fontsize=7, color='grey', va='bottom')
        for ds in range(5):
            t  = np.sort(self.downtime_df[ds].values) / 7  # days → weeks
            cdf = np.arange(1, len(t)+1) / len(t)
            t   = np.concatenate([[0], t, [global_max]])
            cdf = np.concatenate([[0], cdf, [1.0]])
            ax.plot(t, cdf, color=colors[ds], lw=2.5, label=labels[ds])
            _add_percentile_markers(ax, t, cdf, colors[ds], xlim_cap=150)

        ax.set_title('Cumulative distribution of re-occupancy downtime by URM DS',
fontsize=14)
        ax.set_xlabel('Downtime (weeks)', fontsize=12)
        ax.set_ylabel('Probability of non-exceedance', fontsize=12)
        ax.set_xlim(0, 80)
        ax.set_ylim(0, 1.05)
        ax.grid(True, which='both', ls='--', alpha=0.6)
        ax.legend(loc='lower right', fontsize=11)
        fig.tight_layout()
        plt.savefig(os.path.join(self.res_dir, "downtime_CDF_by_DS.png"),
                    dpi=300, bbox_inches='tight')

    def export_capacity_tables(self, scenarios):
        """
        Export p50 and p95 of URM wall capacity (kN/m²) for every CS x DS combination.
        Saves capacity_percentiles_FF.csv and capacity_percentiles_SF.csv to res_dir.
        Cells where DS <= CS are left empty (those states are already exceeded).
        """
        DS_KEYS = ['DS2', 'DS3', 'DS4']
        cols    = [f'{ds}_{p}' for ds in DS_KEYS for p in ('p50', 'p95')]
        tables  = {fl: pd.DataFrame(index=sorted(scenarios), columns=cols, dtype=float)
                   for fl in ('FF', 'SF')}

        for cs in scenarios:
            path = os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl")
            if not os.path.exists(path):
                continue
            df_fail = pd.read_pickle(path)
            is_urm  = df_fail.index.str.contains(r'Brick walls \(DS[2-4]\)', regex=True)

            for k in np.where(is_urm)[0]:
                name = df_fail.index[k]
```

```
                cap  = np.asarray(df_fail.iloc[k, 0], dtype=float)

                if   '(DS2)' in name: ds_key = 'DS2'
                elif '(DS3)' in name: ds_key = 'DS3'
                else:                 ds_key = 'DS4'

                if int(ds_key[2]) <= cs:
                    continue  # physically invalid transition — leave NaN

                floor = 'FF' if 'Floor 0' in name else 'SF'
                tables[floor].loc[cs, f'{ds_key}_p50'] = float(np.percentile(cap, 50))
                tables[floor].loc[cs, f'{ds_key}_p95'] = float(np.percentile(cap, 95))

        for floor, df in tables.items():
            df.index.name = 'CS'
            fpath = os.path.join(self.res_dir, f'capacity_percentiles_{floor}.csv')
            df.to_csv(fpath, float_format='%.3f')
            print(f'  Capacity table saved: {fpath}')

    def plot_capacity_cs_comparison(self, scenarios):
        """
        Produce four capacity CDF plots comparing all CS scenarios for DS2 and DS4:
          urmCapacityCDF_DS{2,4}_allCS.png          — FF + SF, no demand overlay
          urmCapacityCDF_DS{2,4}_allCS_withDemand.png — FF only, with 2 m demand band
        """
        CS_COL = {0: 'C0', 1: 'C1', 2: 'C2', 3: 'C3'}
        CS_LBL = {cs: f'CS{cs}' for cs in range(4)}

        idx_2m    = np.argmin(np.abs(self.depths - 2.0))
        demand_2m = self.demand_matrices['first'][idx_2m, :]
        d_lo, d_med, d_hi = np.percentile(demand_2m, [2.5, 50, 97.5])

        for ds_num in (2, 4):
            ds_tag  = f'DS{ds_num}'
            pattern = rf'Brick walls \({ds_tag}\)'

            # —— FF + SF, no demand ————————————————————————————————————————————
            fig, (ax_ff, ax_sf) = plt.subplots(1, 2, figsize=(16, 6), sharey=True)
            fig.suptitle(f'URM wall panel fragility curve for all Conservation States to
{ds_tag} ',
                         fontsize=14)
            for ax in (ax_ff, ax_sf):
                for p_ref, lbl in [(0.50, 'p50'), (0.95, 'p95')]:
                    ax.axhline(p_ref, color='grey', ls=':', lw=0.8, alpha=0.6)
                    ax.text(0.01, p_ref + 0.01, lbl,
                            transform=ax.get_yaxis_transform(),
                            fontsize=7, color='grey', va='bottom')
            for cs in sorted(scenarios):
                path = os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl")
                if not os.path.exists(path):
                    continue
                df_fail = pd.read_pickle(path)
                color   = CS_COL[cs]
                for k in np.where(df_fail.index.str.contains(pattern, regex=True))[0]:
                    name = df_fail.index[k]
                    cap  = np.sort(np.asarray(df_fail.iloc[k, 0], dtype=float))
                    cdf  = np.arange(1, len(cap) + 1) / len(cap)
```

```
        if 'Floor 0' in name:
            ax_ff.plot(cap, cdf, lw=2, color=color, label=CS_LBL[cs])
            _add_percentile_markers(ax_ff, cap, cdf, color, xlim_cap=50)
        else:
            ax_sf.plot(cap, cdf, lw=2, color=color, label=CS_LBL[cs])
            _add_percentile_markers(ax_sf, cap, cdf, color, xlim_cap=50)
for ax, title in ((ax_ff, 'First floor (FF)'), (ax_sf, 'Second floor (SF)')):
    ax.set_title(title, fontsize=12)
    ax.set_xlabel('Equivalent Uniform Load (kN/m²)', fontsize=11)
    ax.set_xlim(0, 50)
    ax.set_ylim(0, 1.05)
    ax.grid(True, alpha=0.5)
    ax.legend(loc='lower right', fontsize=10)
ax_ff.set_ylabel('P(capacity ≤ demand)', fontsize=11)
fig.tight_layout()
fname = f'urmCapacityCDF_{ds_tag}_allCS.png'
plt.savefig(os.path.join(self.res_dir, fname), dpi=300, bbox_inches='tight')
plt.close(fig)
print(f'  Saved: {fname}')

# ── FF only, with 2 m demand band ─────────────────────────────────
fig, ax = plt.subplots(figsize=(10, 6))
fig.suptitle(
    f'URM wall {ds_tag} capacity CDF – all CS (First floor, 2 m demand)',
    fontsize=14)
for p_ref, lbl in [(0.50, 'p50'), (0.95, 'p95')]:
    ax.axhline(p_ref, color='grey', ls=':', lw=0.8, alpha=0.6)
    ax.text(0.01, p_ref + 0.01, lbl,
            transform=ax.get_yaxis_transform(),
            fontsize=7, color='grey', va='bottom')
for cs in sorted(scenarios):
    path = os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl")
    if not os.path.exists(path):
        continue
    df_fail = pd.read_pickle(path)
    color   = CS_COL[cs]
    for k in np.where(df_fail.index.str.contains(pattern, regex=True))[0]:
        name = df_fail.index[k]
        if 'Floor 0' not in name:
            continue
        cap = np.sort(np.asarray(df_fail.iloc[k, 0], dtype=float))
        cdf = np.arange(1, len(cap) + 1) / len(cap)
        ax.plot(cap, cdf, lw=2, color=color, label=CS_LBL[cs])
        _add_percentile_markers(ax, cap, cdf, color, xlim_cap=50)
ax.axvspan(d_lo, d_hi, alpha=0.15, color='black',
           label=f'Demand at 2 m (95%: {d_lo:.2f}–{d_hi:.2f} kN/m²)')
ax.axvline(d_med, color='black', lw=1.2, ls='-',
           label=f'Demand at 2 m (median: {d_med:.2f} kN/m²)')
ax.set_xlabel('Equivalent Uniform Load (kN/m²)', fontsize=11)
ax.set_ylabel('P(capacity ≤ demand)', fontsize=11)
ax.set_xlim(0, 50)
ax.set_ylim(0, 1.05)
ax.grid(True, alpha=0.5)
ax.legend(loc='lower right', fontsize=10)
fig.tight_layout()
fname = f'urmCapacityCDF_{ds_tag}_allCS_withDemand.png'
plt.savefig(os.path.join(self.res_dir, fname), dpi=300, bbox_inches='tight')
```

```python
            plt.close(fig)
            print(f'  Saved: {fname}')

    def calc_prob(self, cs, d):
        """Print failure probability of each component at depth d metres."""
        df   = pd.read_pickle(os.path.join(self.run_dir, f"damageMatrix_cs{cs}.pkl"))
        idx  = np.argmin(np.abs(self.depths - d))
        prob = df["f_prob"].apply(lambda arr: arr[idx])
        print(pd.DataFrame({f"P(fail) at {self.depths[idx]:.2f}m": prob.values},
                           index=df.index))

# ================================================================================
# HELPERS
# ================================================================================
def _add_percentile_markers(ax, x_vals, cdf_vals, color, xlim_cap, levels=(0.50, 0.95)):
    """Draw vertical drop-lines and intersection dots at p50 and p95 for one CDF curve."""
    for p in levels:
        xp = float(np.interp(p, cdf_vals, x_vals))
        if xp > xlim_cap:
            continue
        ax.plot([xp, xp], [0, p], color=color, ls='--', lw=0.8, alpha=0.4)
        ax.plot(xp, p, 'o', color=color, ms=4, zorder=5)

# ================================================================================
# MAIN — POST-PROCESSING PIPELINE
# ================================================================================
if __name__ == "__main__":
    gui = PostProcessGUI()
    if not gui.settings:
        print("Cancelled.")
        exit()

    print("Loading libraries...")
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import truncnorm
    import pandas as pd
    import os

    cfg     = gui.settings
    RUN_DIR = cfg["run_dir"]

    # ── Load run configuration saved by initialize ────────────────────────
    run_config = pd.read_pickle(os.path.join(RUN_DIR, "run_config.pkl"))
    N_SIM      = run_config["N_SIM"]
    DEPTHS     = np.arange(0, run_config["MAX_DEPTH"] + run_config["INCREMENTS"],
                           run_config["INCREMENTS"])
    FLOORAREA  = run_config["FLOORAREA"]

    # ── Load geometry (only FFE is needed for loss evaluation) ───────────────
    FFE_SAMPLES = pd.read_pickle(os.path.join(RUN_DIR, "master_table.pkl"))["FFE"].values

    # ── Create output folder ──────────────────────────────────────────────────
    RES_DIR = os.path.join(RUN_DIR, "plots")
    os.makedirs(RES_DIR, exist_ok=True)

    # ── Instantiate evaluator (loads demand + downtime pkl automatically) ─────
```

```
    evaluator = FloodRiskEvaluator(N_SIM, FFE_SAMPLES, DEPTHS, FLOORAREA,
                                   RUN_DIR, RES_DIR)

    # ── Compute scenario losses (first pass) ─────────────────────────────────
    if cfg["recompute"]:
        for cs in cfg["scenarios"]:
            print(f"\n--- Computing CS {cs} ---")
            evaluator.compute_scenario_loss(cs)

    # ── Shared y-axis limits across all selected scenarios ────────────────────
    dollar_ylim, euro_ylim = evaluator.compute_global_ylims(cfg["scenarios"])

    # ── Non-structural fragility (CS-independent — plot once) ─────────────────
    if cfg["comp_cdf"] and cfg["scenarios"]:
        print("\n--- Plotting non-structural component fragility ---")
        evaluator.plot_comp_fragility(
            cfg["scenarios"][0],
            min_plot_depth=cfg["min_plot_depth"],
            max_plot_depth=cfg["max_plot_depth"],
        )

    # ── Plot (second pass) ────────────────────────────────────────────────────
    for cs in cfg["scenarios"]:
        print(f"\n--- Plotting CS {cs} ---")
        evaluator.plot_lossCurves(
            cs,
            comp_cdf=cfg["comp_cdf"],
            ratio=cfg["ratio"],
            dollars=cfg["dollars"],
            euros=cfg["euros"],
            min_plot_depth=cfg["min_plot_depth"],
            max_plot_depth=cfg["max_plot_depth"],
            dollar_ylim=dollar_ylim,
            euro_ylim=euro_ylim,
        )

    evaluator.plot_downtime_cdfs()

    if cfg["ratio"] and len(cfg["scenarios"]) > 1:
        print("\nPlotting combined damage ratio curves across CS...")
        evaluator.plot_damage_ratio_allCS(
            cfg["scenarios"],
            xlim=(cfg["min_plot_depth"], cfg["max_plot_depth"]) if cfg["max_plot_depth"] else
None,
        )

    if cfg["comp_cdf"]:
        print("\nPlotting capacity CDF comparisons across CS...")
        evaluator.plot_capacity_cs_comparison(cfg["scenarios"])
        print("\nExporting capacity percentile tables...")
        evaluator.export_capacity_tables(cfg["scenarios"])

    print("\n=== POST-PROCESSING COMPLETE ===")
"""
extract_location_risk.py — Extract damage ratio, downtime, and loss percentiles
at specific flood depths for all CS scenarios.
```

```python
For each CS and each target depth, reports p50 and p95 of:
  – Damage ratio  (dimensionless, 0-1)
  – Repair downtime (weeks)
  – Loss (€/m²) – from euroMatrix_cs{cs}.pkl saved by postprocess.py

Output: <run_dir>/<location>/location_risk.csv
"""

import tkinter as tk
from tkinter import filedialog, ttk
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


CS_SCENARIOS = [0, 1, 2, 3]

LOCATIONS = {
    "kamperfoeliestraat": {"depths": [0.30, 1.32], "subdir": "kamperfoeliestraat"},
    "sGravendijkwal":     {"depths": [0.20, 2.18], "subdir": "sGravendijkwal"},
}


def pick_settings():
    """Show a GUI to select location and results folder. Returns (run_dir, location_cfg) or
(None, None)."""
    result = {"run_dir": None, "location": None}

    root = tk.Tk()
    root.title("Extract Location Risk")
    root.resizable(False, False)

    f = ttk.Frame(root, padding=15)
    f.grid(row=0, column=0, sticky="nsew")

    ttk.Label(f, text="Location:", font=("TkDefaultFont", 10, "bold")).grid(
        row=0, column=0, sticky="w", pady=(0, 4))

    loc_var = tk.StringVar(value=next(iter(LOCATIONS)))
    for i, loc in enumerate(LOCATIONS):
        depths = LOCATIONS[loc]["depths"]
        label  = f"{loc}  (d = {depths[0]:.2f} m, {depths[1]:.2f} m)"
        ttk.Radiobutton(f, text=label, variable=loc_var, value=loc).grid(
            row=i + 1, column=0, sticky="w")

    ttk.Separator(f, orient="horizontal").grid(
        row=len(LOCATIONS) + 1, column=0, sticky="ew", pady=8)

    folder_var = tk.StringVar(value="(no folder selected)")
    ttk.Label(f, text="Results folder:", font=("TkDefaultFont", 10, "bold")).grid(
        row=len(LOCATIONS) + 2, column=0, sticky="w")
    ttk.Label(f, textvariable=folder_var, foreground="grey").grid(
        row=len(LOCATIONS) + 3, column=0, sticky="w", pady=(0, 4))

    def browse():
        path = filedialog.askdirectory(title="Select results folder")
```

```
        if path:
            folder_var.set(path)

    ttk.Button(f, text="Browse…", command=browse).grid(
        row=len(LOCATIONS) + 4, column=0, sticky="w")

    ttk.Separator(f, orient="horizontal").grid(
        row=len(LOCATIONS) + 5, column=0, sticky="ew", pady=8)

    def run():
        if folder_var.get() == "(no folder selected)":
            tk.messagebox.showwarning("Missing folder", "Please select a results folder
first.")
            return
        result["run_dir"]  = folder_var.get()
        result["location"] = loc_var.get()
        root.destroy()

    def cancel():
        root.destroy()

    btn_frame = ttk.Frame(f)
    btn_frame.grid(row=len(LOCATIONS) + 6, column=0, sticky="e")
    ttk.Button(btn_frame, text="Cancel", command=cancel).grid(row=0, column=0, padx=(0, 6))
    ttk.Button(btn_frame, text="Run", command=run).grid(row=0, column=1)

    root.mainloop()

    if result["run_dir"] is None:
        return None, None
    return result["run_dir"], LOCATIONS[result["location"]]


def pick_folder(title):
    root = tk.Tk()
    root.withdraw()
    folder = filedialog.askdirectory(title=title)
    root.destroy()
    return folder


def nearest_depth_col(df, target):
    """Return the column label whose value is closest to target."""
    cols = np.array(df.columns, dtype=float)
    return df.columns[np.argmin(np.abs(cols - target))]


def load_damage_ratio(run_dir, cs):
    df_loss   = pd.read_pickle(os.path.join(run_dir, f"lossMatrix_cs{cs}.pkl"))
    simdata   = pd.read_pickle(os.path.join(run_dir, f"simData_cs{cs}.pkl"))
    cost_mat  = np.stack(simdata.iloc[:, 1].values)
    is_ds1to3 = simdata.index.str.contains(r'Brick walls \(DS[1-3]\)')
    max_costs = np.sum(cost_mat[~is_ds1to3, :], axis=0)
    return df_loss.div(max_costs, axis=0)   # DataFrame: sims × depths


def load_downtime_weeks(run_dir, cs):
```

```python
    df = pd.read_pickle(os.path.join(run_dir, f"downtimeMatrix_cs{cs}.pkl"))
    return df / 7.0   # days → weeks


def load_euro_per_m2(run_dir, cs):
    """Return DataFrame (sims × depths) of repair cost in €/m² from euroMatrix."""
    return pd.read_pickle(os.path.join(run_dir, f"euroMatrix_cs{cs}.pkl"))


def load_ds_probs(run_dir, cs, target_depth):
    """Return {ds: fraction} at the nearest depth step from dsMatrix_cs{cs}.pkl."""
    ds_mat = pd.read_pickle(os.path.join(run_dir, f"dsMatrix_cs{cs}.pkl"))
    col    = nearest_depth_col(ds_mat, target_depth)
    return ds_mat[col].to_dict()


def main():
    run_dir, loc_cfg = pick_settings()
    if not run_dir:
        print("Cancelled.")
        return

    target_depths = loc_cfg["depths"]
    out_dir = os.path.join(run_dir, loc_cfg["subdir"])
    os.makedirs(out_dir, exist_ok=True)

    # raw_data[depth_idx][cs] = {'dr': array, 'euro': array, 'down': array}
    raw_data = {i: {} for i in range(len(target_depths))}
    rows = []
    for cs in CS_SCENARIOS:
        dr   = load_damage_ratio(run_dir, cs)
        down = load_downtime_weeks(run_dir, cs)
        euro = load_euro_per_m2(run_dir, cs)

        row = {"CS": cs}
        for i, depth in enumerate(target_depths):
            dr_col   = nearest_depth_col(dr,   depth)
            down_col = nearest_depth_col(down, depth)
            euro_col = nearest_depth_col(euro, depth)
            actual   = float(dr_col)
            prefix   = f"d{actual:.2f}_"

            dr_vals   = dr[dr_col].values
            down_vals = down[down_col].values
            euro_vals = euro[euro_col].values

            raw_data[i][cs] = {"dr": dr_vals, "euro": euro_vals, "down": down_vals}

            row[prefix + "DR_p50"]          = round(float(np.percentile(dr_vals,   50)), 4)
            row[prefix + "DR_p95"]          = round(float(np.percentile(dr_vals,   95)), 4)
            row[prefix + "downtime_p50_wk"] = round(float(np.percentile(down_vals, 50)), 2)
            row[prefix + "downtime_p95_wk"] = round(float(np.percentile(down_vals, 95)), 2)
            row[prefix + "euro_p50_m2"]     = round(float(np.percentile(euro_vals, 50)), 1)
            row[prefix + "euro_p95_m2"]     = round(float(np.percentile(euro_vals, 95)), 1)

        rows.append(row)
```

```python
out = pd.DataFrame(rows)
out_path = os.path.join(out_dir, "location_risk.csv")
out.to_csv(out_path, index=False)
print(f"Saved: {out_path}")
print(out.to_string(index=False))

# ── Box-plot figures: one PNG per depth ───────────────────────────────────
cs_labels = [f"CS{c}" for c in CS_SCENARIOS]
BOX_COLORS = ['#aec7e8', '#1f77b4', '#2ca02c', '#ff7f0e']

DS_COLORS = {0: '#aec7e8', 1: '#1f77b4', 2: '#2ca02c', 3: '#ff7f0e', 4: '#d62728'}
DS_LABELS = {0: 'DS0', 1: 'DS1', 2: 'DS2', 3: 'DS3', 4: 'DS4'}

# Shared ylims across both depth figures (based on raw 95th percentile)
dt_max   = max(max(np.percentile(raw_data[i][cs]["down"], 95)
                   for cs in CS_SCENARIOS for i in range(len(target_depths))) * 1.15, 1.0)
euro_max = max(max(np.percentile(raw_data[i][cs]["euro"], 95)
                   for cs in CS_SCENARIOS for i in range(len(target_depths))) * 1.15, 1.0)

def _boxplot(ax, arrays, ylabel, ylim):
    bp = ax.boxplot(arrays, tick_labels=cs_labels, patch_artist=True,
                    whis=[5, 95], showfliers=False,
                    medianprops=dict(color='black', lw=1.5))
    for patch, color in zip(bp['boxes'], BOX_COLORS):
        patch.set_facecolor(color)
        patch.set_alpha(0.85)
    ax.set_ylabel(ylabel, fontsize=10)
    ax.set_xlabel("Conservation State", fontsize=10)
    ax.set_ylim(*ylim)
    ax.grid(axis='y', ls='--', alpha=0.5)

for i, depth in enumerate(target_depths):
    ds_probs = {cs: load_ds_probs(run_dir, cs, depth) for cs in CS_SCENARIOS}

    # Layout: 2 rows × 2 cols
    #   [0,0] Damage ratio   [0,1] DS distribution (stacked bar)
    #   [1,0] Loss (€/m²)    [1,1] Downtime (weeks)
    fig, axes = plt.subplots(2, 2, figsize=(12, 10))
    fig.suptitle(f"Location risk by CS — d = {depth:.2f} m", fontsize=13)

    _boxplot(axes[0, 0],
             [raw_data[i][cs]["dr"]   for cs in CS_SCENARIOS],
             "Damage ratio", (0, 1))
    _boxplot(axes[1, 0],
             [raw_data[i][cs]["euro"] for cs in CS_SCENARIOS],
             "Loss (€/m²)", (0, euro_max))
    _boxplot(axes[1, 1],
             [raw_data[i][cs]["down"] for cs in CS_SCENARIOS],
             "Downtime (weeks)", (0, dt_max))

    # DS distribution: stacked bar in [0,1]
    ax_ds   = axes[0, 1]
    x       = np.arange(len(CS_SCENARIOS))
    bottoms = np.zeros(len(CS_SCENARIOS))
    for ds in range(5):
        vals = np.array([ds_probs[cs][ds] for cs in CS_SCENARIOS])
        ax_ds.bar(x, vals, bottom=bottoms, color=DS_COLORS[ds],
```

```
                    alpha=0.9, label=DS_LABELS[ds])
            bottoms += vals
        ax_ds.set_xticks(x)
        ax_ds.set_xticklabels(cs_labels)
        ax_ds.set_ylabel("Fraction of simulations", fontsize=10)
        ax_ds.set_xlabel("Conservation State", fontsize=10)
        ax_ds.set_ylim(0, 1.05)
        ax_ds.legend(fontsize=9, loc='upper right')
        ax_ds.grid(axis='y', ls='--', alpha=0.5)

        fig.tight_layout()
        depth_str  = f"{depth:.2f}".replace(".", "p")
        chart_path = os.path.join(out_dir, f"location_risk_charts_d{depth_str}m.png")
        fig.savefig(chart_path, dpi=300, bbox_inches='tight')
        plt.close(fig)
        print(f"Saved: {chart_path}")


if __name__ == "__main__":
    main()
```

E
Results Data

EA Mitigation Assessment Results

EAA Without Basement

CS1

Table E.1: Mitigation effectiveness — without-basement archetype, CS1.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.1620	5.57	0.0	0.0	−18.6	0.0	−18.6	0.0
1.0	0.2309	5.57	0.0	0.0	−22.5	0.0	−22.5	0.0
1.5	0.2609	5.57	0.0	0.0	−21.7	0.0	−21.7	0.0
2.0	0.3387	5.68	0.0	−0.3	−23.8	0.2	−23.8	0.0
2.5	0.3575	7.45	0.0	0.9	−17.2	−1.1	−17.2	0.9
3.0	0.3975	10.92	−0.1	0.5	−14.7	−0.4	−14.8	−0.5
3.5	0.5291	11.02	0.0	0.4	−9.2	−0.4	−9.3	−0.6
4.0	0.6043	13.09	−0.1	−1.5	−7.4	−0.5	−7.5	−1.6
4.5	0.6587	20.60	−0.1	−0.5	−6.2	−0.4	−6.4	−0.5
5.0	0.6953	24.04	−0.1	0.2	−5.6	−0.1	−5.7	0.1
5.5	0.7193	24.67	0.0	0.0	−5.2	0.0	−5.3	0.1
6.0	0.7681	24.73	0.0	0.0	−4.5	0.0	−4.5	0.0
6.5	0.8077	24.73	0.0	0.0	−4.0	0.0	−4.0	0.0
7.0	0.8455	24.73	0.0	0.0	−3.4	0.0	−3.5	0.0

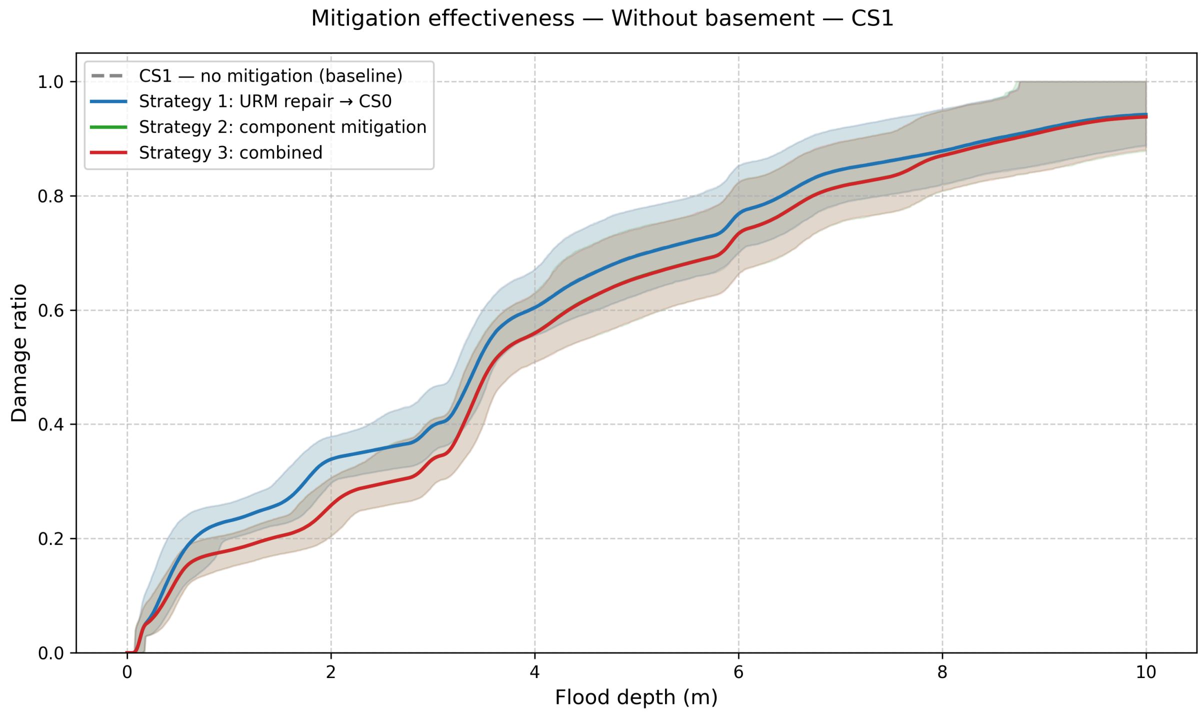


Figure E.1: Vulnerability curves — without-basement archetype, CS1.

CS2

Table E.2: Mitigation effectiveness — without-basement archetype, CS2.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.1680	5.57	−3.6	0.0	−17.7	0.0	−21.5	0.0
1.0	0.2375	5.57	−2.8	0.0	−21.7	0.0	−24.6	0.0
1.5	0.2675	5.62	−2.5	−0.9	−21.0	−0.2	−23.7	−0.9
2.0	0.3473	7.03	−2.5	−19.5	−23.1	−0.1	−25.7	−19.3
2.5	0.3717	12.43	−3.9	−39.5	−16.3	−0.2	−20.4	−39.5
3.0	0.4141	17.17	−4.1	−36.1	−13.7	0.3	−18.2	−36.7
3.5	0.5490	17.38	−3.7	−36.3	−8.6	0.0	−12.6	−37.0
4.0	0.6315	20.47	−4.4	−37.0	−6.7	0.0	−11.5	−37.1
4.5	0.6829	23.94	−3.7	−14.4	−5.7	0.1	−9.7	−14.4
5.0	0.7132	24.78	−2.6	−2.8	−5.2	0.1	−8.1	−2.9
5.5	0.7349	24.89	−2.2	−0.9	−4.9	0.0	−7.3	−0.8
6.0	0.7834	24.91	−2.0	−0.7	−4.2	0.0	−6.4	−0.7
6.5	0.8229	24.91	−1.9	−0.7	−3.7	0.0	−5.8	−0.7
7.0	0.8608	24.91	−1.8	−0.7	−3.2	0.0	−5.2	−0.7

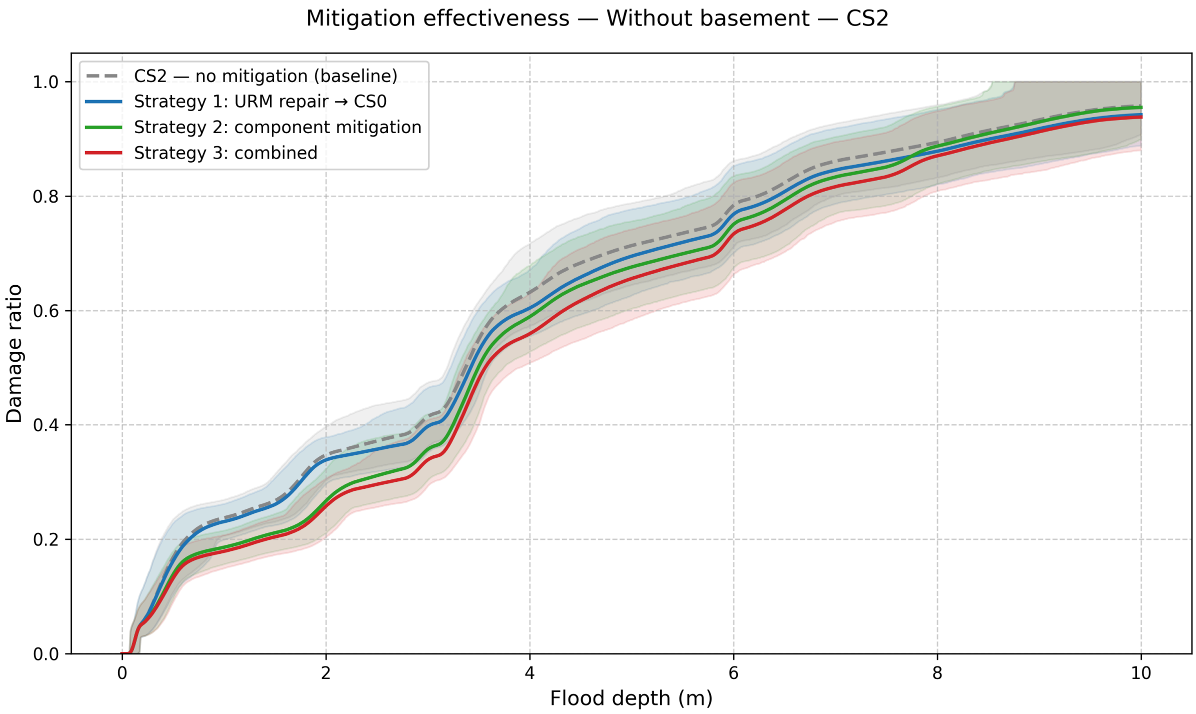


Figure E.2: Vulnerability curves — without-basement archetype, CS2.

CS3

Table E.3: Mitigation effectiveness — without-basement archetype, CS3.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.1767	24.81	−8.4	−77.5	−16.5	0.0	−25.4	−77.5
1.0	0.2470	24.81	−6.5	−77.5	−20.6	0.0	−27.5	−77.5
1.5	0.2782	24.81	−6.2	−77.5	−20.0	0.0	−26.6	−77.5
2.0	0.3625	24.84	−6.6	−77.2	−21.8	0.0	−28.9	−77.1
2.5	0.3885	24.87	−8.0	−69.8	−15.3	0.0	−23.8	−69.8
3.0	0.4277	24.89	−7.1	−55.9	−13.2	0.0	−20.8	−56.4
3.5	0.5670	24.90	−6.7	−55.6	−8.2	0.0	−15.4	−56.0
4.0	0.6565	24.93	−8.0	−48.3	−6.3	0.0	−14.8	−48.3
4.5	0.7023	24.94	−6.3	−17.8	−5.4	0.0	−12.2	−17.9
5.0	0.7289	24.95	−4.7	−3.5	−5.1	0.0	−10.1	−3.6
5.5	0.7497	24.95	−4.1	−1.1	−4.7	0.0	−9.1	−1.0
6.0	0.7981	24.95	−3.8	−0.9	−4.1	0.0	−8.1	−0.9
6.5	0.8377	24.95	−3.6	−0.9	−3.6	0.0	−7.4	−0.9
7.0	0.8755	24.95	−3.5	−0.9	−3.1	0.0	−6.8	−0.9

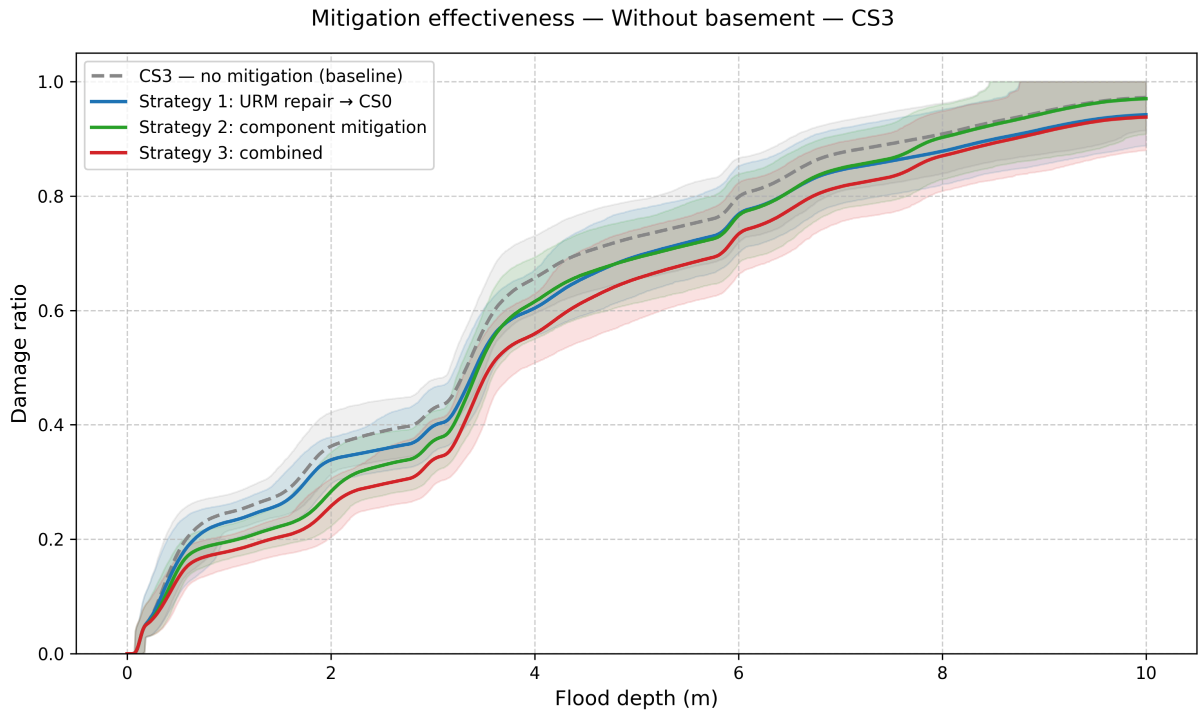


Figure E.3: Vulnerability curves — without-basement archetype, CS3.

EAB With Basement

CS1

Table E.4: Mitigation effectiveness — with-basement archetype, CS1.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.0726	5.57	0.0	0.0	−18.9	0.0	−18.9	0.0
1.0	0.1691	5.57	0.0	0.0	−16.7	0.0	−16.8	0.0
1.5	0.2776	5.57	0.0	0.0	−19.5	0.0	−19.5	0.0
2.0	0.3061	5.57	0.0	0.0	−19.8	0.0	−19.8	0.0
2.5	0.3639	5.78	0.0	−0.2	−23.5	−0.1	−23.5	−0.1
3.0	0.4008	7.83	0.0	1.5	−16.8	0.5	−16.8	2.1
3.5	0.4233	11.69	−0.1	−0.2	−15.1	0.3	−15.2	0.1
4.0	0.5067	12.37	−0.1	−0.9	−11.3	0.0	−11.4	−0.5
4.5	0.6234	13.20	−0.1	−1.8	−7.7	−0.3	−7.8	−0.9
5.0	0.6760	18.97	−0.1	−0.3	−6.5	−0.2	−6.6	−0.6
5.5	0.7162	23.45	−0.1	0.4	−5.6	0.1	−5.7	0.4
6.0	0.7415	24.55	−0.1	0.2	−5.2	0.0	−5.2	0.2
6.5	0.7676	24.74	−0.1	0.0	−4.7	0.0	−4.7	0.0
7.0	0.8141	24.75	−0.1	0.0	−4.0	0.0	−4.0	0.0

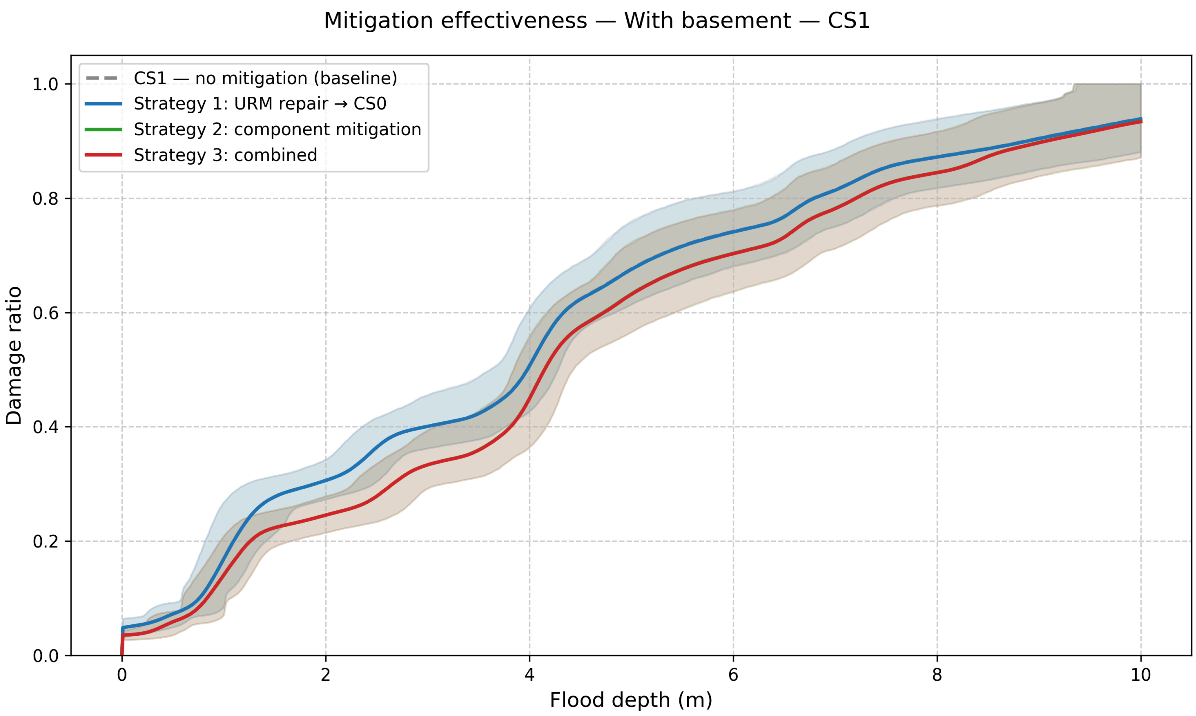


Figure E.4: Vulnerability curves — with-basement archetype, CS1.

CS2

Table E.5: Mitigation effectiveness — with-basement archetype, CS2.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.0726	5.57	0.0	0.0	−18.9	0.0	−18.9	0.0
1.0	0.1712	5.57	−1.3	0.0	−16.4	0.0	−17.8	0.0
1.5	0.2835	5.57	−2.1	0.0	−18.9	0.0	−21.2	0.0
2.0	0.3121	5.63	−1.9	−1.0	−19.3	0.6	−21.3	−1.0
2.5	0.3723	7.55	−2.2	−23.5	−22.8	1.1	−25.2	−23.4
3.0	0.4143	13.40	−3.3	−40.7	−16.0	−0.9	−19.5	−40.3
3.5	0.4383	18.01	−3.5	−35.3	−14.3	0.1	−18.1	−35.1
4.0	0.5238	18.63	−3.4	−34.2	−10.7	−0.2	−14.3	−34.0
4.5	0.6457	20.13	−3.5	−35.6	−7.2	−0.2	−11.0	−35.0
5.0	0.7004	23.51	−3.6	−19.5	−6.0	−0.2	−9.8	−19.8
5.5	0.7347	24.71	−2.6	−4.7	−5.3	0.0	−8.1	−4.7
6.0	0.7571	24.88	−2.1	−1.1	−4.9	0.0	−7.2	−1.1
6.5	0.7824	24.92	−2.0	−0.7	−4.4	0.0	−6.5	−0.7
7.0	0.8289	24.92	−1.8	−0.7	−3.8	0.0	−5.7	−0.7

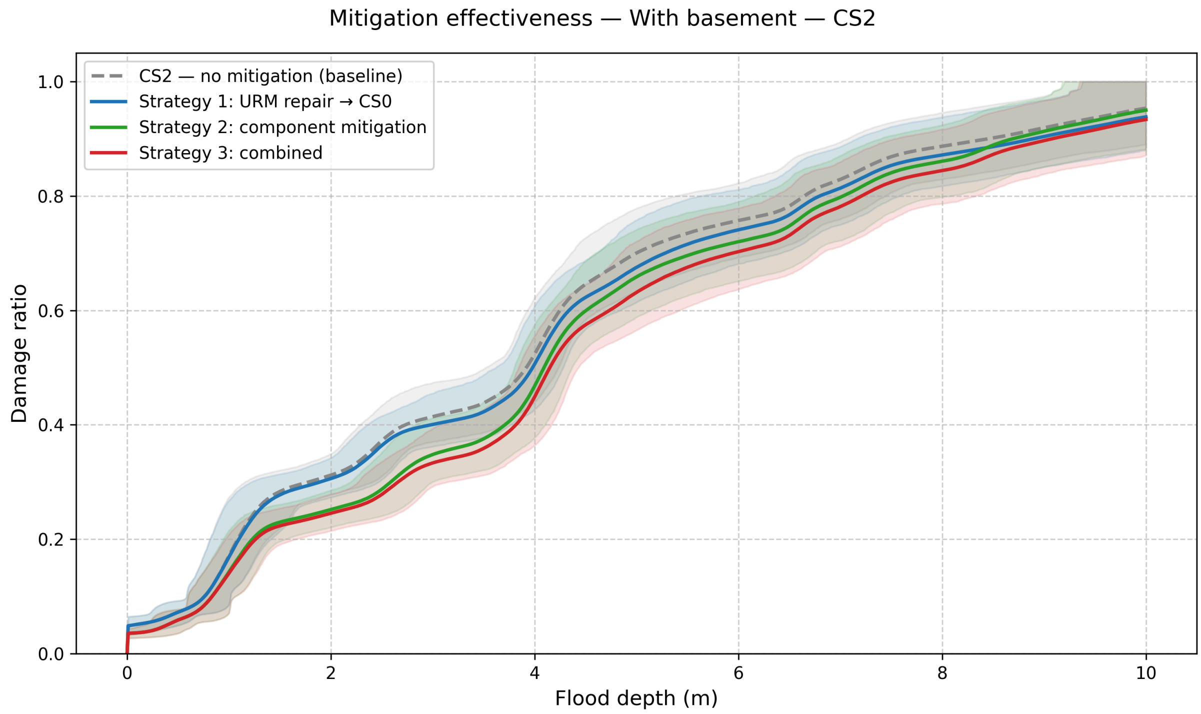


Figure E.5: Vulnerability curves — with-basement archetype, CS2.

CS3

Table E.6: Mitigation effectiveness — with-basement archetype, CS3.

d (m)	Baseline		Strategy 1		Strategy 2		Strategy 3	
	DR_0	DT_0 (wk)	ΔDR_1 (%)	ΔDT_1 (%)	ΔDR_2 (%)	ΔDT_2 (%)	ΔDR_3 (%)	ΔDT_3 (%)
0.5	0.0726	5.57	0.0	0.0	−18.9	0.0	−18.9	0.0
1.0	0.1743	24.81	−3.0	−77.5	−16.0	0.0	−19.3	−77.5
1.5	0.2922	24.81	−5.0	−77.5	−18.1	0.0	−23.5	−77.5
2.0	0.3221	24.82	−5.0	−77.5	−18.5	0.0	−23.8	−77.5
2.5	0.3866	24.84	−5.9	−76.8	−21.6	0.0	−28.0	−76.7
3.0	0.4290	24.88	−6.6	−68.0	−15.1	0.0	−22.3	−67.9
3.5	0.4497	24.90	−6.0	−53.2	−13.7	0.0	−20.2	−53.0
4.0	0.5369	24.90	−5.7	−50.8	−10.2	0.0	−16.4	−50.6
4.5	0.6666	24.92	−6.6	−48.0	−6.7	0.0	−13.8	−47.5
5.0	0.7188	24.94	−6.1	−24.2	−5.6	0.0	−12.2	−24.4
5.5	0.7490	24.95	−4.5	−5.7	−5.0	0.0	−9.8	−5.6
6.0	0.7698	24.95	−3.8	−1.4	−4.7	0.0	−8.7	−1.4
6.5	0.7949	24.95	−3.5	−0.8	−4.2	0.0	−8.0	−0.8
7.0	0.8414	24.95	−3.3	−0.8	−3.6	0.0	−7.1	−0.8

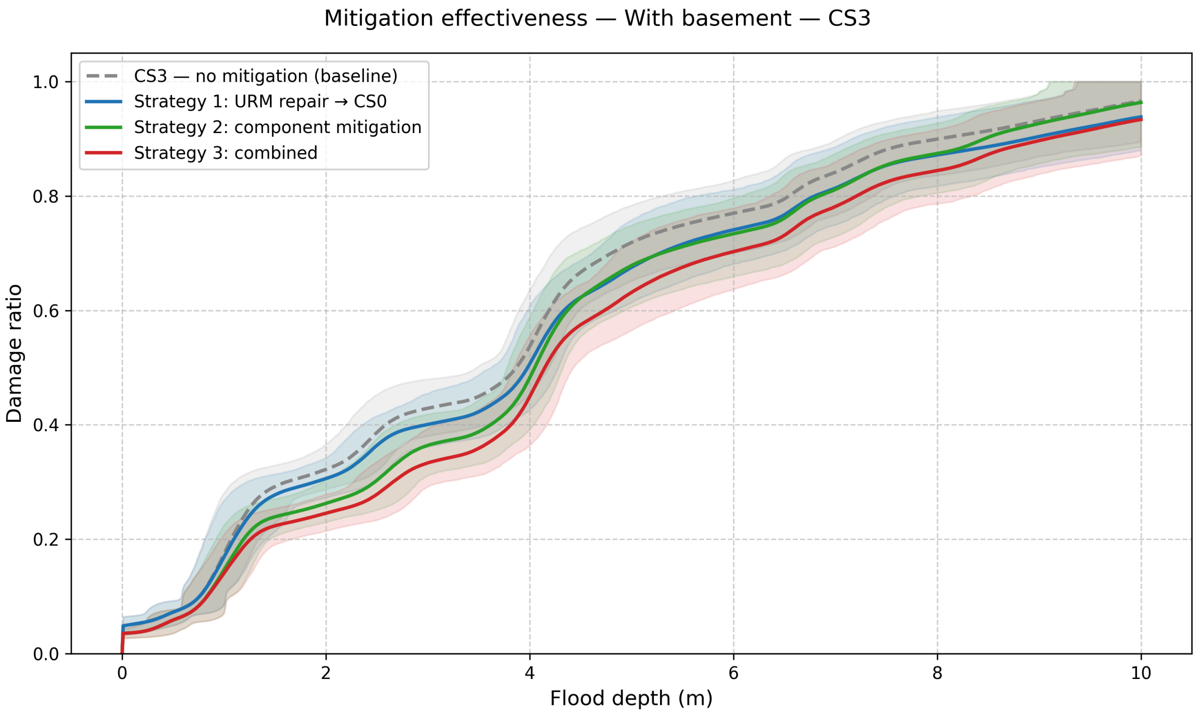


Figure E.6: Vulnerability curves — with-basement archetype, CS3.

EB Location Based Risk Results

EBA Kamperfoeliestraat 71

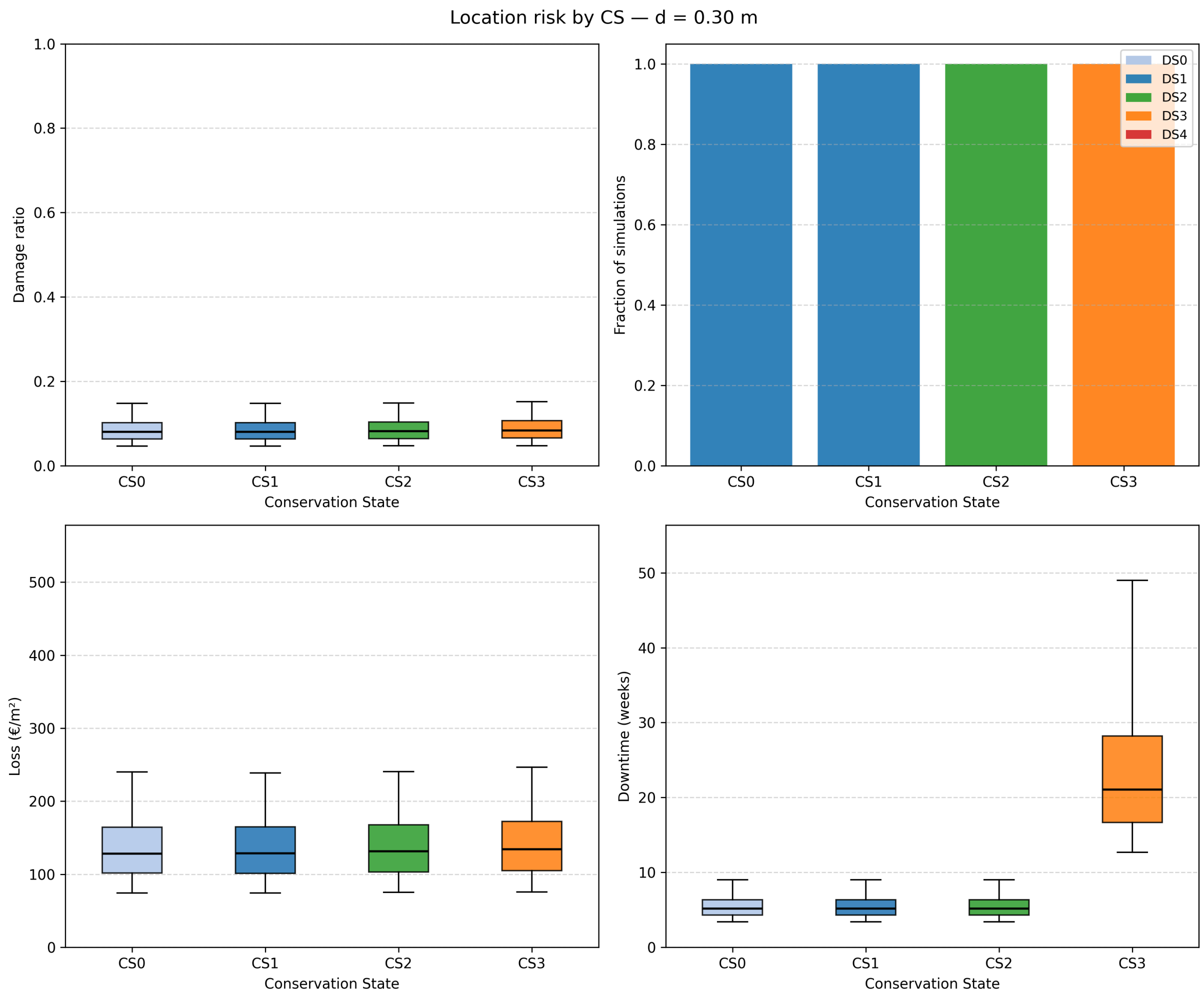


Figure E.7: Overview of risk metrics at the Kamperfoeliestraat (location 1) for pluvial flooding.

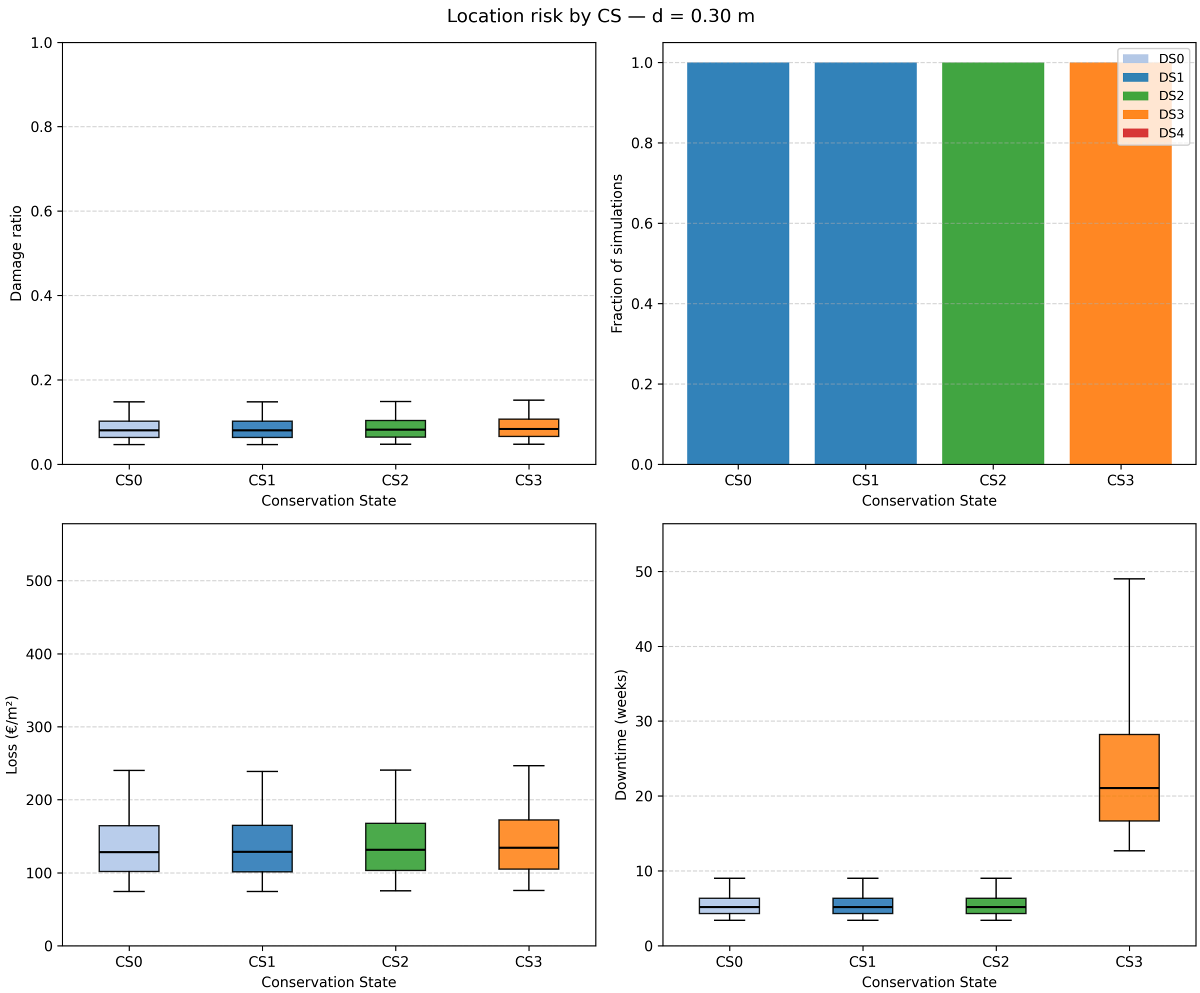


Figure E.8: Overview of risk metrics at the Kamperfoeliestraat (location 1) for fluvial/coastal flooding.

Table E.7: State-dependent absolute losses for two flooding scenarios at location 1 (unmitigated).

CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)
0	0.080 (0.15)	5.2 (9.0)	128 (240)	0.25 (0.28)	5.2 (9.0)	400 (473)
1	0.080 (0.15)	5.2 (9.0)	129 (239)	0.25 (0.28)	5.2 (9.0)	399 (471)
2	0.082 (0.15)	5.2 (9.0)	131 (241)	0.26 (0.28)	5.2 (9.0)	409 (482)
3	0.084 (0.15)	21 (49)	134 (247)	0.27 (0.29)	21 (49)	425 (503)

Table E.8: Relative changes in losses for two flooding scenarios at location 1 after applying mitigation strategies.

Original CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)	Damage Ratio	Downtime (weeks)	Repair Costs (€/m2)
1	−13.8 (-29.0)	0.0 (0.0)	−13.8 (-25.5)	−21.6 (-20.1)	0.0 (0.0)	−21.6 (-20.6)
2	−13.7 (-25.3)	0.0 (0.0)	−13.6 (-24.3)	−23.6 (-22.2)	0.0 (0.0)	−23.8 (-22.4)
3	−17.3 (-27.9)	−75.3 (-81.6)	−18.9 (-28.0)	−26.4 (-24.9)	−75.3 (-81.6)	−26.5 (-25.5)

EBB 's Gravendijkwal 56

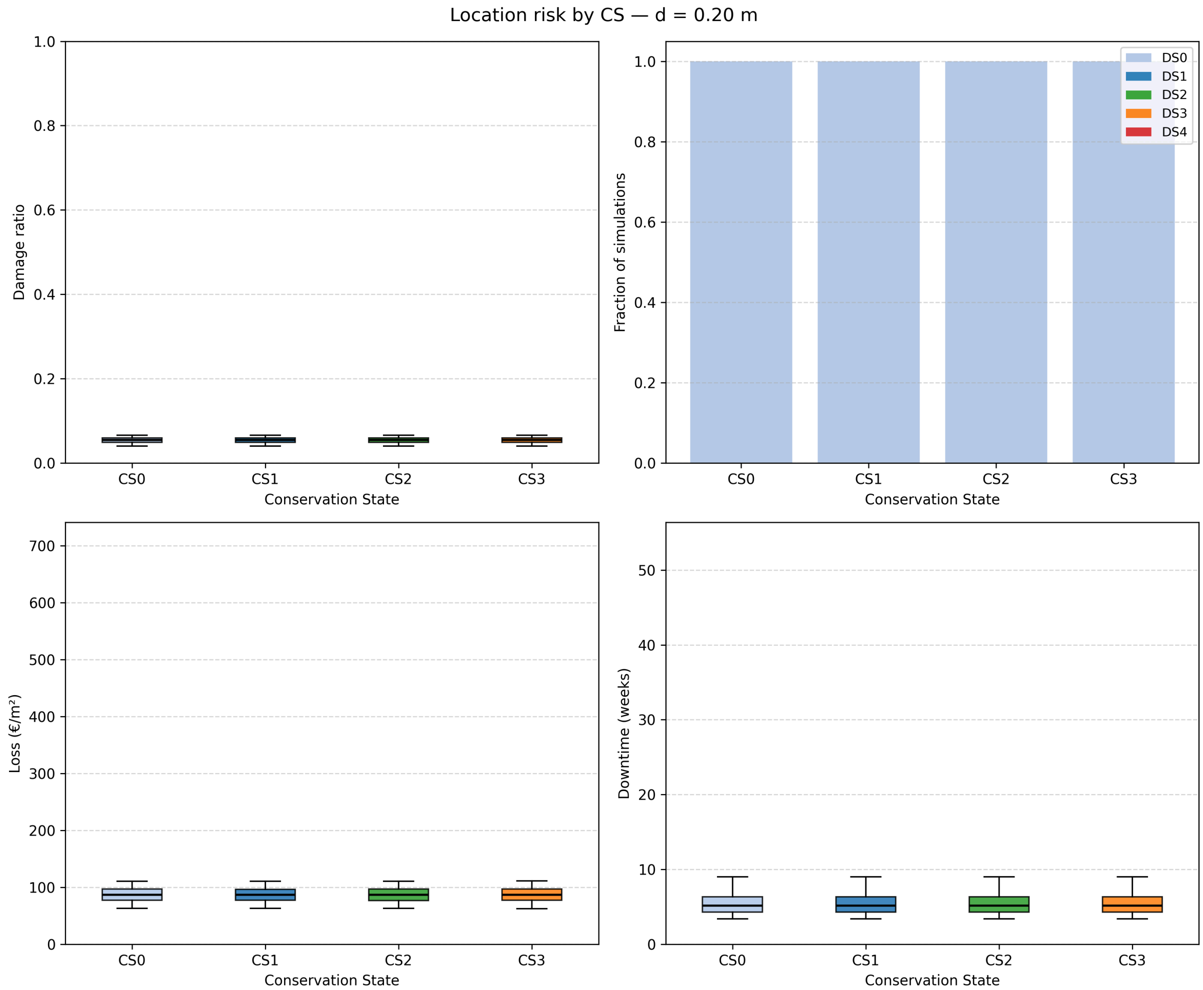


Figure E.9: Overview of risk metrics at the 's Gravendijkwal (location 2) for pluvial flooding.

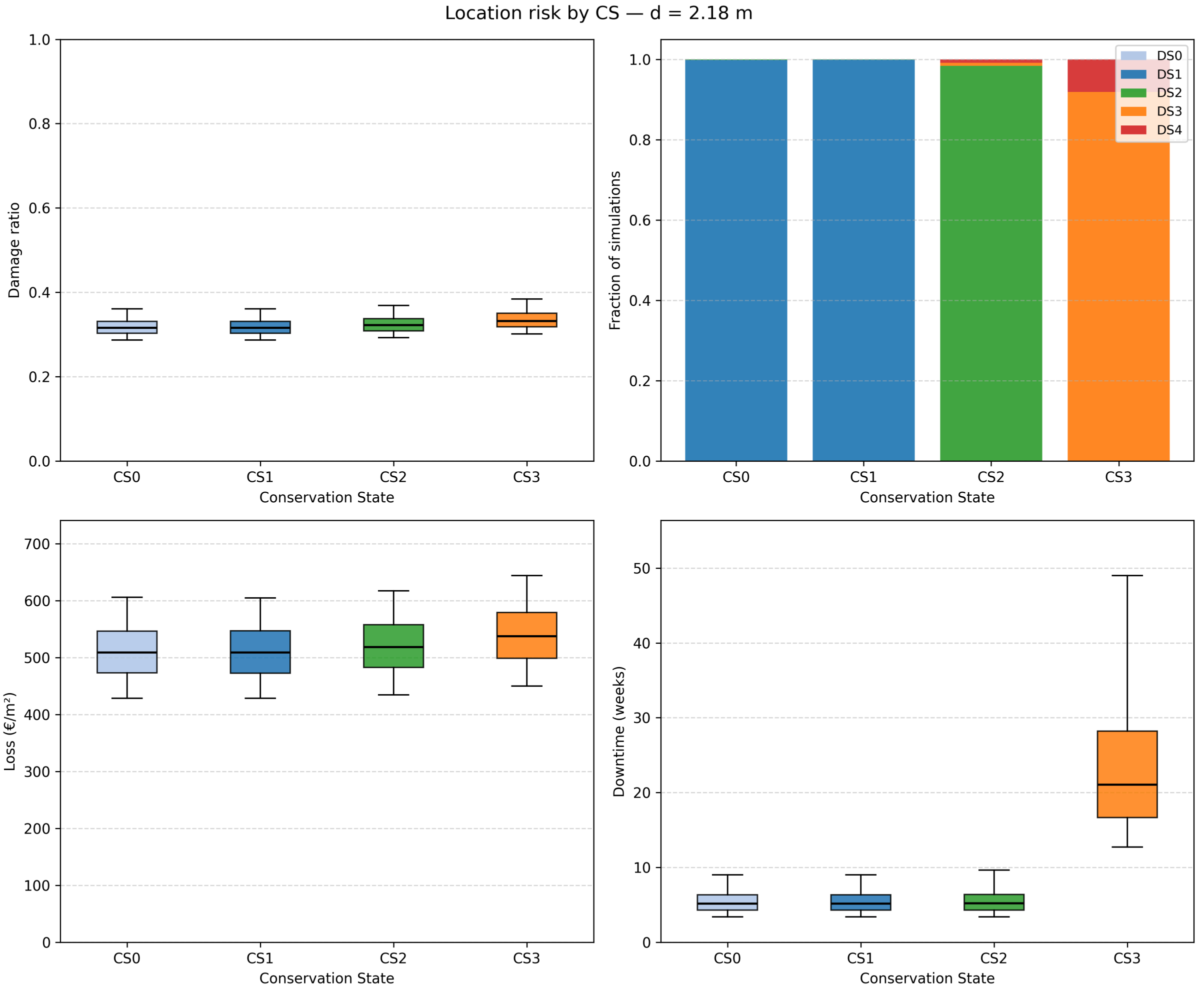


Figure E.10: Overview of risk metrics at the ‘s Gravendijkwal (location 2) for fluvial/coastal flooding.

Table E.9: State-dependent absolute losses for two flooding scenarios at location 2 (unmitigated).

CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	**Downtime (weeks)**	**Repair Costs (€/m2)**	**Damage Ratio**	**Downtime (weeks)**	**Repair Costs (€/m2)**
0	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.36)	5.2 (9.0)	509 (606)
1	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.36)	5.2 (9.0)	509 (605)
2	0.055 (0.067)	5.2 (9.0)	87 (111)	0.32 (0.37)	5.2 (9.7)	519 (617)
3	0.055 (0.067)	5.2 (9.0)	87 (111)	0.33 (0.38)	21 (49)	538 (644)

Table E.10: Relative changes in losses for two flooding scenarios at location 2 after applying mitigation strategies.

CS	Pluvial Flooding (0.30m)			Fluvial/Coastal Flooding (1.32m)		
	Damage Ratio	**Downtime (weeks)**	**Repair Costs (€/m2)**	**Damage Ratio**	**Downtime (weeks)**	**Repair Costs (€/m2)**
1	−31.0 (-25.1)	0.0 (0.0)	−30.1 (-25.4)	−19.9 (0.36)	0.0 (0.0)	−20.3 (-20.8)
2	−30.9 (-25.1)	0.0 (0.0)	−29.9 (-25.7)	−21.4 (0.37)	−0.4 (-6.7)	−21.7 (-22.5)
3	−30.9 (-25.1)	0.0 (0.0)	−30.3 (-26.0)	−23.9 (0.38)	−75.3 (-81.6)	−24.4 (-25.8)