ČERVENKA CONSULTING

Červenka Consulting s.r.o.
Na Hřebenkách 55
150 00 Prague
Czech Republic
Phone: +420 220 610 018
E-mail: cervenka@cervenka.cz
Web: https://www.cervenka.cz

ATENA Module for Green Concrete Modelling and Design

Project Result TM04000013-V2

Software Documentation

Written by

Jan Červenka

Michal Ženíšek

Tomáš Altman

Version 4.0

Prague, December 2025

Acknowledgements:

The software was developed with partial support of ***TAČR DELTA 2 Programme***

T A

Č R

("DELTA 2 Funding programme for applied research, experimental development and innovation", project TM04000013 "Virtual modelling of green concrete - structures with novel multi-spiral reinforcement and steel members")

Contents

Executive Summary

This document presents the **ATENA Module for Green Concrete Modelling and Design**, developed as a **software result (TM04000013-V2)** within the bilateral research project **CeSTaR-3** funded by the Technology Agency of the Czech Republic under the **DELTA 2 Programme**. The software module represents an extension of the ATENA nonlinear finite element analysis system and is dedicated to the advanced modelling, simulation, and design verification of concrete structures made of **green concrete with partial replacement of Portland cement by supplementary cementitious materials (SCMs)**.

The primary purpose of the software is to provide **structural engineers and researchers** with a robust and practically applicable computational tool for the nonlinear analysis of concrete structures incorporating modern low-carbon concrete technologies. The module enables the simulation of **time-dependent material properties**, nonlinear fracture-plastic behavior, and construction-sequence effects that are characteristic of green concrete mixes, including delayed strength development and altered fracture parameters due to SCM content. These capabilities are essential for the reliable design and assessment of sustainable concrete structures, where conventional material models and simplified design assumptions are often insufficient.

The functionality of the software is documented through a comprehensive **user manual**, **technical and theoretical background**, and **programming documentation**, covering both graphical and script-based workflows. The correctness and applicability of the implemented models are demonstrated through **validation examples and benchmark studies**, including comparisons with experimental data and established ATENA validation cases.

The ATENA Module for Green Concrete Modelling and Design is available as a **software plugin within the ATENA system** and requires a valid ATENA license for use. The result directly supports the objectives of the CeSTaR-3 project by enabling the practical adoption of low-carbon concrete technologies in structural design and assessment, thereby contributing to more sustainable construction practices.

Originality and Novelty of the Software Result

The originality of the software result **TM04000013-V2** lies in the development and implementation of **time dependent construction process modelling, specialized material models and modelling workflows for green concrete with supplementary cementitious materials (SCMs)** that go beyond the standard capabilities of nonlinear structural analysis tools. While the ATENA system provides a general framework for nonlinear finite element analysis of concrete structures, the presented module introduces **new functionalities specifically tailored to sustainable concrete technologies**, reflecting the outcomes of experimental and theoretical research conducted within the CeSTaR-3 project.

In particular, the software enables the **explicit modelling of SCM replacement effects**, including modified strength development, fracture properties, and time-dependent material behaviour, which are not available in conventional concrete material models. The integration of these features into a unified nonlinear analysis environment allows engineers to perform **virtual testing and parametric studies** of green concrete structures under realistic loading and construction scenarios.

The novelty of the result further consists in the **direct transfer of research findings into an engineering-grade software implementation**, validated against experimental data and benchmark studies. This approach bridges the gap between laboratory research and practical structural design, enabling the application of low-carbon concrete materials in engineering practice without reliance on oversimplified assumptions or empirical safety margins. As such, the software represents an original and practically applicable outcome of the CeSTaR-3 project, supporting sustainable construction through advanced numerical modelling.

1 Introduction

The **CeSTaR-3 bilateral research project**, supported by the Technology Agency of the Czech Republic within the DELTA 2 Programme, addresses current challenges in structural engineering related to **sustainability, material efficiency, and reliable structural performance**. A central objective of the project is the development of advanced tools that enable the safe and efficient application of **low-carbon concrete technologies** in engineering practice, while maintaining compliance with structural performance and safety requirements.

One of the key research directions of the project is the use of **green concrete**, in which a significant portion of Portland cement is replaced by **supplementary cementitious materials (SCMs)**. Such materials offer substantial reductions in CO_2 emissions and environmental impact; however, they also introduce changes in material behavior, including modified strength development, fracture properties, and time-dependent effects. These characteristics cannot be adequately captured by conventional material models commonly used in structural design and analysis.

Within the CeSTaR-3 project, these material characteristics have been **systematically investigated through an experimental programme**, the results of which are compiled in the **experimental database of green concrete materials (project result TM04000013-V6)**. This database provides validated material parameters and reference test data for concrete mixtures with different types and levels of SCM replacement and forms a key experimental basis for subsequent numerical modelling activities.

To enable the practical use of this experimental knowledge in structural analysis, the CeSTaR-3 project includes the development of **specialized numerical modelling tools** capable of representing the nonlinear behaviour of green concrete within a robust finite element framework. The software result **TM04000013-V2**, documented in this report, represents a direct outcome of this effort. It extends the ATENA nonlinear finite element system by incorporating **experimentally informed material models and modelling workflows**, which can utilize input parameters derived from the experimental database V6.

The software module enables the simulation of **nonlinear structural response**, including cracking, crushing, confinement effects, and time-dependent material evolution, while remaining fully compatible with the existing ATENA solver architecture. This allows the developed models to be applied to practical engineering tasks such as **design verification, parametric studies, and virtual testing** of structural elements and systems made of green concrete.

This document provides comprehensive documentation of the developed software, including a **user-oriented description of the implemented functionality**, a **technical and theoretical background of the underlying models**, and **programming and scripting guidelines** for advanced usage. Together with the experimental database result **TM04000013-V6**, the presented software result supports the transfer of CeSTaR-3 research outcomes into engineering practice and contributes to the broader adoption of sustainable concrete technologies through reliable numerical simulation.

2 User's Manual

This section provides a **practical user-oriented description** of the ATENA software module developed within the CeSTaR-3 project for the modelling and simulation of **green concrete structures with supplementary cementitious materials (SCMs)**. The User Manual is intended primarily for **structural engineers and researchers** who are familiar with nonlinear finite element analysis and the ATENA software environment.

The presented module is implemented as a **plugin to the ATENA system** and is fully integrated into the ATENA preprocessor and solver workflow. The user interface enables intuitive definition of **material properties, structural geometry, reinforcement layout, and loading scenarios**, with a particular focus on parameters relevant to green concrete mixtures and time-dependent material behavior. The module supports both **graphical input via dedicated dialog windows** and **advanced parametric input using scripting**, allowing flexible application ranging from single-case analyses to systematic parametric studies.

The typical workflow when using the module consists of:

- installation and activation of the CeSTaR-3 plugin within the ATENA environment,
- definition of green concrete material parameters, optionally utilizing data derived from the experimental database **TM04000013-V6**,
- creation of the structural model, including geometry, reinforcement, and boundary conditions,
- specification of loading history and analysis type,
- execution of nonlinear analysis and evaluation of results using standard ATENA post-processing tools.

The following subsections describe the individual dialogs, input options, and modelling steps in detail and are accompanied by illustrative figures demonstrating the typical use of the software module.

2.1 Installation and Activation of CeSTaR-3 Software Modules

Software modules developed during CeSTaR-3 project are part of the new installation package for ATENA 2026, which is currently available as a public beta version. It is necessary to possess a valid ATENA maintenance license to be able to use the functionality of the software. Trial license is available upon request from ATENA 2026 Center after the project installation. The full ATENA package including the CeSTaR-3 software modules can be downloaded from the project website:

https://www.cervenka.cz/projects/rd-projects/cestar-3

CeSTaR-3 modules can be activated after launching ATENA Engineering Preprocessor from the Plugin menu:

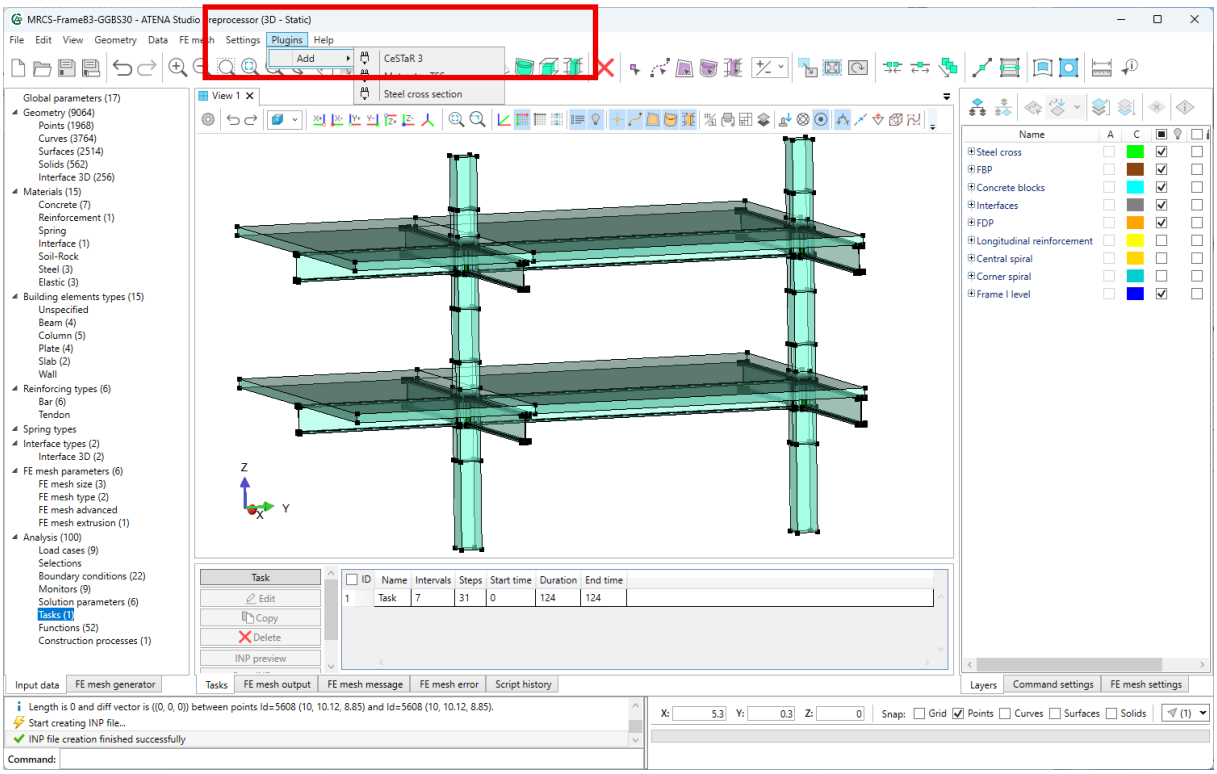


Fig. 1: CeSTaR-3 plugin activation

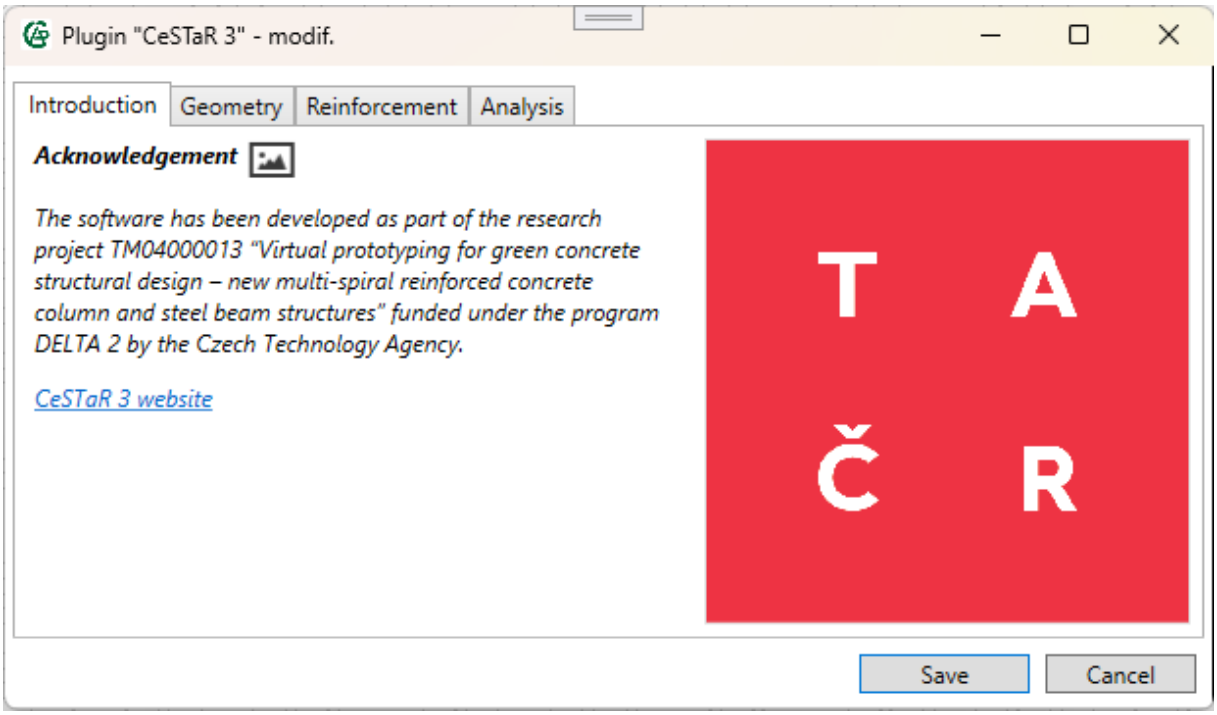


Fig. 1: Plugin activation with CeSTaR-3 acknowledgement

In the case, of ATENA – CeSTaR-3 Green Concrete module the following new commands are available:

- Modelling and simulation of material maturing and time development of material parameters.

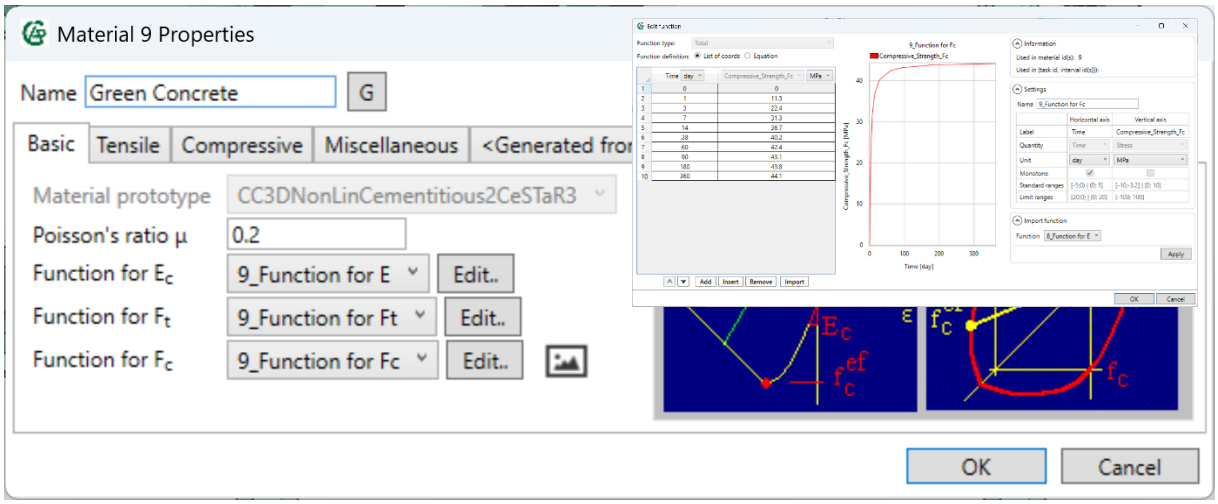


Fig. 2: Material definition dialog for the new time dependent CeSTaR3 material model.

- Modelling of construction process, i.e. gradual activation of various parts of the numerical model and their corresponding loads.

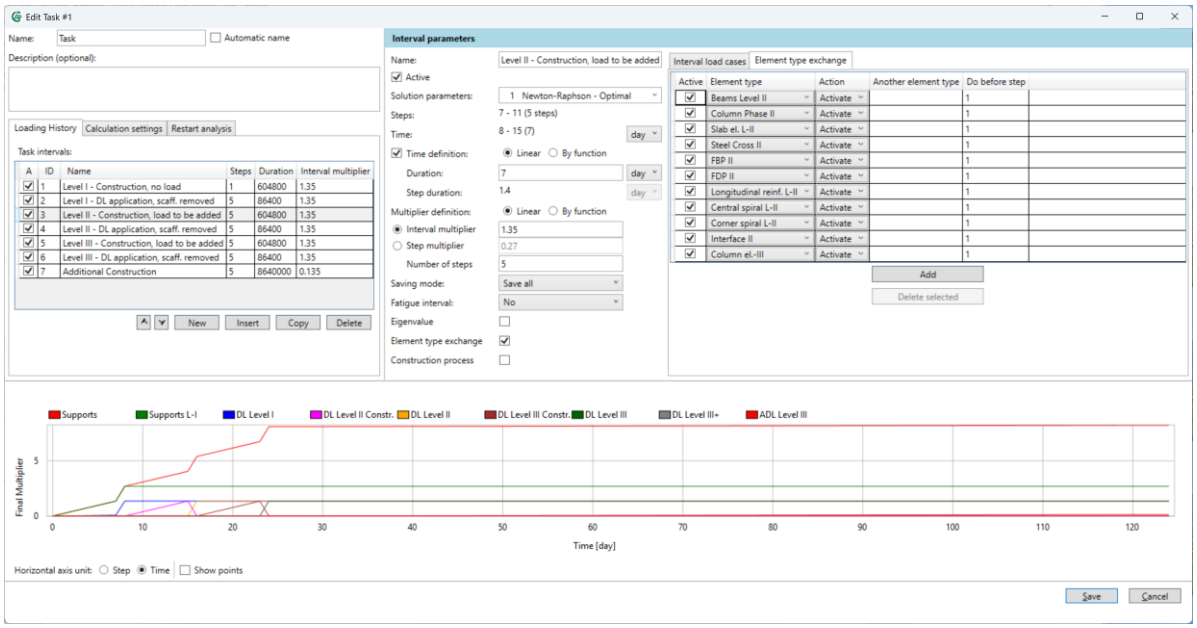


Fig. 3: Loading history and activation of individual structural elements in the ATENA – CeSTaR3 Green Concrete module.

The following two section describes the user dialogs and the parameters of these two key elements of the ATENA Green Concrete module.

2.2 Time Dependent Material for Green Concrete

This part of the ATENA Green Concrete is activated by the CeSTaR-3 plugging by integrating appropriate xml and python script plugins into ATENA material library. This is described in more detail in the Programming Manual Section 5. From the user point of view these new plugin material libraries become accessible from the Materials / Concrete menu as shown in Fig. 4. The specialized CeSTaR-3 materials are developed as an extension of the fracture-plastic Cementitious2 material as described in the Theoretical manual Section 4.

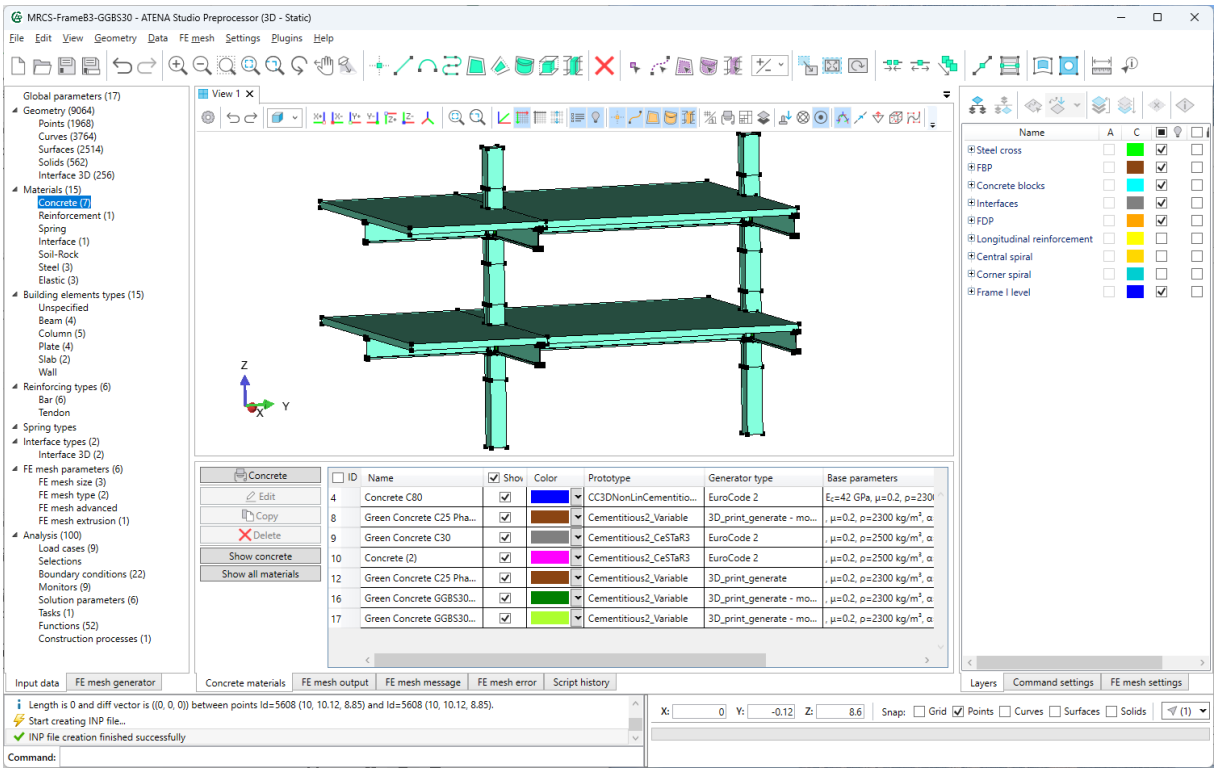


Fig. 4: CeSTaR-3 material plugin library is accessible as a special Concrete type material.

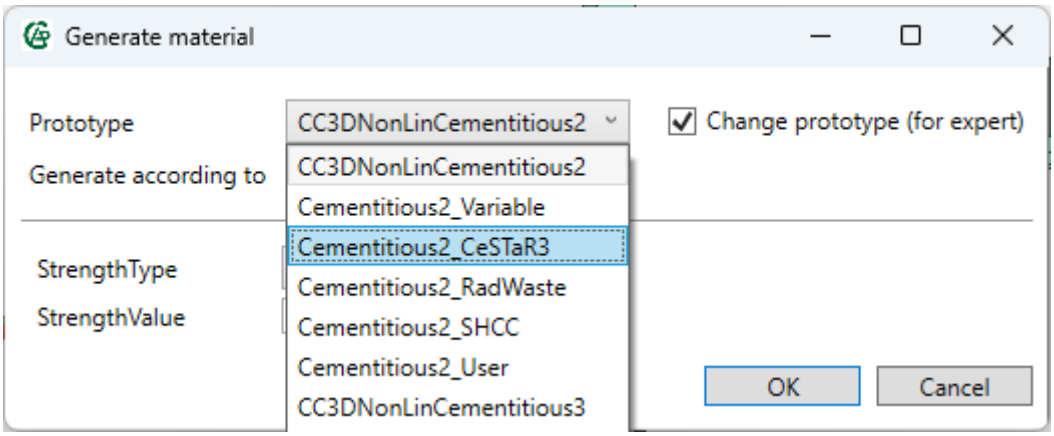


Fig. 5: The specialized CeSTaR-3 materials are developed as an extension of the fracture-plastic Cementitious2 material.

The main new functionality of this material is that it can provide time dependent laws for the selected material parameters such as:

Young's modulus E: elastic modulus controlling mainly the elastic material response.

compressive strength f_c:

tensile strength f_t,

fracture energy G_F,

onset of nonlinear behavior in compression f_{c0}:

value of plastic strain at compressive strength ε_{cp}:

critical crushing displacement w_d:

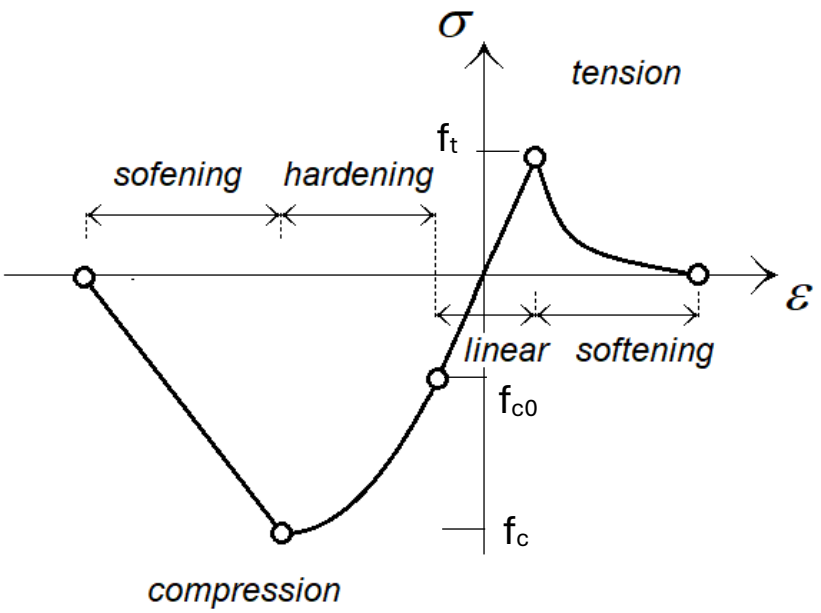


Fig. 6: Meaning of main material parameters to be time dependent.

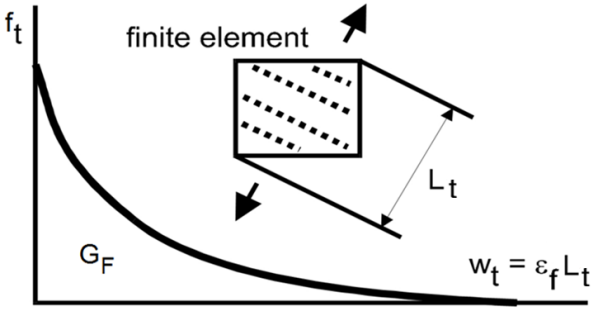


Fig. 7: Meaning of tensile parameters.

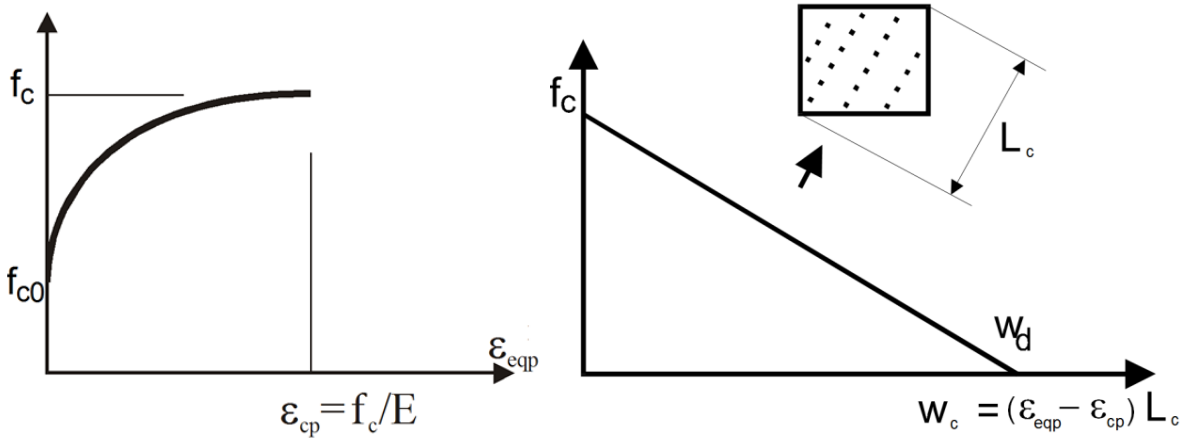


Fig. 8: Meaning of compressive parameters.

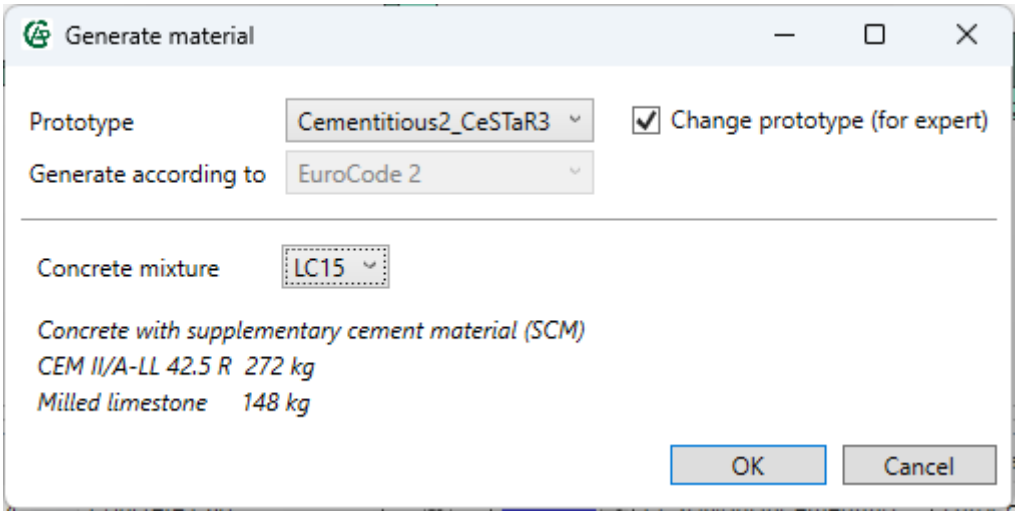


Fig. 9: Selection of appropriate Green Concrete mixture from the available database.

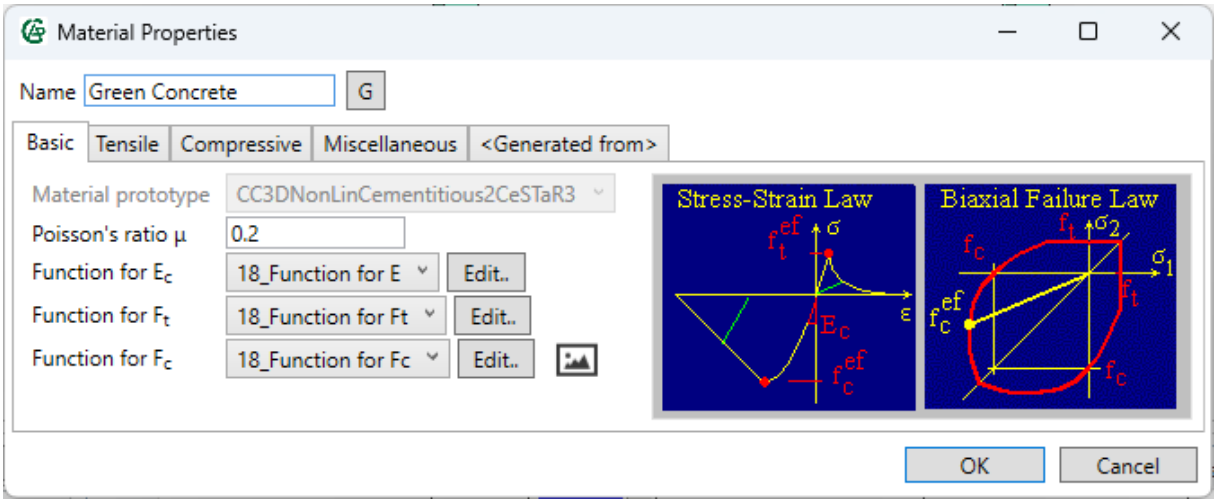


Fig. 10: Green Concrete CeSTaR-3 material dialog enables the modification of the material time evolution laws for the selected material parameters.

Each material parameter time dependency curve can be adjusted and modified by selecting the **Edit** button.

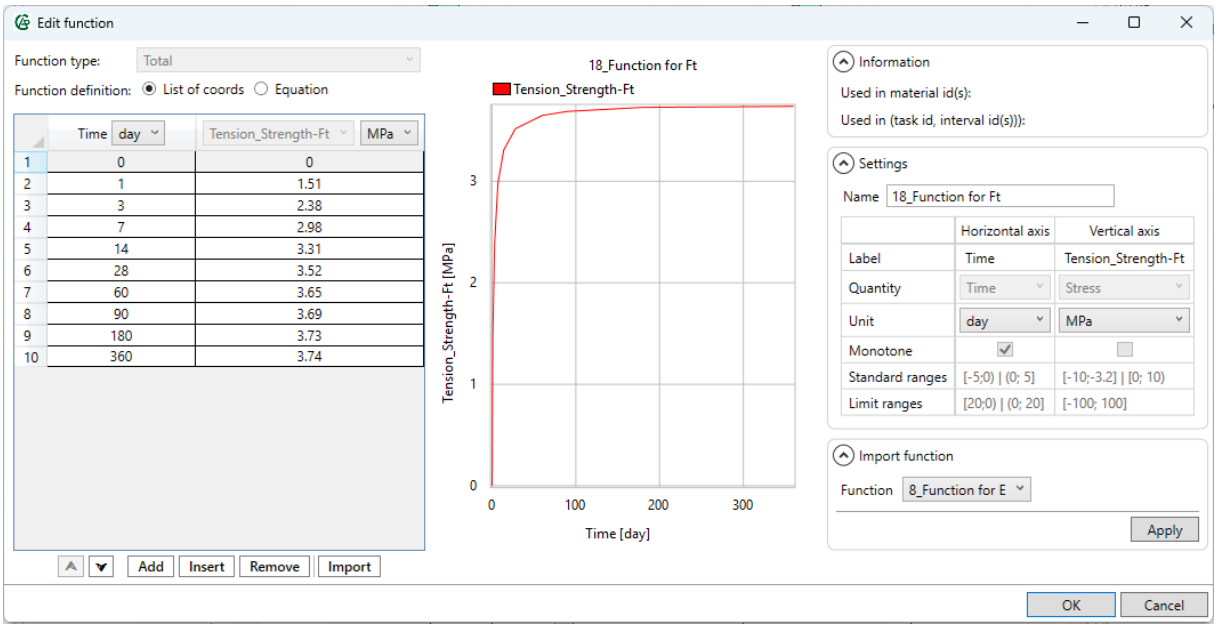


Fig. 11: The edit dialog for material parameter time dependency law.

The dialog in Fig. 11 enables the definition of the material parameter time dependency law by several methods:

- Manually by defining individual points for the parameter time dependence,
- Using copy/paste operation from other application such as for instance Excel or external material database,
- It is also possible to use the equation editor by activating the Equation button at the top.

2.3 Modelling of Construction Sequence Activation

The important part of the modelling of green concrete is the simulation of the construction process inside ATENA software. This feature allows the activation or deactivation of various parts of the model related to various stages of the analysis. It allows also the replacement of material in certain parts of the model.

This functionality is accessible through the Task dialog as shown in Fig. 12

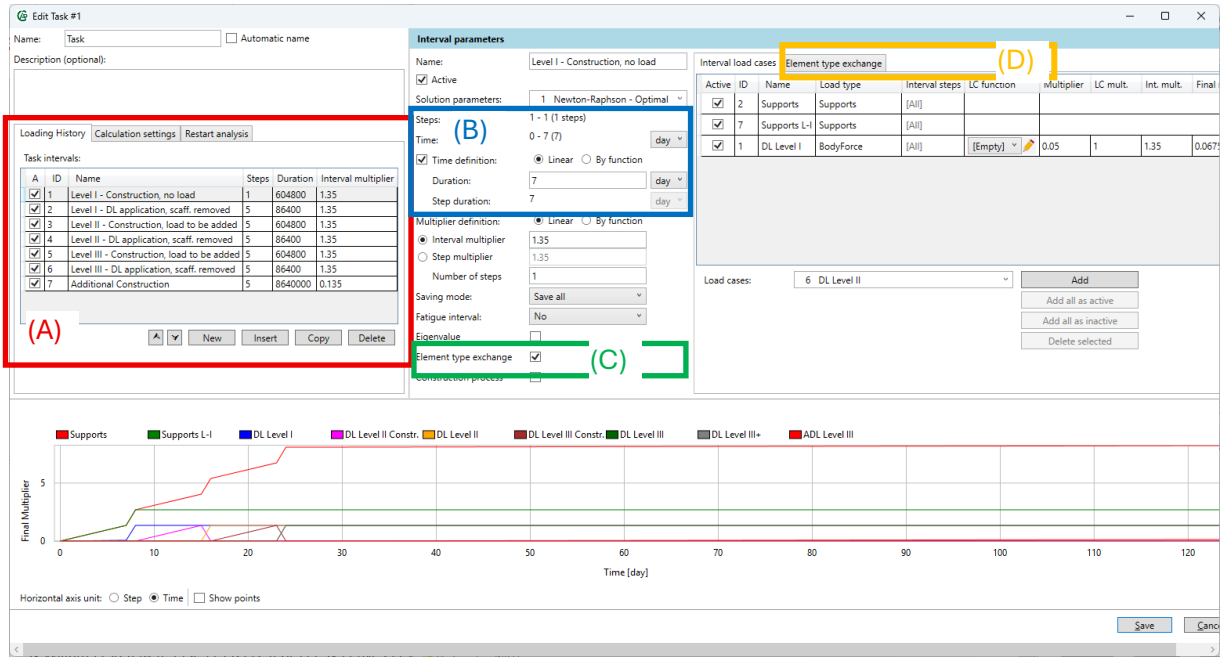


Fig. 12: Definition of construction process inside Task dialog.

This dialog contains important input sections for the definition of the construction process:

(A) This table is used to add Intervals to the analysis history. Each interval represents a special stage in the analysis with same loads, materials and boundary conditions.
(B) This section of the dialog is used to define the time duration of the selected interval.
(C) This button activates the possibility to activate or deactivate different parts of the model as shown in Fig. 13.

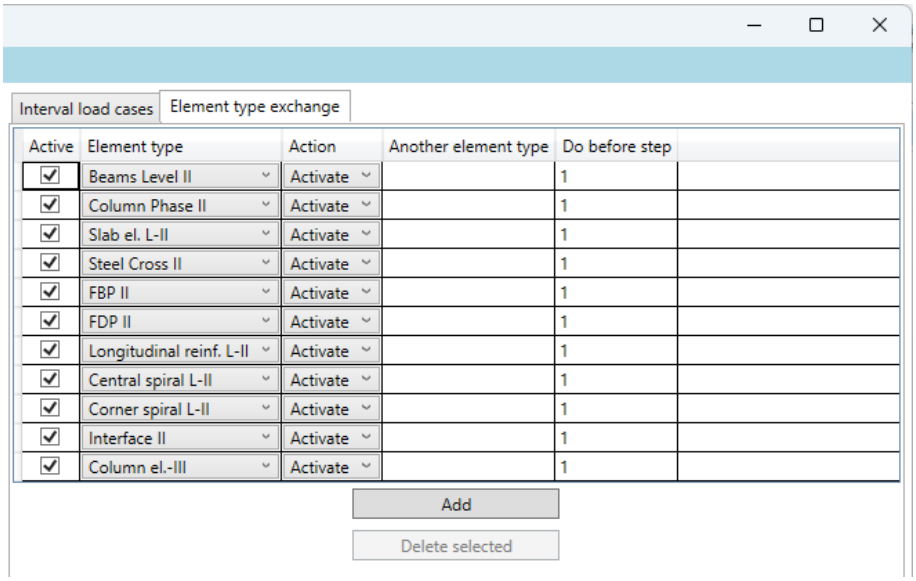


Fig. 13: Table for activation, deactivation or replacement of materials for selected building types.

3 Analysis of Functional Requirements

This section summarizes the **functional requirements of the software result TM04000013-V2** and documents how these requirements have been fulfilled within the ATENA Module for Green Concrete Modelling and Design developed in the CeSTaR-3 project. The functional requirements were derived from the project objectives, the needs of engineering practice, and the characteristics of green concrete materials identified through the experimental programme and compiled in the experimental database **TM04000013-V6**.

The primary goal of the software is to enable **reliable nonlinear numerical simulation of concrete structures made of green concrete with supplementary cementitious materials (SCMs)**, while maintaining compatibility with standard engineering workflows and the existing ATENA software environment. The implemented functionality supports both **research-oriented analyses** and **practical engineering applications**, such as design verification, parametric studies, and virtual testing of structural elements.

The following subsections describe the key functional areas of the software module, including user interaction, material modelling capabilities, analysis features, output and validation tools, and options for extensibility and interoperability.

3.1 User Interface Requirements

The software provides a **user interface fully integrated into the ATENA preprocessor**, enabling intuitive definition and modification of models involving green concrete materials. Dedicated dialog windows allow users to specify material parameters, geometry, reinforcement, boundary conditions, and loading histories using standard ATENA concepts extended by parameters relevant to green concrete and time-dependent behavior.

The interface supports both:

- **graphical input** through interactive dialogs for routine engineering use, and
- **parametric and scripted input** for advanced users performing systematic studies or automation tasks.

This dual approach ensures accessibility for practicing engineers while retaining flexibility for research and development applications.

3.2 Material Model Functionalities

A core functional requirement of the software is the ability to represent the **nonlinear and time-dependent behavior of green concrete** with SCM replacement. The module supports constitutive models that capture:

- nonlinear fracture–plastic behavior in tension and compression,
- confinement effects relevant to reinforced concrete elements,
- time-dependent evolution of material properties, including delayed strength development,
- parameterization based on experimentally derived material data, including values obtained from the experimental database **TM04000013-V6**.

These capabilities enable realistic modelling of green concrete mixtures whose mechanical response differs from conventional Portland-cement-based concrete.

3.3 Analysis and Simulation Capabilities

The developed module is fully compatible with the **nonlinear static and cyclic solvers of the ATENA system**. It supports incremental–iterative solution procedures required for the simulation of cracking, crushing, stiffness degradation, and redistribution of internal forces.

The software enables analyses under:

- monotonic and cyclic loading,
- displacement-controlled and force-controlled regimes,
- construction-sequence-dependent scenarios where material properties evolve with time.

These features allow the software to be applied to a wide range of structural problems relevant to green concrete applications.

3.4 Output, Visualization, and Reporting

The software utilizes standard ATENA post-processing tools to provide comprehensive output for engineering evaluation and interpretation. Available results include:

- load–displacement relationships,
- stress and strain distributions in concrete and reinforcement,
- crack patterns and damage indicators,
- time-dependent evolution of structural response.

Results can be visualized graphically and exported in numerical form for further processing, reporting, or comparison with experimental data.

3.5 Validation and Traceability

Validation of the implemented functionality is a key requirement of the software result. The developed models and workflows have been verified through:

- comparison with experimental results obtained within the CeSTaR-3 project,
- use of experimentally derived material parameters from the database **TM04000013-V6**,
- reference simulations and benchmark studies previously established within the ATENA framework.

The software supports traceability of input parameters, modelling assumptions, and analysis results, ensuring reproducibility and transparency of numerical simulations.

3.6 Extensibility and Interoperability

The module has been designed with an emphasis on **extensibility and long-term usability**. It is fully compatible with existing ATENA data formats and modelling concepts and supports:

- scripting and automation through the ATENA programming interface,
- parametric studies involving systematic variation of material and geometric parameters,
- integration with external data sources and post-processing tools.

This ensures that the software can be further extended and adapted for future research and engineering applications beyond the scope of the CeSTaR-3 project.

4 Technical Documentation and Theoretical Manual

This section provides the **technical documentation and theoretical background** of the ATENA software module for Green Concrete Modelling and Design developed as the software result **TM04000013-V2** within the CeSTaR-3 project. The purpose of this section is to document the **theoretical formulations, numerical methods, and modelling concepts** that underpin the implemented software functionality and to ensure transparency, traceability, and reproducibility of the numerical simulations.

The presented theoretical framework focuses on **nonlinear finite element analysis of concrete structures**, with particular emphasis on aspects relevant to **green concrete containing supplementary cementitious materials (SCMs)**. These aspects include nonlinear fracture–plastic behavior, confinement effects, strain localization, crack width modelling, and time-dependent evolution of material properties. The theoretical formulations described in this section form the basis for the constitutive models and analysis procedures implemented in the ATENA module.

The material models documented here are **directly linked to the software implementation** and are not intended as a general overview of nonlinear concrete modelling. Wherever applicable, the formulations reflect **experimentally validated assumptions and parameters**, including those derived from the experimental database of green concrete materials (**project result TM04000013-V6**). This ensures consistency between experimental observations, numerical modelling, and practical engineering application.

The section is intended for **advanced users, researchers, and developers** who require a deeper understanding of the theoretical background of the software, including its limitations and assumptions. It complements the User Manual by providing the scientific context necessary for informed application, extension, and validation of the software module within both research and engineering workflows.

4.1 Nonlinear Structural Analysis

The nonlinear finite element analysis (NLFEA) was first applied for of reinforced concrete structures already in the 70`s by landmark works of Ngo & Scordelis (1967) [48], Rashid (1968) [50] and Červenka V. & Gerstle (1971) [10]. Various material models for concrete and reinforced concrete were developed in 70's, 80's and 90's such as for instance Suidan & Schnobrich (1973) [54], Lin & Scordelis (1975) [44], De Borst (1986) [34], Rots & Blaauwendrad (1989) [52], Pramono & Willam (1989) [49], Etse (1992) [29] or Lee & Fenves (1998) [43]. These models are typically based on the finite element method and a concrete material model is formulated as a constitutive model applied at each integration point for the evaluation of internal forces. It was soon discovered that material models with strain softening, if not formulated properly, exhibit severe mesh dependency (De Borst & Rots 1989) [37], and tend to zero energy dissipation if the element size is reduced (Bažant 1976) [2].

The crack band approach was introduced by Bažant and Oh (1983) [3] to remedy the convergence towards zero energy dissipation. A more rigorous solution of the ill-posed nature of the strain softening problem represent nonlocal or higher-order continuum models: such as non-local damage model by Bažant & Pijaudier-Cabot (1987) [4], gradient plasticity model by de Borst & Muhlhaus (1992) [36] or gradient damage model by de Borst et. al. (1996) [35]. The non-local models introduce additional material parameters related to an internal material length scale. These models are mathematically rigorous, but currently are seldom used in engineering practice or available in commercial finite element codes.

The limitations of the crack band model in practical engineering calculations, namely when it comes to large finite elements or in the presence of reinforcement, were clearly understood already in the work of Bažant and Oh (1983) [3]. These limitations were described and treatment

was proposed by Červenka (2018a) [17] for the cases when large element sizes as well as small ones are used in the finite element nonlinear analyses.

Until recently the application of these advanced nonlinear analyses in design practice was difficult since it was not compatible in many cases with the existing design codes and engineering practice, which is still mostly based on linear analysis. The fib Model Code 2010 (MC2010) [32] introduced a comprehensive system for the treatment of safety and model uncertainty for structural assessment and design based on nonlinear analysis. This discrepancy was eliminated by introduction of the global safety formats in the fib Model Code 2010 [32] and in the evolution of the new fib Model Code 2020 and of the Eurocodes. The model uncertainty of the approaches based on NLFEA reflects the knowledge quality of numerical simulations and serves as an important reliability tool.

A design verification based on NLFEA offers an alternative to the standard design method based on elastic analysis. It is typically used for complex structures, where a robust reliability assessment is substantiated and in the cases of accidents and forensic investigations.

The original publications to the finite element method are due to J. Argyris 0 [1], M.J. Turner and R.W.Clough (1956) [57], and O.C. Zienkiewicz (1967) [59]. It became the most significant tool for structural analysis and a basis of computer codes for solutions of engineering problems. A displacement-based version of the finite element method is the most popular version for the engineering applications and is used in this report.

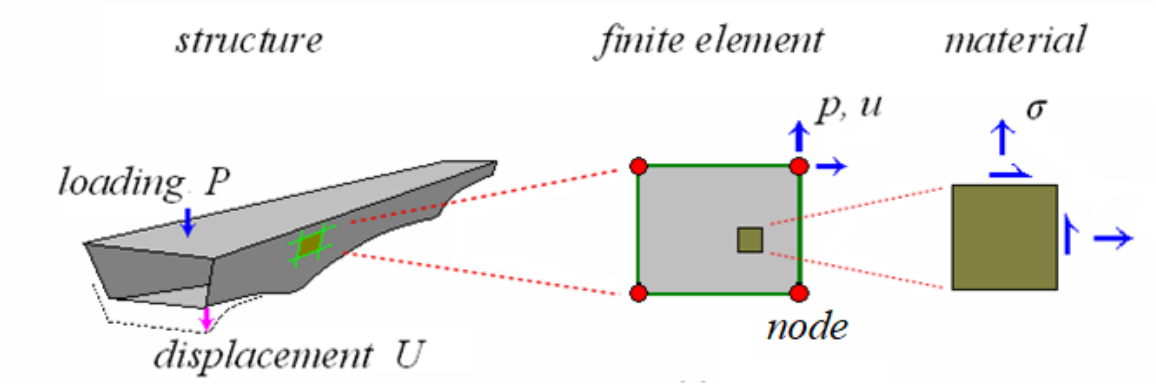


Fig. 14. NLFEA solution scheme.

The exact solution of the continuum problem is approximated by a solution of the set of linear equations for discrete number of nodal displacements. This discretization is illustrated in Fig. 14, where three solution levels are recognized, namely structure (describing the engineering task to be solved), finite element (defining an approximation by the finite element mesh), and material (defining a non-linear behavior). A nonlinear response (due to material behavior or geometry) is solved by an iterative Newton-Raphson method. The solution is illustrated in Fig. 15and is described by the following set of matrix equations:

$$K_{n,i}\,\Delta U_{n,i} = P_n - R_{n,i} \tag{1}$$

$$K_{n,i} = \sum_{j=1}^{m} k_j,\ R_{n,i} = \sum_{j=1}^{m} r_j\ ,\ U_{n+1,i} = U_{n,i} + \Delta U_{n,i} \tag{2}$$

In this: K_i – stiffness matrix of the structure at the load step n and iteration i, ΔU_i – vector of displacement increments, $U_{n,i}$ – vector of total displacements of the structure in step n and iteration i, P_n – vector of total nodal forces at the load step n, $R_{n,i}$ – vector of resisting nodal forces, k_j – element stiffness matrix, r_j – vector of resisting element nodal forces.

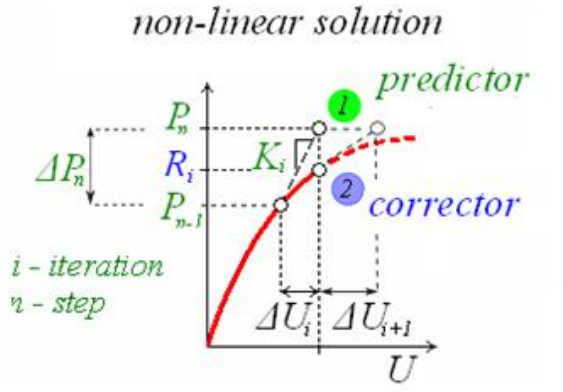


Fig. 15. Non-linear iterative solution.

The number of rows in the matrix Eq. (1) corresponds to the number of nodal displacements in the structure. Eq. (2) represents the assembly of the global stiffness matrix, nodal forces and nodal displacements as a summation from local element entities, where m is the number of finite elements. It should be noted that that the above formulation is schematic and does not indicate the detail structure of the matrices due to the displacement boundary conditions and corresponding reactions.

The non-linear response is solved in increments. In a load step, a displacement increment $\Delta U_{n,i}$ due to the load increment ΔP_n is estimated by the *predictor*, marked as point 1. It estimates the response based on the element stiffness matrix k, which is defined in the matrix format as follows:

$$\varepsilon = Bu, \;\; \sigma = D\varepsilon, k = \int_v \; B^T D \; B \; dv \tag{3}$$

where ε – element strains, σ – element stresses, B – strain-displacement matrix for given finite element formulation, u – element nodal displacements, k – element stiffness matrix. The Integration is performed over the volume of the finite element. The matrix B can also include high order terms and reflect the geometrical non-linearity. Matrix D is based on the linear elastic theory (with two parameters, elastic modulus and Poison's ratio). It is based on a tangent elastic modulus, but alternative methods are available based on other values. Eq.(3) follows form the theorem of minimum of total potential energy and more detailed derivation can be found, for example, in Zienkiewicz (1967) [59].

Due to the linear approximation the solution obtained by the predictor differs from a nonlinear response. Therefore, a correct resistance is found by the *corrector* as follows:

$$\sigma = F(\sigma, \varepsilon), \; r = \int_v \; B^T \sigma \; dv \tag{4}$$

In this $F(\sigma, \varepsilon)$ – constitutive law, and r – vector of resisting nodal forces consistent with the constitutive law. The constitutive law is a function of stress and strain and typically include additional parameters. In this presentation two laws, based on the theories of plasticity and fracture are described.

In case of a unique solution the difference $P_n - R_{n,i}$ wannish and the iterative solution converges to a nonlinear response. The process can be refined by adopting methods to improve the iteration stability, such as the methods of line-search and arc-length.

4.2 Constitutive Models

4.2.1 Constitutive model for concrete

Considering a large variety of existing material models or approaches to nonlinear analysis, it is impossible to provide guidelines or recommendations for their general application in engineering practice. It is therefore evident that very detailed guidelines or approaches for the treatment of for instance the modelling uncertainties can be only specific to certain class of constitutive models or even to a particular material model or even only to a single software. A comprehensive summary of this topic is available for instance at Jirásek & Bažant (2001) [42]. The models presented in this paper are based on the research of the authors, which are implemented in the finite element software ATENA, Červenka et al. (2025) [5] . Some of the conclusions are valid only for this software or at most are applicable for the class of models based on smeared crack approach and crack band method. However, the presented safety formats or the general treatment of model uncertainties is applicable for other nonlinear models for concrete structures as well. The material model used in this paper is a fracture-plastic model described in more detail in Červenka at al. (1998) [14] and Červenka & Papanikolaou (2008) [15].

The constitutive model formulation assumes small strains and is based on the strain decomposition into elastic (ε_{ij}^{e}), plastic (ε_{ij}^{p}) and fracture (ε_{ij}^{f}) components. The stress development is described by a rate equation reflecting the progressive damage (concrete cracking) and plastic yielding (concrete crushing):

$$\dot{\sigma}_{ij} = D_{ijkl} \cdot (\dot{\varepsilon}_{kl} - \dot{\varepsilon}^{p}_{kl} - \dot{\varepsilon}^{f}_{kl}) \tag{5}$$

The flow rules govern the evolution of plastic and fracturing strains:

$$\text{Plastic model:} \qquad \dot{\varepsilon}^{p}_{ij} = \dot{\lambda}^{p} \cdot m^{p}_{ij}, m^{p}_{ij} = \frac{\partial g^{p}}{\partial \sigma_{ij}} \tag{6}$$

$$\text{Fracture model:} \qquad \dot{\varepsilon}^{f}_{ij} = \dot{\lambda}^{f} \cdot m^{f}_{ij}, \; m^{f}_{ij} = \frac{\partial g^{f}}{\partial \sigma_{ij}} \tag{7}$$

where $\dot{\lambda}^{p}$ is the plastic multiplier rate and g^{p} is the plastic potential function, $\dot{\lambda}^{f}$ is the inelastic fracturing multiplier and g^{f} is the potential defining the direction of inelastic fracturing strains. The multipliers are evaluated from the consistency conditions.

4.2.1.1 Compressive failure model

Compressive behavior is governed by the plasticity of concrete in multiaxial stress state with nonlinear hardening/softening. In the plasticity model, Eq.(5), the plastic strain rate $\dot{\bar{\varepsilon}}_{p}$ is calculated from the consistency condition. λ_{p} is the plastic multiplier, and $\bar{n}$ and $\bar{m}$ are stress derivatives of the plastic and potential surface respectively. The plastic surface F_{p} is defined by the three-parameter criterion, (f_{c}, f_{t}, e), according to Menetrey & Willam (1995), Fig. 16.

$$F_{p} = \left[\sqrt{1.5}\,\frac{\rho}{\hat{f}_{c}}\right]^{2} + m\left[\frac{\rho}{\sqrt{6}\hat{f}_{c}} r(\theta, e) + \frac{\xi}{\sqrt{3}\hat{f}_{c}}\right] - c = 0 \tag{8}$$

$$m = 3\,\frac{\hat{f}_{c}^{2} - f_{t}^{2}}{\hat{f}_{c} f_{t}}\,\frac{e}{e+1}$$

$$r(\theta, e) = \frac{4(1-e^{2})\cos^{2}\theta + (2e-1)^{2}}{2(1-e^{2})\cos\theta + (2e-1)\left[4(1-e^{2})\cos^{2}\theta + 5e^{2} - 4e\right]^{\frac{1}{2}}}$$

In the above equations, (ξ, ρ, θ) are the Haigh-Westergaard stress coordinates and f_c and f_t are the compressive strength and tensile strength, respectively. Parameter $e \in (0.5, 1.0)$ defines the roundness of the Menétrey-Willam failure surface, with a recommended value e = 0.52. Note, that the shape of the failure function reflects the typical behavior of concrete, namely the confinement effect, described by the cone opening with increased hydrostatic pressure, and a quasi-triangular shape on the deviatoric plane. The plastic surface is not constant. Its evolution is governed by the equivalent plastic strain ε_{eqp} :

$$\dot{\varepsilon}_{eqp} = \min\left(\dot{\varepsilon}_{p1}, \dot{\varepsilon}_{p2}, \dot{\varepsilon}_{p3}\right) \tag{9}$$

where ε_{pi} is the i-th component of the principal plastic strains. The hardening is modeled by adjusting the compressive strength $\widehat{f}_c$, while the softening is controlled through the parameter c:

hardening $\varepsilon_{eqp} \in \langle -\varepsilon_{cp}; 0 \rangle$ $\varepsilon_{eqp} = \langle -\varepsilon_{cp}; 0 \rangle$

$$\widehat{f}_c(\varepsilon_{eqp}) = f_{c0} + (f_c - f_{c0}) \sqrt{1 - \left(\frac{\varepsilon_{cp} - \varepsilon_{eqp}}{\varepsilon_{cp}} \right)^2} \tag{10}$$

softening $\varepsilon_{eqp} \in \langle -\infty; -\varepsilon_{cp} \rangle$

$$\begin{aligned} & c = \left(1 - \frac{w_c}{w_{co}} \right)^2, \quad w_c \in \langle -w_d; 0 \rangle \\ & c = 0, \quad w_c \in (-\infty; w_d) \\ & w_c = (\varepsilon_{eqp} - \varepsilon_{cp}) L_c^{'} \end{aligned} \tag{11}$$

In the above formulas, ε_{cp} - plastic strain when the compression strength f_c is reached in a uni-axial compression test, f_{c0} - onset of nonlinear behavior in compression, w_d is the critical value of compressive displacement and w_c - displacement of the compressive plastic zone, $L_c^{'}$, - crush band, as will be described in Section 4.2.1.4.

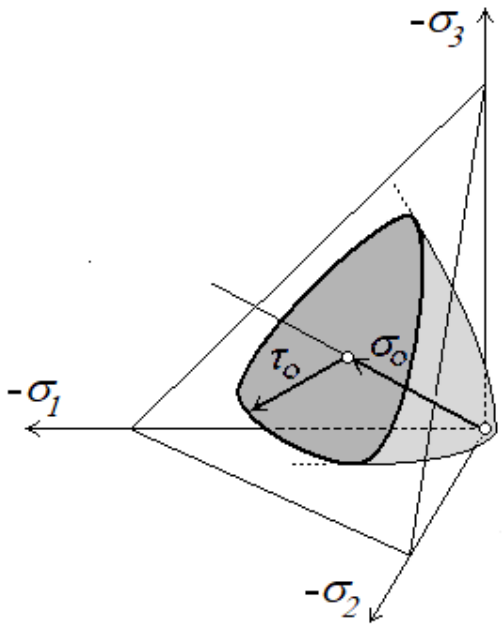


Fig. 16. Concrete failure criterion due to Menetrey&Willam 1995 [46].

The plastic flow in compression is not associated (normal) to the plastic function in Eq.(8), see Fig. 17. It is controlled by the parameter β, which defines a volumetric change during the crushing process:

$$G_p(\bar{\sigma}) = \beta \frac{1}{\sqrt{3}} I_1 + \sqrt{2J_2} \tag{12}$$

In which $\beta > 0$ - volume expansion, $\beta < 0$ - material compaction, $\beta = 1$ - material volume is preserved, I_1 - 1st invariant of stress vector and J_2 - 2nd invariant of deviatoric stress vector.

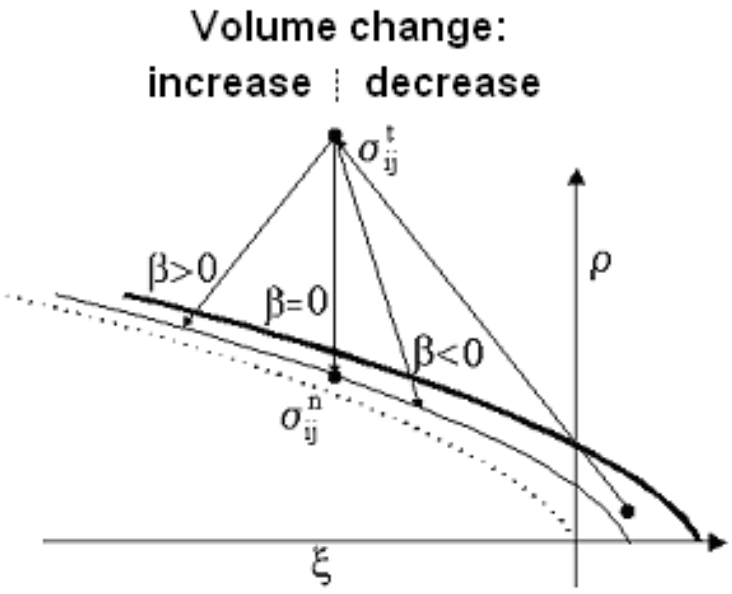


Fig. 17. Non-associated plastic flow in compression.

4.2.1.2 Tensile failure model - smeared cracks

In an uncracked state the concrete is represented by homogenous isotropic body. A crack forms when the tensile strength is reached in principal tension. In the smeared crack concept, a discrete crack is represented by a damage due to a family of parallel cracks distributed in a volume associated with a material point and introduces an orthotropy in the material. The considered stress and strain variables are indicated in Fig. 18. (For clarity a 2D plane stress state is illustrated, but an extension to a crack in 3D is straightforward). Two crack models can be considered:

(a) Fixed crack model. The material axes of orthotropy m_1 and m_2 are fixed in the subsequent nonlinear analysis, whereas the weak axis m_1 is normal to the crack direction and the strong axis m_2 is parallel with the crack direction. Under a general loading, the axes of principal stress σ_1, strain ε_1, and orthotropy m_1 do not coincide and a shear stress τ^{cr} can occur on the crack face. Angle α is a normal to the crack direction and ϕ is a deviation of the maximal principal strain form the crack normal.

(b) Rotated crack model. The material axis m_1 is coincident with the principal tension ε_1, and with the principal stress σ_1, and angle $\varphi=0$. Thus, when the principal stress axes rotate, so does the crack direction. No shear stress occurs on the crack face, $\tau^{cr} = 0$.

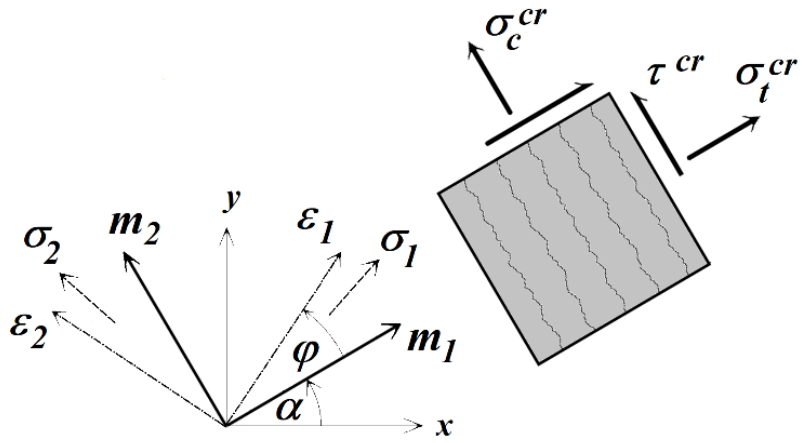


Fig. 18. Cracked concrete stress and strain state in 2D situation.

The two crack models can be combined in a mixed formulation, in which the crack direction rotates until a threshold softening stress $\sigma = c_{fix} f_t$ is reached and is fixed thereafter. This model reflects the experimentally observed crack formation process (e.g. Bazant , where the threshold softening stress reflects a transition from a fracture process zone to a localized crack.

4.2.1.3 Softening laws of cracked concrete

The cohesive stresses on the crack face gradually decrease with growing deformation, which is referred as strain softening and is controlled by a nonlinear fracture mechanics. The following fracture types are recognized in cracked concrete: (1) Crack opening (mode I fracture); (2) Crack sliding in shear (mode II and III fracture); (3) Compressive failure (crushing fracture) due to the normal stress parallel with the crack direction.

(1) Crack opening law by Hordijk (1991) [38] describes the softening of the cohesive crack model, Fig. 19:

$$\frac{\sigma_t^{cr}}{f_t} = [1 + (c_1 \frac{w_t}{w_{to}})^3]\ exp(-c_2 \frac{w_t}{w_{to}}) - \frac{w_t}{w_{to}}(1 + {c_1}^3)\ exp(-c_2) \quad (13)$$

$$w_{to} = 5.14 \frac{G_f}{f_t},\ c_1 = 3,\ c_2 = 6.93.$$

where w_t – crack width, σ_t^{cr} – tensile stress on the crack face.

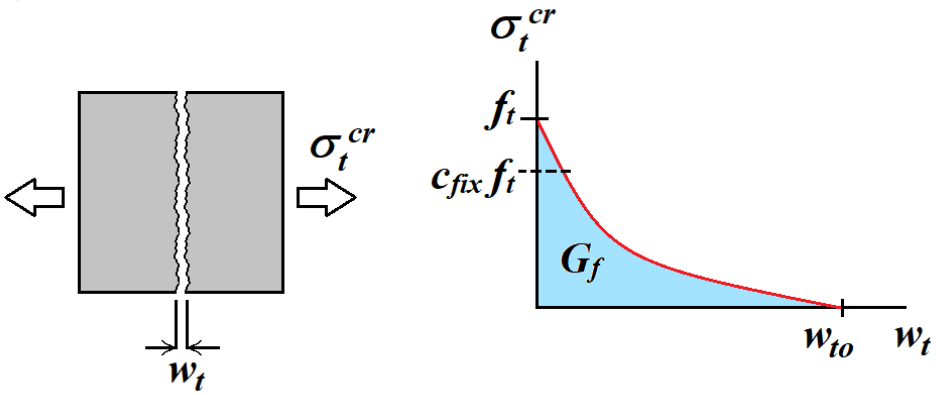


Fig. 19. Crack opening law.

(2) The shear stress in cracked concrete is described by models for stiffness and strength:

$$\tau^{cr} = s_F K_n^{cr} \gamma^{cr},\ K_n^{cr} = \frac{\sigma_t^{cr}}{\varepsilon_t^{cr}} \quad (14)$$

$$\tau^{cr} \le \frac{0.18\sqrt{f_c}}{0.31 + \frac{24 w_t}{a_g + 16}} \quad (15)$$

The shear stress in Eq.(14) is based on the shear retention idea, in which s_F – shear factor, K_n^{cr} - stiffness of the crack opening. (For the typical value s_F=20 the model gives the fracture energy in shear $G_F^{II} \sim 10\, G_F^I$, where G_F^{II} and G_F^I are the fracture energy is shear sliding and crack opening, respectively.)

(3) Compressive strength is reduced due to cracking damage:

$$\sigma_c^{cr} = r_c f_c \quad , \quad r_c = \frac{1}{0.8 + 170\, \varepsilon_t^{cr}},\ r_c^{\lim} \le r_c \le 1.0 \quad (16)$$

Where r_c – strength reduction factor and $r_c^{\lim}$ - limit of the reduction. Equations (15) and (16) are according to MCFT proposed by Collins (Bentz at al. 2006) [6].

A typical stress-strain curve resulting from the above formulations including tension and compression is illustrated in Fig. 20. The formulation also allows more complex forms, such as including a tension hardening, etc.

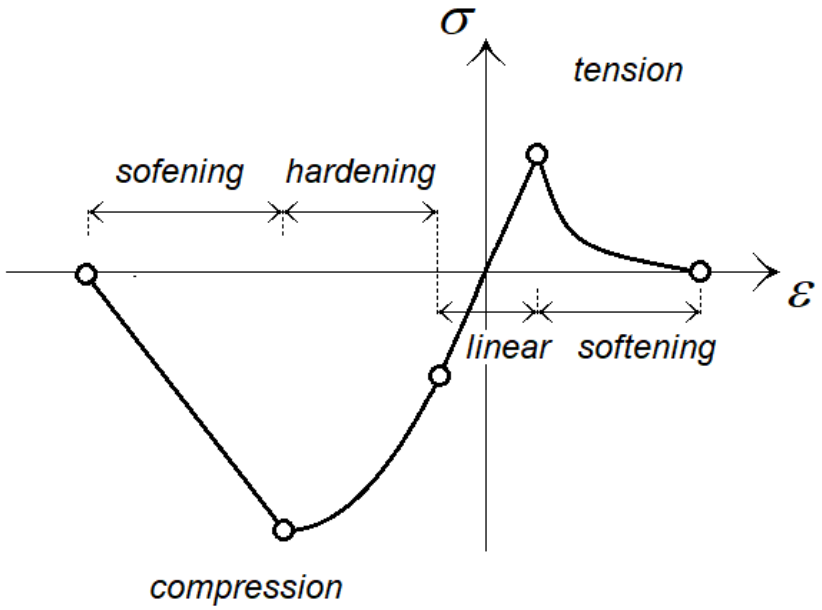


Fig. 20. Stress-strain curve of concrete.

4.2.1.4 Strain localization limiter

The characteristic feature of concrete failure is the stress softening in the post-peak loading range, which is the source of instability and must be appropriately treated. In the smeared crack model this leads to a zero-energy dissipation when the constitutive model is based on strains. To remedy this malfunction an approach based on the localization limiter was introduced. The crack band concept was proposed by Bazant (1983) [3], and extended by Cervenka (1995) [12]. The idea of crack band is illustrated in Fig. 21. It is assumed, that the failure energy dissipates within the volume described by localization bands. The band size is defined as a projection of the element size. In tension:

$$w_t = \varepsilon_f L_t' \quad , \quad L_t' = \alpha\gamma L_t \ , \ \gamma = 1 + (\gamma_{\max} - 1)\frac{\theta_1}{45} \ , \ \theta_1 \in \langle 0;45 \rangle \ , \ \gamma_{\max} = 1.5 \tag{17}$$

where w – displacement of the crack band (crack width), ε_f – fracture strain, L_t' - crack band size, and L_t - element size projection parallel to the crack plane. α, γ are parameters for element type (Slobe 2013) and crack orientation angle θ, respectively. The orientation factor γ reflects the fact that for inclined cracks the crack band size exceeds the row of single elements as illustrated in Fig. 22. (For a low order iso-parametric quadrilateral element with full integration $\alpha = 1$, for the crack direction aligned with element sides $\gamma = 1$.)

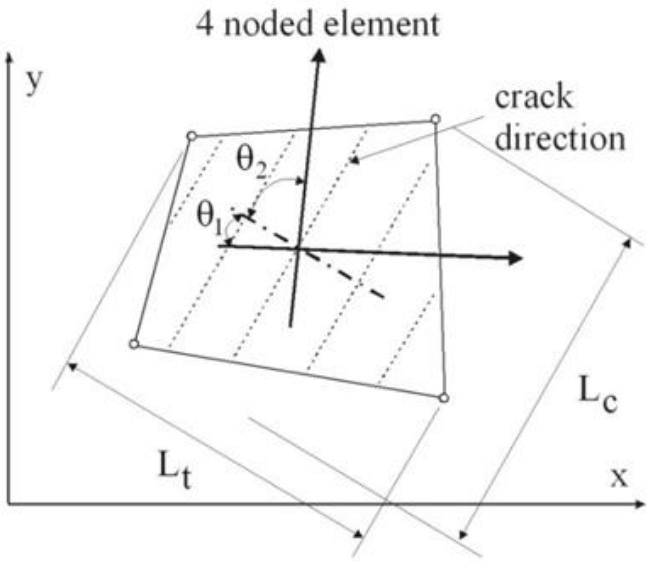


Fig. 21. Crack and crush bands in 2D situation.

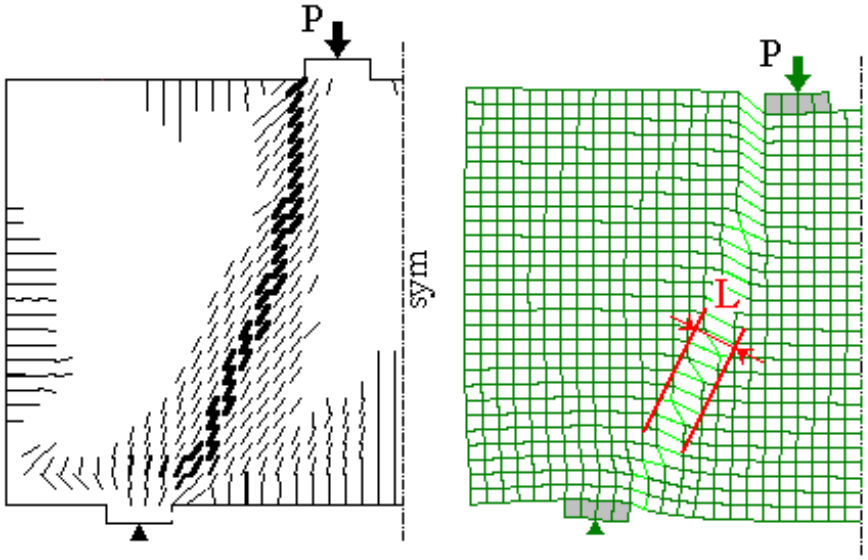


Fig. 22. Crack band for cracks not parallel with the finite element edges.

In analogy, for compression a similar model is used, Červenka J. (2014):

$$w_c = \varepsilon_p L_c' \quad , L_c' = \alpha\gamma L_c \, , \quad \gamma = 1 + (\gamma_{\max} - 1)\frac{\theta_2}{45} \, , \quad \theta_2 \in \langle 0; 45 \rangle \tag{18}$$

where w_d – displacement of the compression band, ε_p – plastic strain, L_c' - crush band size, and L_c - element size projection normal to the crack plane.

Cervenka (2018) [17] demonstrated the limits of crack band approach for modelling of reinforced concrete when very large or very small finite elements are used.

The problem is demonstrated by experiments of cracked reinforced concrete panels, Cervenka and Gerstle (1971) [10], see Fig. 23. When very large finite elements are used, a standard assumption that a single crack will develop inside the finite element, is not valid anymore. Due to a variety of reasons, but namely due to the reinforcement, the cracks localize at distances smaller

than the used finite element size. For very small element finite elements the crack spacing cannot be smaller than a non-homogeneity of material.

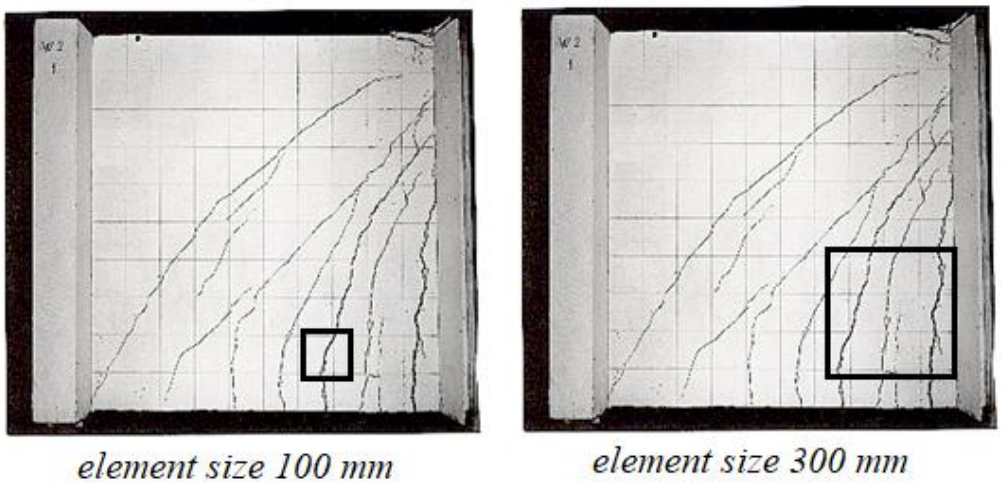


Fig. 23. Crack spacing in small and large finite elements, Cervenka&Gerstle 1971 [10].

A similar defective behavior can be observed in compression, as illustrated on the case of cylinder test subjected to compression in Fig. 24. In compression, namely due to the effects of the dilatancy and hardening, the plastic deformation may involve many elements. In this case a crush band size should be defined by the plastic deformation zone, which is approximately given by the smallest structural dimension such as the thickness.

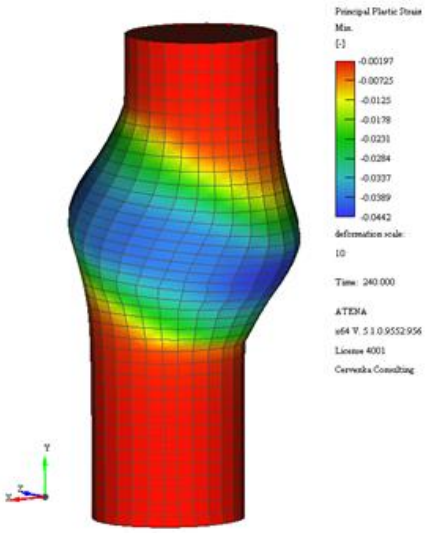


Fig. 24. Strain localization in compressive cylinder test.

In summary three localization limiters are proposed (Červenka j. 2018):

$L_{t,\min}$ - minimal crack spacing limiter in tension related to aggregate size

$L_{t,\max}$ - maximal crack spacing limiter in tension related to reinforcement arrangement

$L_{c,\min}$ - minimal crush band limiter in compression related to the minimal size of the compression zone.

4.2.1.5 Crack width simulation based on the smeared crack model

The smeared crack approach simulates a real discrete crack indirectly by a damage distributed within a crack band. This simplifies the numerical analysis based on the finite elements but implies a question about the crack width. This problem was recently investigated by the authors in Cervenka et al. (2022) [22]. The study included 18 beam specimens tested at Vilnius Tech and covered a range of reinforcing arrangements and reinforcement types. Only the group of five beams representing typical RC beams is described here for brevity. The beam geometry and cross sections of 5 beams are shown in Fig. 25. The beams were subjected to a four-point bending scheme, where central span of 1000 mm was under a constant moment loading. The specimen series covers two reinforcing ratios, ρ = 1, and 0.6, and bar profiles are 6, 14, 18, and 22 mm. The concrete cover of bars was 20 mm. The bars were in a single layer (except of the beam S 1-6). The concrete strength obtained in laboratory was in average f_c=49 MPa. The other parameters needed for the simulation were derived from the strength using the model code MC2010 relations as: elastic modulus E=35.5 GPa, tensile strength ft=3.6 MPa and fracture energy Gf=147 N/m. (The parameters slightly changed for each beam).

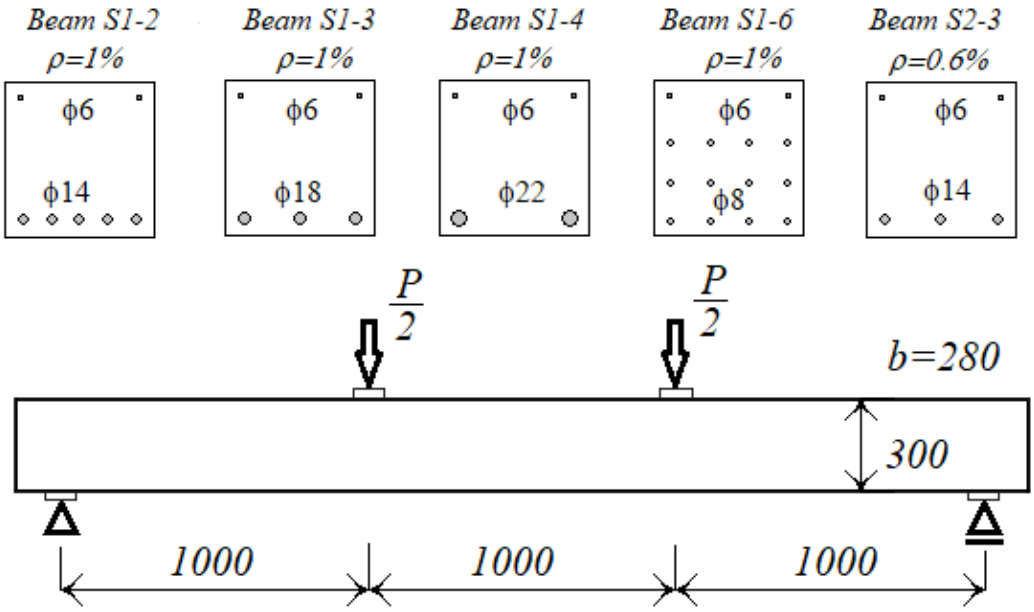


Fig. 25. Dimensions in mm and cross section reinforcement of beams.

The mean and maximum crack widths were investigated experimentally and numerically within the central span on the concrete surface along the reinforcement line. The simulation of the beam series was in 3D representation, with the smeared crack model according to the description above.

The model uncertainty of the crack width was defined as $\theta_w = w_{exp}/w_{sim}$, where w_{exp} and w_{sim} are the crack widths from experiment and simulation, respectively. The summary of the model uncertainty of this series is shown in Tab. 1 for two mesh sizes M30, M15 (element sizes 30 and 15 mm, respectively). θ_μ and θ_{max} are model uncertainties for mean and maximum crack width, respectively. The investigation revealed an excellent agreement between simulation and experiment for both meshes. (Note that θ =1 indicates a perfect match). Also the ratio between maximum and mean crack widths w_{max}/w_μ agrees well with the experiment. The low coefficient of variation confirms a strong validity of the conclusions.

Tab. 1: Crack width model uncertainty.

	Experiment			*Simulation M30*				*Simulation M15*				
Beam	w_μ	w_{max}	$\frac{w_{max}}{w_\mu}$	w_μ	w_{max}	$\frac{w_{max}}{w_\mu}$	$q_{w,mean}$	$q_{w,max}$	w_μ	w_{max}	$\frac{w_{max}}{w_\mu}$	$q_{w,mean}$	$q_{w,max}$
S1-2	0.071	0.102	1.43	0.062	0.106	1.710	1.150	0.965	0.076	0.112	1.460	0.931	0.915
S1-3	0.085	0.140	1.64	0.075	0.142	1.899	1.145	0.987	0.071	0.152	2.137	1.199	0.919
S1-4	0.063	0.120	1.90	0.073	0.124	1.687	0.862	0.970	0.080	0.149	1.877	0.795	0.804
S1-6	0.038	0.060	1.57	0.048	0.070	1.456	0.789	0.852	0.042	0.064	1.535	0.919	0.941
S2-3	0.055	0.120	2.18	0.056	0.091	1.624	0.982	1.319	0.049	0.107	2.172	1.120	1.125
Average			**1.745**			**1.675**	**0.985**	**1.019**			**1.836**	**0.993**	**0.941**
COV			**0.170**			**0.095**	**0.165**	**0.173**			**0.180**	**0.165**	**0.123**

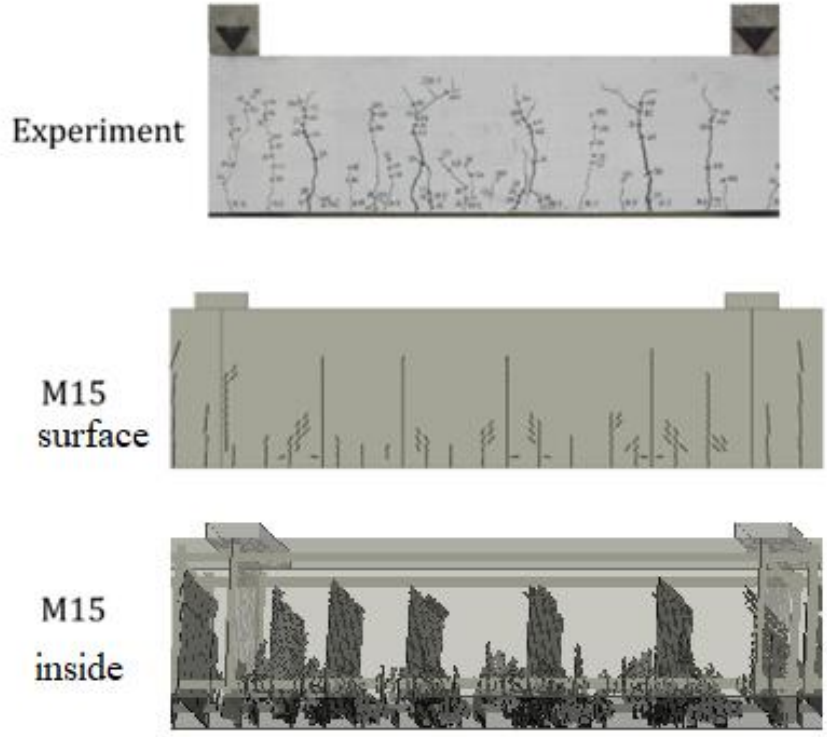


Fig. 26. Cracks in experiment and simulation.

The cracking analysis is illustrated for the beam S1-4 in Fig. 26. It compares the crack patterns on the surface between the experiment and simulation with mesh M15. It also shows the cracks inside the beam obtained by simulation, which is not available from experiments. It confirms the image of primary and secondary cracks. The primary cracks are the main open cracks extending through the whole beam width. The secondary cracks are smaller inclined cracks near the reinforcement, which reflect the force transfer from the bars to concrete and form a bond mechanism. The research work in this project provided a conclusion, that the cracking analysis based on fracture mechanics is a sufficient model for cracking simulation and crack width prediction in reinforced concrete, and that it well describes a bond mechanism.

4.2.1.6 Reinforcement, cables, contacts and other models

Reinforcement and prestressing cables are, of course, important components of the reinforced concrete structures. They are typically represented by truss elements in a uniaxial stress state and constitutive models are described by multilinear functions. They are embedded in the solid elements of concrete. The bond interaction between reinforcing bars and can be included by considering interface elements on bar surface. Interaction between solid objects can be also modelled by including contact elements reflecting the surface movements.

The issues requiring address environmental effects, durability and time require to include the classes of solutions based on transport mechanics, creep, dynamics and others. They can be coupled with the presented model and offer multi-branch solutions.

These and other features are necessary for a complete successful simulation of concrete structures. However, for brevity they are not included in this document, and reader should consult the appropriate references.

4.3 Safety formats for nonlinear analysis of reinforced concrete

7.1 Design condition

The design condition is generally formulated as:

$$E_{\mathrm{d}} < R_{\mathrm{d}} \tag{19}$$

where E_{d}, R_{d} represent the design values of load and resistance, respectively. They include the specified safety margins based on a probabilistic theory as proposed by Vrouwenvelder (2002) [58] and the JCSS Probalilistic Model Code (2001) [39]. In the present approach, for simplicity, the random effects of load and resistance are considered separately.

Two types of uncertainties are recognized in the reliability analysis, Swiler (2007) [55]. The first one, referred as “aleatory” uncertainty, is due to inherent random properties of material and dimensions given by the concrete technology and is referred in the further discussion as the material uncertainty. The second one, referred a “epistemic”, is reflecting the knowledge level of the background theoretical model, and is called as the model uncertainty.

The design condition in Eq. (19) in the standard design practice is applied to critical cross-sections. The inconsistency of this concept is well known because different assumptions are used for the calculation of the load effect of internal forces E_d (obtained by a linear analysis) and, the cross-section resistance R_d (nonlinear material model). In general, the section forces may change due to a stress redistribution during the non-linear response. Furthermore, the local safety checks do not reflect a reliability of the entire system. In a nonlinear analysis a nonlinear material response is implicitly reflected in Eq. (4). Therefore a local check of the condition in Eq.(19) is satisfied by definition. However, a global check is required. The load effect E_{d} in the equation (19) is considered at the global level (typically it represents a of the relevant load combination), and analogically the resistance R_{d} is the ultimate load level at failure for the given load combination imposed on the structure.

MC2010 [32] introduces four methods for the global assessment using nonlinear analysis, which differ in estimate of the random parameters of the ultimate resistance. In this study only two most prominent methods will be treated. i.e. the partial factor (PFM) (Section 7.11.3.4) and ECoV method (Section 7.11.3.3 of fib model code 2010). These two methods are expected to be included in the new version of Eurocodes, and therefore they will be treated in more detail. A full probabilistic approach on the other hand can be used as an exact reference solution.

7.2 Partial safety factor method (PFM)

This approach is most appealing for practicing engineers as it is a quite natural extension of the current design practice. The design resistance R_d is obtained as:

$$R_{\mathrm{d}} = \frac{R\left(X_{\mathrm{d}}; a_{\mathrm{nom}}\right)}{\gamma_{\mathrm{Rd}}}, \quad X_{\mathrm{d}} = \frac{X_{\mathrm{k}}}{\gamma_{\mathrm{m}}} \tag{20}$$

Where,

$R\left(X_{\mathrm{d}}; a_{\mathrm{nom}}\right)$ - structural resistance by numerical simulation,

X_{d} - design value of the material property considering material and geometric uncertainties, but excluding the model uncertainty,

X_k - characteristic value of material parameter,

γ_{m} - partial safety factor of material (for concrete, reinforcement, etc.) without model uncertainty,

a_{nom} - nominal value of the geometric property, i.e. for instance reinforcement depth or element dimension,

γ_{Rd} - partial safety factor for the model uncertainty of NLFEA (to be described later).

7.3 GFM - ECoV method - an estimate of the coefficient of variation

ECoV method originally proposed by Červenka (2008 [18], 2013 [19]) is a semi probabilistic approach assuming that the statistical distribution of resistance due to the variability of materials can be estimated by Eq.(21)

$$V_{\mathrm{m}} = \frac{1}{1.65} \ln\left(\frac{R_{\mathrm{m}}}{R_{\mathrm{k}}}\right), \tag{21}$$

where the coefficient of variation V_m is estimated by two resistance values, the mean resistance R_{m} (based on the mean material parameters) and characteristic resistance R_{k}. These two points describe a scatter of the resistance. The underlying assumption is that the statistical distribution of the resistance is according to a lognormal distribution:

The global safety factor for material uncertainty of the resistance can be calculated as:

$$\gamma_{\mathrm{m}} = \exp(\alpha_{\mathrm{R}} \beta V_{\mathrm{m}}) \cong \exp(3.04\, V_{\mathrm{m}}) \tag{22}$$

Typical values are, for sensitivity factor α_R=0.8 and reliability index β=3.8 (50 years reference period). The design value of resistance is calculated as:

$$R_{\mathrm{d}} = \frac{R_{\mathrm{m}}}{\gamma_{\mathrm{m}} \gamma_{\mathrm{Rd}}} \tag{23}$$

Alternatively, depending on data availability, a total coefficient of variation of resistance V_R can be estimated as:

$$V_{\mathrm{R}} = \sqrt{\sum_{i}^{n} V_i^2}, \tag{24}$$

Where V_i stands for the coefficient of variation of the random effect *i* of material, geometry, model uncertainty, etc. Then, Eq. (22) and (23) read:

$$R_{\mathrm{d}} = \frac{R_{\mathrm{m}}}{\gamma_{\mathrm{R}}}, \quad \gamma_{\mathrm{R}} = \exp(\alpha_{\mathrm{R}} \beta V_{\mathrm{R}}) \tag{25}$$

The mean and characteristic values of resistance are estimated from two separate nonlinear analyses using mean and characteristic values of the input material parameters, respectively.

The method is quite general and a suitable reliability level can be chosen by β if required. Variety of failure models and the sensitivity to a random variation of the material parameters is adequately captured in most cases of practical relevance.

7.4 Model uncertainty

The evaluation of model uncertainty of the resistance based on NLFEA is critical for a robust and reliable design. The model uncertainty can be determined from the comparison of the model predictions to experimental data. In this respect it should be acknowledged that the obtained partial factors of model uncertainty will be valid only for the investigated material model or simulation software. The model uncertainty is usually defined as the ratio:

$$\theta = R_{\text{exp}} / R_{\text{sim}} \tag{26}$$

where R_{exp} and R_{sim} are the resistances obtained from experiment and numerical simulation, respectively. The model uncertainty covers the imperfection of the model (lack of knowledge). The best agreement between a model and an experiment is obtained when θ_w =1.

The experimental resistance is considered as a reference, i.e. true value. Therefore, to investigate pure model uncertainties, it is essential to reduce other effects such as aleatory uncertainties to minimum. Material properties of concrete are typically identified by the concrete compressive strength tested on accompanying concrete samples (e.g. cylinders). Other material parameters (elastic modulus, tensile strength, fracture energy, etc.) are usually determined indirectly by formulas available in codes or should be provided as guidelines for a particular model Furthermore, there is a random variability of material properties within the tested structure. These effects are therefore included in the model uncertainties.

The experimental data base for the calibration of model uncertainty for a particular material model or modelling approach should include results of experiments relevant to the considered resistance model. Parameters of experiments, such as reinforcement, size or concrete strength class, determine the relevant range of validity. Failure mode is often also suggested as a classification parameter. However, it may be useful to include more failure modes in one group, since in general a failure mode identification is not unique and straightforward.

For a database containing *n* samples (experiments) the central moment characteristics of model uncertainty can be estimated for mean μ_θ, standard deviation σ_θ and coefficient of variation V_θ. Considering a log-normal statistical distribution (according to *fib* MC2010) a safety factor for model uncertainty can be obtained as:

$$\gamma_{Rd} = \frac{\exp(\alpha_R \beta \times V_\theta)}{\mu_\theta}, \tag{27}$$

The reliability parameters should be in accordance with those applied for the material safety in equation (22).

A validation of the model uncertainty is by its nature dependent on the used constitutive model, its implementation in a particular software or even on the analyst or engineer himself. The human factor can be eliminated or at least addressed by providing modelling guidelines, training and education. Several investigations of model uncertainty for various nonlinear finite element software have been published recently.

Engen (2017) [28] investigated 38 RC members under monotone loading analyzed by several authors. Rather low partial safety factor for the model uncertainty was proposed based on the failure mode in the range 1.02-1.04.

Castaldo (2018) [8] considered 25 structural members from various literature sources including deep beams, shear panels and walls. The investigated members had statically determined static scheme and were tested up to failure with a monotonic incremental loading process. The tests were reproduced by non-linear analysis adopting 9 different modelling hypotheses distinguishing between the software platform and concrete tensile response. A total number of 225 simulations has been performed, and the resulting value of the model uncertainty partial factor was 1.15.

Castaldo (2020) [9] investigated also the model uncertainty for cyclic loading of 17 shear walls with statically determined scheme. 18 different modelling hypothesis were considered and altogether 306 simulations have been performed. This study proposes a model uncertainty factor γ_{Rd} = 1.35.

Gino (2021) [33] focused on the problem of model uncertainty related to nonlinear analysis of slender RC members, A total number of 40 experiments of concrete columns with slenderness ratio between 15-275 are considered, and the model uncertainty factor γ_{Rd} is proposed in the range 1.15 – 1.19.

Finally the authors also performed a study in Červenka V. (2018b) [20] including 33 RC members, slabs and beams, with failure modes ranging from ductile up to brittle failure modes. The results of this study are summarized in [20].

It should be noted that the model uncertainties parameters in [20] are valid only for the used software (Cervenka et al, 2020) [5] and the constitutive model (Cervenka & Papanikolaou 2008) [15].

5 Programming Manual

This section describes the **programming and scripting interface** of the ATENA Module for Green Concrete Modelling and Design developed as the software result **TM04000013-V2** within the CeSTaR-3 project. The Programming Manual is intended for **advanced users, researchers, and software developers** who require a higher level of control over model generation, parameter definition, and analysis workflows than is available through the graphical user interface alone.

The module leverages the existing **ATENA programming infrastructure**, including XML-based definitions and Python scripting, to enable **parametric modelling, automation of simulation workflows, and systematic studies** of green concrete structures. These capabilities are particularly important when working with green concrete materials, where multiple material variants, curing times, or SCM replacement levels must be evaluated efficiently and consistently, often using data derived from the experimental database **TM04000013-V6**.

The programming interface allows users to:

- define material properties and their time-dependent evolution,
- generate structural geometry and reinforcement layouts parametrically,
- automate loading protocols and analysis sequences,
- perform batch analyses and post-processing of results.

The scripting commands documented in this section are **directly generated by the ATENA preprocessor** during interactive model creation and can be reused or adapted by the user. This ensures consistency between graphical and scripted workflows and reduces the risk of user error when developing complex or repetitive models.

The Programming Manual complements the User Manual and Technical Documentation by demonstrating how the implemented functionality can be **extended, customized, and integrated** into research-oriented or engineering-oriented computational workflows. It supports the long-term usability of the software beyond the scope of individual case studies and facilitates reproducibility and transparency of numerical simulations.

5.1 Programming of Material Models

5.1.1 Concrete Material with Variable Properties – class header

This is the material class allowing the definition of a general material with time dependent material properties. It is derived from CCMaterial class. Any material can be used as a base Material, such as for instance CC3DNonLinCementitious2 for time dependent concrete modelling.

```
#ifndef CC_MATERIAL_WITH_VAR_PROPERTIES_DEFINED
#define CC_MATERIAL_WITH_VAR_PROPERTIES_DEFINED

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> INCLUDE FILES <<<<<<<<<<<<<<<<<<<<<<<<<<<<

#include "CCUtils\iostream.h"
#include "CCMaterial\CCMaterial.h"
#include "CCFEModel\CCFEModel.h"

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> PREPROCESSOR'S CONSTANTS <<<<<<<<<<<<<<<<<<

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> CLASS TYPE DEFINITIONS <<<<<<<<<<<<<<<<<<<<

/**************************************************************************
 *                                                                        *
 *  Class: CCMaterialWithVariableProperties                               *
 *                                                                        *
 *  Purpose: This class can be used to create a material, whose selected *
 *         material parameters will change according to a specified      *
 *         function depending on time or step number.                    *
 *                                                                        *
 *  Usage: This material is derived from the base material class.         *
 *                                                                        *
 **************************************************************************/

class CCMaterials_Basic_DECL_SPEC CCMaterialWithVariableProperties : public CCMaterial
{
public:
```

```
   enum VersionId {VERSION_ID=3};
   DECLARE_SERIAL(CCMaterialWithVariableProperties)

   CCMaterialWithVariableProperties();

   CCMaterialWithVariableProperties( const CCObject &src );
   CCMaterialWithVariableProperties( const CCMaterialWithVariableProperties &mat );

public:
  /*
   * Destructor
   */
   virtual ~CCMaterialWithVariableProperties ();

  /*
   * Other
   */

   virtual CCObject& operator = (const CCObject& src);

   virtual void write() const;
   virtual CCInt read();
   virtual void delete_contents();

   // Set of virtual function for changing material parameters
   // Returns CCTRUE if parameter was changed. Otherwise CCFALSE
   virtual CCBool change_parameter( CCString &name, CCReal&   value );
   virtual CCBool change_parameter( CCString &name, CCInt&    value );
   virtual CCBool change_parameter( CCString &name, CCString&  value );

   // Creates an appropriate material state, initializes it and
   // returns its pointer
   // !!! User must explicitly delete it !!!
   virtual CCMaterState* create_state() const;

   // Resets material state to initial values
   virtual void reset_state( CCMaterState &mstate ) const;

   // Compute tangent matrix
   virtual void tangent_stiff( CCMaterPoint &mp,
                               CCRealMatrix &d );
   // Compute secant matrix
   virtual void secant_stiff( CCMaterPoint &mp,
                              CCRealMatrix &d );
   // Compute tangent matrix
   virtual void tangent_compl( CCMaterPoint &mp,
                               CCRealMatrix &c );
   // Compute secant matrix
   virtual void secant_compl( CCMaterPoint &mp,
                              CCRealMatrix &c );
   // Update of state variables based on strain increment
   // current state (state is updated if requested)
   virtual const CCReal calculate_response( const CCRealColumnVector &deps,
                                            UpdateFlag update,
                                            CCMaterPoint &mp,
                                            CCRealColumnVector &sigma );
   // Transform state variables to account for large
   // deformations. For instance 2nd Piola Kirchhof stresses
   // to Cauchy stresses
   // (called typically after UPDATE_MATERIAL)
   virtual void transform_state( const CCRealMatrix& deformation_gradient,
                                 CCMaterPoint &mp ) const;

   // Compute elastic matrix
   virtual void calculate_elast_stiff();
   virtual void elast_stiff( CCMaterPoint &mp, CCRealMatrix &d );

   // Compute elastic matrix
   virtual void calculate_elast_compl();
   virtual void elast_compl( CCMaterPoint &mp, CCRealMatrix &d );

   // This method is called to register output data available
   // for this material law.
   virtual void register_output_data( LPCSTR label_extension = NULL ) const;

   // Get material weight density
   virtual CCReal get_weight_density( CCMaterPoint &mp ) const;
   // Get material young modulus density
   virtual CCReal get_young_modulus( CCMaterPoint &mp ) const;
   // Get material mu
   virtual CCReal get_mu( CCMaterPoint &mp ) const;
   // Get material expansion coeff
   virtual CCReal get_expansion_coeff( CCMaterPoint &mp ) const;
   // Get material idealisation type
   virtual CCInt get_material_ideal_type() const;
   // Get material fc (compressiv                                    e strength)
   virtual CCReal get_fc(CCMaterPoint &mp) const;
   // Get material ft (tensile strength)
   virtual CCReal get_ft(CCMaterPoint &mp) const;
   // Get material Gf (fracture energy)
   virtual CCReal get_Gf(CCMaterPoint &mp) const;
   // Get material damping alpha coeff
   virtual CCReal get_damping_alpha_coeff() const;
   // Get material damping_beta coeff
   virtual CCReal get_damping_beta_coeff() const;

   virtual void check( CCFEModel* model_ptr  );

   // dynamic type creation

   CCStatic CCObject* copy_constructor(const CCObject& src) ;
   virtual CopyConstructor get_copy_constructor_ptr() ;

// Info helpers
   virtual CCString description() const  ;
   virtual const CCInt type_of_items() const ;
   virtual CCString item_label(CCInt item) const ;
   virtual const CCInt item_type(CCInt item_id) const;
   virtual const CCInt data_format() const ;
   virtual const CCString item_units(CCInt item) const ;

   virtual void* item_ptr(CCInt item) const;
   virtual CCInt number_of_items() const;

// MFC stuff
   virtual void Serialize( CArchive& archive ); // override
```

```
#ifdef _DEBUG  // Functions for _DEBUG version only
   virtual void AssertValid() const; // override
   virtual void Dump( CDumpContext& dc ) const; // override
#endif

public:
   // Virtual function for setting material parameters
   // Returns true if parameter was successfully set
   virtual CCBool set_parameter( CCString &name, CCReal& value );
   virtual CCBool get_parameter(CCString &name, CCReal &value);
   CCReal get_value(CCInt ipar, CCFEModel *model_ptr=nullptr, CCMaterPoint *mp_ptr=nullptr) const;

   // Data members

   // The id of the base material, whose material properties can
   // be varied. This material utilizes the change_parameter
   // function of the base material, so it is important that
   // the base material is capable of changing its parameters.
   CCMaterialSmartId base_material_id;

   // Heap of material properties names, which will be varied from
   // step to step based on time or step number.
   CStringHeap param_names;

   // Parameter for LITS
   CCReal parameter_a;

   // Heap of function ids that define the evolution of each
   // material parameter, which is specified in the list of
   // param_names
   CCFunctionSmartIdHeap function_t_ids;
   CCFunctionSmartIdHeap function_temp_ids;
   CCRealHeap const_values;   // the final mater. parameter = const_values * function_ids(t)

   CCInt tmp_elem_total_temperature_data_id=-1; // temporary, used only in value(...)
   CCInt tmp_elem_temperature_incr_data_id=-1; // temporary, used only in value(...)
};

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> INLINE FUNCTIONS <<<<<<<<<<<<<<<<<<<<<<<<<<

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> EXTERN VARIABLES AND FUNCTION <<<<<<<<<<<<

// >>>>>>>>>>>>>>>>>>>>>>>>>>>> END OF FILE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#endif // CC_MATERIAL_WITH_VAR_PROPERTIES_DEFINED
```

5.1.2 Variable Material Calculate Response Method

This is the base method controlling the evaluation of stresses based on given strain increment and a set of time dependent functions for the selected material parameters.

```
/**************************************************************************
 *                                                                        *
 *                                                                        *
 * Function: update material point state based on the increment of strain*
 *          and return material response in sigma. Depending on the      *
 *        value of variable update material point state variables.    *                                                                  *
 *                                                                        *
 * Return: returns performance index, which in this case is zero since   *
 *         function does nothing.                                         *
 *                                                                        *
 **************************************************************************/
const CCReal
CCMaterialWithVariableProperties::calculate_response
(
  /*
   * Arguments
   */
   const CCRealColumnVector &deps,   // strain increment
   CCMaterial::UpdateFlag upd_flag,  // CCTRUE is state data are to be
                                     // updated
   CCMaterPoint &mp,                 // Material point with state data
   CCRealColumnVector &sigma         // Calculated stress
)
{
   CCFEModel* model_ptr = FE_Model_Ptr.operator ->();
   CCInt n_param = param_names.GetSize();
   CCMaterial* mater_ptr = model_ptr->get_material_ptr( base_material_id );

   for( CCInt ipar=1; ipar<=n_param; ipar++ )
      {
      CCReal value = get_value(ipar, model_ptr, &mp);
      mater_ptr->change_parameter( param_names( ipar ), value );
      }

   return mater_ptr->calculate_response( deps, upd_flag, mp, sigma );
}
```

5.1.3 Reinforcement Cycling Bond Model

Cycling reinforcement bond model is simulated using an appropriate class derived from CCExternalCable_2D class for FE geometry.

```
// Calculates element nodal slips for a cable element. This routine is
// called for all elements (of the group) from the first element's calculate_element() routine.
void CCBarWithMemoryBond_2D::calculate_slips
   (
   CCElementGroup *element_group_ptr, // [in] current element group pointer
   CCGlobalVector &global_vector,     // [in] displs. within step (for calc. strain)
   CCExternalCableGeometry *geometry_ptr, // [in] ptr. to cable geometry description
   CCMaterial *default_material_ptr,         // [in] pointer to material
   CCFEModel *model_ptr_,               // [in] pointer to CCFEModel
   CCRealMObject*& deviators_data_ptr,    // [in,out] total and delta slip in deviators
   CCBool reset_dslips_at_end,         // [in] flag for reset delta slips (at 1st iteration)
   CCRealMObject &r_coord,             // [tmp] r coords in IPs
   CCRealMObject &dh_dr_value,         // [tmp] diff(nodal functions, r)
   CCRealMObject &dh_dx_value,         // [tmp] diff(nodal functions, x)
```

```
   CCRealMObject &coords_transf,      // [tmp] not used (no transformation global->local coord. system)
   CCRealMObject &dofs_transf,        // [tmp] not used (no transformation global->local coord. system)
   CCRealMObject &element_coords,     // [tmp] local element coordinates
   CCRealMObject &b_l,                // [tmp] B_l matrix
   CCRealMObject &deformation_gradient, // [tmp] matrix of def. gradient
   CCRealCVObject &elem_displs,       // [tmp] element displacements
   CCRealCVObject &nodal_vector,      // [tmp] temporary array (displs. in a node)
   CCRealCVObject &tau,               // [tmp] stress increment
   CCRealCVObject &eps,               // [tmp] strain increment
   CCRealCVObject &deps_ini,          // [tmp] initial strain increment
   CCRealHeap &init_dsig_deps,        // [tmp] temporary storage for initial strain
   CCReal lambda_prev                 // [in] current loading factor lambda (not used here, needed for external cable only)
   )
   {
   CCStructures *model_ptr = dynamic_cast<CCStructures *>(model_ptr_);
   ASSERT(model_ptr);

   // Deal with backward compatibility
   if(is_equal(geometry_ptr->bond_cohesion, -CCREAL_HUGE))
      {
      // ensure compatibility with CCExternalCableGeometry::VERSION_ID<=4
      geometry_ptr->bond_cohesion = geometry_ptr->deviator_friction_constant;
      geometry_ptr->deviator_friction_constant = 0.;
      }
   if(is_equal(geometry_ptr->slip_coeff_unload, CCREAL_HUGE))
      {
      // ensure compatibility with CCExternalCableGeometry::VERSION_ID<=5
      geometry_ptr->slip_coeff_unload = !is_zero(geometry_ptr->deviator_friction_constant) ?
         geometry_ptr->deviator_friction_coefficient/geometry_ptr->deviator_friction_constant                                :
geometry_ptr->deviator_friction_coefficient;
      geometry_ptr->deviator_friction_coefficient = 0;
      }

   CCReal deviator_friction_coefficient_save =  geometry_ptr->deviator_friction_coefficient;
   CCReal slip_coeff_unload_save = geometry_ptr->slip_coeff_unload;
   CCReal deviator_friction_constant_save = geometry_ptr->deviator_friction_constant;
   CCReal bond_cohesion_save = geometry_ptr->bond_cohesion;
   CCReal wobble_coefficient_save = geometry_ptr->wobble_coefficient;

   if(geometry_ptr->release_cable)
      {
      // temporarily release fix of external cable in pre-stressing phase
      geometry_ptr->deviator_friction_coefficient = 0.;
      geometry_ptr->slip_coeff_unload = 0;
      geometry_ptr->deviator_friction_constant = 0.;
      geometry_ptr->bond_cohesion = 0;
      geometry_ptr->wobble_coefficient = 0;
      }

   // Calculate cable geometry
   CCObjectPtrHeap &group_data = (element_group_ptr->group_data);
   CCRealCVObject *element_lengths_ptr = NULL;
   allocate_object_ptr(group_data, "ExternalCableLengths", element_lengths_index,
      element_lengths_ptr, element_lengths_index);
   ASSERT(element_lengths_ptr);
   CCRealCVObject &element_lengths = *element_lengths_ptr;

   CCRealCVObject *deviator_angles_ptr = NULL;
   allocate_object_ptr(group_data, "ExternalCableAngles", deviator_angles_index,
      deviator_angles_ptr, deviator_angles_index);
   ASSERT(deviator_angles_ptr);
   CCRealCVObject &deviator_angles = *deviator_angles_ptr;

   CCReal deviator_radius_save = geometry_ptr->deviator_radius;
   geometry_ptr->calculate_cable_geometry(element_group_ptr, deviator_angles, element_lengths);


   //   CCBool init_mpoints;
   //   init_mpoints= (model_ptr->step_id == 1 && model_ptr->iteration_id ==1);
   CCFEModel::Switches &switches= model_ptr->switches;

   CCBool fix_the_bar;
   long number_of_elements=element_group_ptr->elements.GetSize();
   long number_of_slips=number_of_elements+1;
   CCString msg;
   CCString element_group_label= " in " + element_group_ptr->label(model_ptr);
   CCFixedSize<CCRealHeap, 3> xx, yy; xx.SetSize(3); yy.SetSize(3);

   if(allocate_resize_object_ptr(group_data,"ExternalCableData", deviators_data_index,
      number_of_slips, MINIMUM_SLIP, deviators_data_ptr, deviators_data_index))
      deviators_data_ptr->zero();
   ASSERT(deviators_data_ptr);
   CCRealMObject &deviators_data=*deviators_data_ptr;

   CCRealCVObject *slip_diagonal_ptr=NULL, *slip_super_diagonal_ptr=NULL,
      *slip_sub_diagonal_ptr=NULL, *slips_ptr=NULL;

   CCObjectPtrHeap &group_tmp_heap = CCElementGroup::get_threads_group_tmp_heap().get_container();

   get_object_ptr(group_tmp_heap,slip_diagonal_index, slip_diagonal_ptr);
   get_object_ptr(group_tmp_heap,slip_super_diagonal_index, slip_super_diagonal_ptr);
   get_object_ptr(group_tmp_heap,slip_sub_diagonal_index,slip_sub_diagonal_ptr);
   get_object_ptr(group_tmp_heap,slips_index,slips_ptr); // rhs or delta slips increment

   CCReal x1=0, y1=0;  // coords of the first node of the cable (in local coordinate system)
   CCReal xn=0, yn=0;  // coords of the current node of the cable, for which we calculate slip (in local coordinate system)
   // ccassert allocations
   ASSERT(slip_diagonal_ptr->length() == number_of_slips);
   ASSERT(slip_super_diagonal_ptr->length() == number_of_slips);
   ASSERT(slip_sub_diagonal_ptr->length() == number_of_slips);
   ASSERT(slips_ptr->length() == number_of_slips);
   ASSERT(deviators_data.columns() == 4); // slip increment, total slip
   ASSERT(deviators_data.rows() == number_of_slips); // the above data arranged by deviators

   CCExternalCablePrestress *step_related_prestress_ptr = NULL, *group_prestress_ptr = NULL;
   CCReal prestress = 0.;
   CCInt& prestress_incr_load_id = TLS_allocate_cell_CCInt(CCExternalCable_2D___calculate_slips___prestress_incr_load_id);
   CCInt& prestress_tot_load_id = TLS_allocate_cell_CCInt(CCExternalCable_2D___calculate_slips___prestress_tot_load_id);

   CCObjectPtrHeap &element_step_related_data = element_group_ptr->elements(number_of_elements).
      element_step_related_data;    // grabbing load from the last cable element!! even in the case, when the first element in
the cable is prestressed; see actualize_bcs(CCStructures.cpp)


   // Get element pre-stressing (increment !!)
```

```
   if((prestress_incr_load_id   =   element_step_related_data.FindClassF(RUNTIME_CLASS(CCExternalCablePrestress),   -1,   -1,
prestress_incr_load_id))
      > 0) // check if CCExternalCablePrestress is applied at all. if not, return
      {
      step_related_prestress_ptr = // CCExternalCablePrestress::prestress_incr object
         (CCExternalCablePrestress*)*(element_step_related_data(prestress_incr_load_id)); // ptr to step prestress increment
      }

   // Get element pre-stressing (total with respect to the previous step )
   CCInt load_id;
   if(allocate_object_ptr(group_data, "ExternalCablePrestress",
      load_id, group_prestress_ptr, prestress_tot_load_id))  // check if CCExternalCablePrestress is applied at all. if not,
return
      {
      prestress_tot_load_id = load_id; // save load index for the next search
      }
   ASSERT(group_prestress_ptr);

   if(group_prestress_ptr->reset_dslips)
      {
      group_prestress_ptr->prestress_tot += group_prestress_ptr->prestress_incr;
      group_prestress_ptr->prestress_incr = 0.;
      }

   if(step_related_prestress_ptr)
      {   // prestress total specified, increment specified
      group_prestress_ptr->prestress_incr =
         step_related_prestress_ptr->prestress_incr*lambda_prev;
      }
   else
      {    // prestress total specified, increment not specified
           // do nothing
      }

   prestress = group_prestress_ptr->prestress_tot+group_prestress_ptr->prestress_incr;

   CCReal e_a_stiff_left = 0, e_a_stiff_right = 0;

   // Check cable end nodes BCs

   CCUInt fixed_end_geometry_map = 0;
   if(geometry_ptr->fix_the_bar_left)   // BCs based on geometry
      set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_LEFT);
   if(geometry_ptr->fix_the_bar_right)
      set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_RIGHT);

   CCUInt fixed_end_total_load_map = 0;       // BCs based on previous (total) pre-stressing
   if(is_set_bit(group_prestress_ptr->flag, CCExternalCablePrestress::PRESTRESS_RIGHT))
      set_bit(fixed_end_total_load_map, CCExternalCablePrestress::PRESTRESS_RIGHT);   // = FIX_LEFT
   if(is_set_bit(group_prestress_ptr->flag, CCExternalCablePrestress::PRESTRESS_LEFT))
      set_bit(fixed_end_total_load_map, CCExternalCablePrestress::PRESTRESS_LEFT); // = FIX_RIGHT

   if( step_related_prestress_ptr)
      {
      CCUInt fixed_end_incr_load_map = 0;  // BCs based on current pre-stressing
      if(is_set_bit(step_related_prestress_ptr->flag, CCExternalCablePrestress::PRESTRESS_RIGHT))
         set_bit(fixed_end_incr_load_map, CCExternalCablePrestress::PRESTRESS_RIGHT); // = FIX_LEFT
      if(is_set_bit(step_related_prestress_ptr->flag, CCExternalCablePrestress::PRESTRESS_LEFT))
         set_bit(fixed_end_incr_load_map, CCExternalCablePrestress::PRESTRESS_LEFT); // = FIX_RIGHT

      if(fixed_end_incr_load_map==0)
         {
         fixed_end_incr_load_map = fixed_end_geometry_map; // prestress incr.LCs not defined; used those from geometry
         step_related_prestress_ptr->flag = fixed_end_geometry_map;
         }

      if(fixed_end_total_load_map==0)
         {
         fixed_end_total_load_map = fixed_end_incr_load_map; // prestress total (=previous) LCs not defined; used those from
prestress incr.LCs
         group_prestress_ptr->flag = step_related_prestress_ptr->flag;
         }

      if(fixed_end_total_load_map!=fixed_end_incr_load_map)
         throw new CCFEModelExc(THROW_EXC_PLACE, "CCFEModel", IDS_CCFEModelExc_INCOMPAT_PRESTRESS, 1,
            element_group_ptr->label(model_ptr));
      }
   else
      {
      if(fabs(group_prestress_ptr->prestress_tot+group_prestress_ptr->prestress_incr)>1.e-3)
         {
         set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_LEFT);
         set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_RIGHT);
         }
      }

   CCReal slip_displs_error=CCREAL_HUGE, slip_left, slip_right=0.;
   CCInt slip_iteration_id=1;

   CCReal element_force_left, element_force_right;
   CCReal cable_length_right=0, cable_length_left=0., total_cable_length=0.;
   CCReal cohesion_left=0, cohesion_right=0;
   CCReal wobble_cohesion_left = 0, wobble_cohesion_right = 0;
   CCReal variable_cohesion_left, variable_cohesion_right;
   CCReal reduction_factor_left, reduction_factor_right;
   CCReal temperature_left=0, temperature_right=0;

   CCReal fi;
   CCReal deviator_a, deviator_b, deviator_a_left, deviator_b_left,
      deviator_a_right, deviator_b_right;

   if(group_prestress_ptr->reset_dslips)
      {  // zeroize delta slips
      memset( &deviators_data(1,DELTA_SLIP), 0, deviators_data.rows()*sizeof(CCReal));
      group_prestress_ptr->reset_dslips = CCFALSE;
      }

   CCInt mater_prop_id=0;
   CCRealColumnVector *h_value_ptr =  (CCRealCVObject*) get_object_ptr(group_tmp_heap, h_value_index);
   ASSERT(h_value_index);
   CCReal area = geometry_ptr->get_area();

   while(slip_iteration_id <= switches.iteration_limit &&
      slip_displs_error > switches.displacement_error)
      {
      CCReal scoord = 0; // longitudinal s coordinates of prestress bas, (from start to end)
```

```
      fix_the_bar=CCTRUE; // assume the bar need fixing;

      if(is_set_bit(fixed_end_total_load_map, CCExternalCablePrestress::PRESTRESS_LEFT))
         {
         CCElement *element_ptr = &element_group_ptr->elements(1);
         reduction_factor_left = get_reduction_factor(element_ptr, geometry_ptr, group_tmp_heap);
         element_force_left = prestress*area*reduction_factor_left;
         }
      else
         {
         reduction_factor_left = 1;
         element_force_left = 0.;
         }

      cohesion_left=0.;
      wobble_cohesion_left = 0.;
      slip_left=deviators_data(1, DELTA_SLIP);

      temperature_right = model_ptr->base_temperature_ref;

      for(CCInt slip_id = 1; slip_id<= number_of_slips; slip_id++)
         {
         if(slip_id>1)
            scoord += element_lengths(slip_id-1);
         else
            scoord = 0;

         // solving node, i.e. slip_id, i.e. left node of element slement_id=slip_id

         if(slip_id< number_of_slips)
            {
            CCInt element_id = slip_id;

            // element exists, calculate its normal force

            CCElement *element_ptr = &(element_group_ptr->elements(element_id));
            CCMaterPoint *mater_point_ptr = element_ptr->element_state->mpoints(1);

            reduction_factor_right = get_reduction_factor(element_ptr, geometry_ptr, group_tmp_heap);
            CCReal reduced_area = area * reduction_factor_right;

             // Obtain reference to current material at element
            CCMaterial *material_ptr;
            if(get_mater_ip_prop_id(1, element_group_ptr, element_ptr, mater_prop_id))
               material_ptr = model_ptr->get_material_ptr( mater_prop_id );
            else
               material_ptr=default_material_ptr;
            // if(init_mpoints) element_ptr->set_mpoints( element_group_ptr, material_ptr);
            //         if(init_mpoints) create_element_state( element_group_ptr,
            //                                                element_ptr );
            // right element slip
            slip_right= deviators_data(slip_id+1, DELTA_SLIP);
            // calculate right element force
            element_force_right=calculate_cable_force
               ( element_group_ptr, element_ptr, global_vector,
               material_ptr, model_ptr, slip_left, slip_right,
               r_coord, dh_dr_value, dh_dx_value, coords_transf, dofs_transf,
               element_coords, b_l, deformation_gradient,
               elem_displs, nodal_vector, tau, eps, deps_ini,
               init_dsig_deps);

            // right element stiffness
            e_a_stiff_right = reduced_area * material_ptr->get_young_modulus(*mater_point_ptr);
            e_a_stiff_right *= geometry_ptr->reduction_factor(element_ptr, h_value_ptr);

            // calculate cable length
            cable_length_right = element_lengths(slip_id);
            ASSERT(cable_length_right>0);

            if(slip_id==1)
               {
               x1=element_coords(1,1);
               y1=element_coords(2,1);
               }
            xn=element_coords(1,1);
            yn=element_coords(2,1);

            // calculate cohesion
            if(geometry_ptr->bond_cohesion > 0)
               cohesion_right = geometry_ptr->perimeter*0.5*cable_length_right*   // surface reduced_area of the bar
                  geometry_ptr->bond_cohesion;  // cohesion force per unit reduced_area
            else // negative value input directly * (-1.)
               cohesion_right=-geometry_ptr->bond_cohesion;

            wobble_cohesion_right = geometry_ptr->perimeter*0.5*cable_length_right*   // surface reduced_area of the bar
               geometry_ptr->wobble_coefficient*max(0.,  element_force_right)/reduced_area;    //  cohesion  force  per  unit
reduced_area
            }
         else
            {
            CCInt element_id = slip_id-1;

            // element does not exist, the very last slip of the cable
            if(is_set_bit(fixed_end_total_load_map, CCExternalCablePrestress::PRESTRESS_RIGHT))
               {
               CCElement *element_ptr = &(element_group_ptr->elements(element_id));
               reduction_factor_right = get_reduction_factor(element_ptr, geometry_ptr, group_tmp_heap);
               element_force_right = prestress*area*reduction_factor_right;
               }
            else
               {
               element_force_right = 0.;
               reduction_factor_right = 1;
               }

            xn=element_coords(1,2);
            yn=element_coords(2,2);

            cohesion_right=0;
            wobble_cohesion_right = 0;
            }
         variable_cohesion_right=cohesion_right;
         variable_cohesion_left=cohesion_left;

         CCReal slip_cf_load = 1.;
```

```
         CCReal slip_cf_load_max = 1.;
         CCReal slip_cf_load_min = 1.;
         CCReal dcoh_dslip = CCREAL_ZERO;
         CCReal slip_cf_unload = geometry_ptr->slip_coeff_unload;
         CCReal slip_cf;

         CCReal SMOOTH_CF=0.2;  // smoothing between  frict_coeff and frict_cf_unload
         CCReal tot_slip= deviators_data(slip_id,TOTAL_SLIP);
         CCReal max_slip= deviators_data(slip_id,MAXIMUM_SLIP);
         CCReal min_slip= deviators_data(slip_id,MINIMUM_SLIP);
         ASSERT(max_slip+CCREAL_SMALL>=min_slip);
         CCReal delta_slip = max_slip-min_slip;
         CCReal dforce = element_force_right - element_force_left;

         if (geometry_ptr->slip_function_id)
         {
            slip_cf_load     = (**model_ptr->functions(geometry_ptr->slip_function_id))(fabs(tot_slip));
            slip_cf_load_max = (**model_ptr->functions(geometry_ptr->slip_function_id))(fabs(max_slip));
            slip_cf_load_min = (**model_ptr->functions(geometry_ptr->slip_function_id))(fabs(min_slip));
            dcoh_dslip       = (**model_ptr->functions(geometry_ptr->slip_function_id)).df_dx(fabs(tot_slip));
            if (dcoh_dslip < CCREAL_ZERO)
               dcoh_dslip = CCREAL_ZERO;
         }
         slip_cf_unload = min(geometry_ptr->slip_coeff_unload, slip_cf_load);

         if (is_equal(delta_slip, CCREAL_ZERO))
         {
            slip_cf = slip_cf_load;
         }
         else if (is_within_range(tot_slip, max_slip - SMOOTH_CF*delta_slip, max_slip, CCMIN_ABS, delta_slip*0.001))
         {  // transition interval near max_slip,
            // pushing towards max_lip use interpolated value between slip_cf_unload,  slip_cf_load
            if (dforce > CCREAL_ZERO)
            {
               xx(1) = max_slip - SMOOTH_CF*delta_slip; xx(3) = max_slip; xx(2) = (xx(1) + xx(3)) / 2.;
               yy(1) = slip_cf_unload; yy(3) = slip_cf_load_max; yy(2) = (yy(1) + yy(3)) / 2.;
               slip_cf = interpolate_lagrange(tot_slip, xx.GetSize(), xx.GetData(), yy.GetData());
            }
            else // unloading
            {
               slip_cf = slip_cf_unload;
            }
         }
         else if (is_within_range(tot_slip, min_slip, min_slip + SMOOTH_CF*delta_slip, CCMIN_ABS, delta_slip*0.001))
         {  // transition interval near min_slip
            // pushing towards min_slip, use interpolated value between slip_cf_load,  slip_cf_unload
            if (dforce < CCREAL_ZERO)
            {
               xx(1) = min_slip; xx(3) = min_slip + SMOOTH_CF*delta_slip; xx(2) = (xx(1) + xx(3)) / 2.;
               yy(1) = slip_cf_load_min; yy(3) = slip_cf_unload; yy(2) = (yy(1) + yy(3)) / 2.;
               slip_cf = interpolate_lagrange(tot_slip, xx.GetSize(), xx.GetData(), yy.GetData());
            }
            else // unloading
            {
               slip_cf = slip_cf_unload;
            }
         }
         else if (is_within_range(tot_slip, min_slip + SMOOTH_CF*delta_slip, max_slip - SMOOTH_CF*delta_slip, CCMIN_ABS,
delta_slip*0.001))
         { // tot_slip inside of <min_slip+cf*delta_slip...max_slip-cf*delta_slip, use frict_cf_unload
            slip_cf = slip_cf_unload;
         }
         else // in any other case it must be loading
         {
            slip_cf = slip_cf_load;
         }

         CCReal dcoh_dslip_left  = dcoh_dslip * variable_cohesion_left;
         CCReal dcoh_dslip_right = dcoh_dslip * variable_cohesion_right;

         variable_cohesion_right *= slip_cf;
         variable_cohesion_left *= slip_cf;

         variable_cohesion_right += wobble_cohesion_right;
         variable_cohesion_left += wobble_cohesion_left;

         CCReal loc_cf=1;

         if(geometry_ptr->loc_function_id)
            {
            CCReal location=sqrt(pow(xn-x1,2)+pow(yn-y1,2));
            loc_cf = (**model_ptr->functions(geometry_ptr->loc_function_id))(location);
            variable_cohesion_right *= loc_cf;
            variable_cohesion_left  *= loc_cf;
            }

         CCReal prestress_loss = 0;
         if(geometry_ptr->prestress_loss_function_id)
            {
            CCReal element_dforce = element_force_right-element_force_left;
            CCFunction *prestress_loss_function_ptr = model_ptr->functions(geometry_ptr->prestress_loss_function_id);
            CCReal       prestress_loss_cf       =       (*prestress_loss_function_ptr)(scoord+cable_length_right/2.)       -
(*prestress_loss_function_ptr)(scoord-cable_length_left/2.);
            prestress_loss = fabs(prestress_loss_cf)*prestress*area /* *(reduction_factor_left+reduction_factor_right)/2.) */;
            prestress_loss = min(prestress_loss, fabs(element_dforce));
            prestress_loss = -sign_transfer(prestress_loss, element_dforce);
            }

         if(geometry_ptr->temperature_function_id)
            {
            if(slip_id==1)
               temperature_left = model_ptr->base_temperature_ref;
            if(slip_id<number_of_slips)
               {
               CCElement *element_ptr = &(element_group_ptr->elements(slip_id));

               CCReal  aver_humid_at_start_time  =  0,  aver_temp_at_start_time  =  0,  aver_humid_at_target_time  =  0,
aver_temp_at_target_time = 0;
               CCTimeTempHumid::mean_humid_temp_at_element(element_ptr, aver_humid_at_start_time, aver_temp_at_start_time,
                  aver_humid_at_target_time, aver_temp_at_target_time);
               temperature_right = (aver_temp_at_start_time+aver_temp_at_target_time)/2.;
               }
            else
               temperature_right = model_ptr->base_temperature_ref;
```

```
            CCReal                                                  temper_cf                                                                       =
(**model_ptr->functions(geometry_ptr->temperature_function_id))((temperature_left+temperature_right)/2.);
            variable_cohesion_right *= temper_cf;
            variable_cohesion_left *= temper_cf;
            }

         if(geometry_ptr->corrosion_function_id)
            {
            CCReal corr_level = 1. - (reduction_factor_left+reduction_factor_right)/2.;
            CCReal corr_cf = (**model_ptr->functions(geometry_ptr->corrosion_function_id))(corr_level);
            variable_cohesion_right *= corr_cf;
            variable_cohesion_left *= corr_cf;
            }

         if(geometry_ptr->aging_function_id)
            {
            CCReal        constr_time=        model_ptr->t        -        element_group_ptr->group_construct_time        -
element_group_ptr->elements(slip_id).elem_construct_time;
            CCReal age_cf = (**model_ptr->functions(geometry_ptr->aging_function_id))(constr_time);
            variable_cohesion_right *= age_cf;
            variable_cohesion_left *= age_cf;
            }

         variable_cohesion_right=max(variable_cohesion_right,CCREAL_SMALL);  // there must be at least some cohesion there
         variable_cohesion_left=max(variable_cohesion_left, CCREAL_SMALL);

         if(slip_iteration_id==1) total_cable_length += cable_length_right;

         fi = deviator_angles(slip_id);
         /*
         Testing - for suppressing action of deviators set

         deviator_a = 1;
         deviator_b = 0;

         which produces

         deviator_a_left = deviator_a_right = 1;
         deviator_b_left = deviator_b_right = 0;
         */


         if(geometry_ptr->deviator_friction_coefficient > -CCREAL_SMALL) // correct calculation
            deviator_a = exp(-fi*geometry_ptr->deviator_friction_coefficient*geometry_ptr->perimeter /* *slip_cf */ *loc_cf);
         else   // negative value input directly * (-1.)
            deviator_a = -geometry_ptr->deviator_friction_coefficient /* *slip_cf */ *loc_cf;

         if(geometry_ptr->deviator_friction_constant > -CCREAL_SMALL)
            deviator_b = geometry_ptr->perimeter*fi*geometry_ptr->deviator_friction_constant*geometry_ptr->deviator_radius /*
*slip_cf */ *loc_cf;
         else // negative value input directly * (-1.)
            deviator_b = -geometry_ptr->deviator_friction_constant /* *slip_cf */ *loc_cf;

         if(element_force_right >= element_force_left)
            { // use F_right
            deviator_a_left = 1.;
            deviator_b_left = 0.;
            deviator_a_right = deviator_a;
            deviator_b_right = deviator_b;
            }
         else
            { // use F_left
            deviator_a_left = deviator_a;
            deviator_b_left = deviator_b;
            deviator_a_right = 1.;
            deviator_b_right = 0.;
            }

         // CCReal tensile_element_force_left = max(0, element_force_left);
         // CCReal tensile_element_force_right = max(0, element_force_right);
         CCReal tensile_element_force_left  = element_force_left;
         CCReal tensile_element_force_right = element_force_right;

         CCReal bond_strength = variable_cohesion_left + variable_cohesion_right
                              + (1. - deviator_a_left)* tensile_element_force_left + deviator_b_left
                              + (1. - deviator_a_right)*tensile_element_force_right + deviator_b_right;

         //CCReal d_slip = deviators_data(slip_id, DELTA_SLIP);

         if  ((fabs(element_force_right  -  element_force_left+prestress_loss)  >  bond_strength)  /*  ||  (  !is_equal(d_slip,
CCREAL_ZERO)) */)
            { // slip activated, should be used also if the slip is activated in the current step, in this case
              // bond stress must be exactly on the slip law curve

            if(slip_id == 1)
               { // the very first slip

               (*slip_diagonal_ptr)(slip_id) = e_a_stiff_right/cable_length_right + dcoh_dslip_right;  // diagonal
               (*slip_sub_diagonal_ptr)(slip_id) = 0.;  // sub_diagonal
               (*slip_super_diagonal_ptr)(slip_id) = -e_a_stiff_right/cable_length_right; // super_diagonal

               (*slips_ptr)(slip_id) = element_force_right-element_force_left+prestress_loss-
                  sign_transfer(variable_cohesion_right+
                  (1-deviator_a_right)*tensile_element_force_right+deviator_b_right,
                     tensile_element_force_right-tensile_element_force_left);  // rhs
               }
            else if(slip_id == number_of_slips)
               { // the very last slip
               (*slip_diagonal_ptr)(slip_id) = e_a_stiff_left/cable_length_left + dcoh_dslip_left;  // diagonal
               (*slip_sub_diagonal_ptr)(slip_id) = -e_a_stiff_left/cable_length_left;  // sub_diagonal
               (*slip_super_diagonal_ptr)(slip_id) = 0.; // super_diagonal

               (*slips_ptr)(slip_id) = element_force_right-element_force_left+prestress_loss-
                  sign_transfer(variable_cohesion_left+
                  (1.-deviator_a_left)*tensile_element_force_left+deviator_b_left,
                     tensile_element_force_right-tensile_element_force_left);  // rhs
               }
            else
               { // intermediate slips

               (*slip_diagonal_ptr)(slip_id)= e_a_stiff_left/cable_length_left
                  + e_a_stiff_right/cable_length_right
                  + dcoh_dslip_left + dcoh_dslip_right;  // diagonal
               (*slip_sub_diagonal_ptr)(slip_id)=
                  -e_a_stiff_left/cable_length_left;  // sub_diagonal
               (*slip_super_diagonal_ptr)(slip_id)=
```

```
                  -e_a_stiff_right/cable_length_right; // super_diagonal
               (*slips_ptr)(slip_id) = element_force_right - element_force_left+prestress_loss -
                  sign_transfer(bond_strength, tensile_element_force_right-tensile_element_force_left);  // rhs
               }
            }
         else
            { // slip not activated, ie. it will not change

            if(slip_id == 1)
               { // the very first slip
               (*slip_diagonal_ptr)(slip_id)= e_a_stiff_right/cable_length_right; // any nonzero number
               }
            else if(slip_id == number_of_slips)
               { // the very last slip
               (*slip_diagonal_ptr)(slip_id)= e_a_stiff_left/cable_length_left;  // any nonzero number
               }
            else
               { // intermediate slips
               (*slip_diagonal_ptr)(slip_id)= e_a_stiff_left/cable_length_left+
                  e_a_stiff_right/cable_length_right; // any nonzero number
               }
            (*slip_super_diagonal_ptr)(slip_id)= 0.;
            (*slip_sub_diagonal_ptr)(slip_id)= 0.;
            (*slips_ptr)(slip_id) = 0.;
            fix_the_bar=CCFALSE;
            }

         // copy some vital data for the next element
         cable_length_left=cable_length_right;
         element_force_left=element_force_right;
         cohesion_left=cohesion_right;
         wobble_cohesion_left = wobble_cohesion_right;
         reduction_factor_left = reduction_factor_right;
         temperature_left = temperature_right;
         e_a_stiff_left=e_a_stiff_right;
         slip_left= slip_right;
         }

      // solve the system for new slip increments

      if(is_set_bit(fixed_end_geometry_map,    CCExternalCablePrestress::FIX_LEFT)    &&    is_set_bit(fixed_end_geometry_map,
CCExternalCablePrestress::FIX_RIGHT))
         {  // fixing of the left and right end points required by user
         (*slips_ptr)(1)=0.;
         (*slips_ptr)(number_of_slips)=0.;
         long number_of_eqns=number_of_elements-1;
         if(number_of_eqns>0)
            MY_DLSLTR(&number_of_eqns, &((*slip_sub_diagonal_ptr)(2)),   // number_of_elements (=number_of_slips-1) dofs
            &((*slip_diagonal_ptr)(2)), &((*slip_super_diagonal_ptr)(2)),
            &((*slips_ptr)(2)));
         }
      else if(is_set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_LEFT))
         { // fixing of the left end points required by user
         (*slips_ptr)(1)=0.;
         if(number_of_elements>0)
            MY_DLSLTR(&number_of_elements, &((*slip_sub_diagonal_ptr)(2)),   // number_of_elements (=number_of_slips-1) dofs
            &((*slip_diagonal_ptr)(2)), &((*slip_super_diagonal_ptr)(2)),
            &((*slips_ptr)(2)));
         }
      else if(is_set_bit(fixed_end_geometry_map, CCExternalCablePrestress::FIX_RIGHT))
         {  // fixing of the right end points required by user
         (*slips_ptr)(number_of_slips)=0.;
         if(number_of_elements>0)
            MY_DLSLTR(&number_of_elements, &((*slip_sub_diagonal_ptr)(1)),   // number_of_elements (=number_of_slips-1) dofs
            &((*slip_diagonal_ptr)(1)), &((*slip_super_diagonal_ptr)(1)),
            &((*slips_ptr)(1)));
         }
#ifdef FIX_1ST_SLIP_IF_ALL_NODES_SLIP
      else if(fix_the_bar)
         {  // all segments of the bar are slipping, fix the 1st. slip incr. (within the iteration) in order to avoid singularity
            // (much the same as for geometry_ptr->fix_the_bar_left)
         (*slips_ptr)(1)=0.;
         if(number_of_elements>0)
            MY_DLSLTR(&number_of_elements, &((*slip_sub_diagonal_ptr)(2)),   // number_of_elements (=number_of_slips-1) dofs
            &((*slip_diagonal_ptr)(2)), &((*slip_super_diagonal_ptr)(2)),
            &((*slips_ptr)(2)));
         }
#endif
      else
         {  // normal processing, the bar somewhere fixed due to not activated slip
         if(number_of_slips>0)
            MY_DLSLTR(&number_of_slips, &((*slip_sub_diagonal_ptr)(1)),   // number_of_slips dofs
            &((*slip_diagonal_ptr)(1)), &((*slip_super_diagonal_ptr)(1)),
            &((*slips_ptr)(1)));
         }

      // solve the system for new slip increments
      slip_displs_error = pow(dot_product(
         (const CCRealColumnVector&) *slips_ptr,
         (const CCRealColumnVector&) *slips_ptr),0.5)
         * geometry_ptr->precision_factor / total_cable_length;


      for(CCInt slip_id = 1; slip_id <= number_of_slips; slip_id++)
         {
         deviators_data(slip_id,TOTAL_SLIP) += geometry_ptr->damping_factor *
            (*slips_ptr)(slip_id); // update deviator slips
         deviators_data(slip_id,DELTA_SLIP) += geometry_ptr->damping_factor *
            (*slips_ptr)(slip_id); // update deviator slips
         if(deviators_data(slip_id,MAXIMUM_SLIP)<deviators_data(slip_id,TOTAL_SLIP))
            deviators_data(slip_id,MAXIMUM_SLIP) = deviators_data(slip_id,TOTAL_SLIP);
         if(deviators_data(slip_id,MINIMUM_SLIP)>deviators_data(slip_id,TOTAL_SLIP))
            deviators_data(slip_id,MINIMUM_SLIP) = deviators_data(slip_id,TOTAL_SLIP);
         }

      slip_iteration_id++;

//       afxDump << "\n step_id: " << model_ptr->step_id << " iteration_id: " << model_ptr->iteration_id << " slip_iteration_id:
" << slip_iteration_id << "\n";
//       dRM(deviators_data);
      }

   if(slip_iteration_id > switches.iteration_limit)
      {
      //       message(MSG_EXC_PLACE, "CCStructures", IDS_CCStructuresWar_FAIL_CONV, 0); // divergence
      msg.Format("iteration %5i, error %10.2g", slip_iteration_id, slip_displs_error);
```

```
   message(MSG_PLACE_HIDDEN /*MSG_EXC_PLACE*/, "CCStructures", IDS_CCStructuresWar_EXTERNAL_CABLE_SLIPS_HDR, 2, msg,
      element_group_label);
   if(slip_iteration_id == 1) element_group_label = ""; // delete cable id for the subsequent iterations
   }

if(reset_dslips_at_end)
   group_prestress_ptr->reset_dslips=CCTRUE;

geometry_ptr->deviator_radius = deviator_radius_save;
if(geometry_ptr->release_cable)
   {
   // restore after temporary release fix of external cable in pre-stressing phase
   geometry_ptr->deviator_friction_coefficient = deviator_friction_coefficient_save;
   geometry_ptr->slip_coeff_unload = slip_coeff_unload_save;
   geometry_ptr->deviator_friction_constant = deviator_friction_constant_save;
   geometry_ptr->bond_cohesion = bond_cohesion_save;
   geometry_ptr->wobble_coefficient = wobble_coefficient_save;
   }
}
```

5.2 Setting Up the Environment

Before scripting, ensure:

- ATENA 2026 Python interface is installed
- The atena2026 Python package is available (via ATENA installer)
- Your working environment is configured (e.g., VSCode, PyCharm, or a Jupyter notebook)

5.3 XML part

This method is used to create new material types into the ATENA material library. It consists of the following key elements and features:

- User-definable material library
- Material is defined in an Xml file
  - UI controls (labels, text box, check box, etc.) are defined directly using Xml
  - UI controls functionality is defined in Python
  - Output to the INP file using Python
- Types of Materials:
  - Type – concrete, reinforcement, interface, steel,...
  - Prototype – CC3DNonLinCementitious2, CC3DNonLinCementitious2FRC,...
  - Analysis type – Static, Dynamic, Creep and Transport

```
<ConcreteMaterial Prototype="CC3DNonLinCementitious2"
        AnalysisType="Static, Dynamic, Creep" >
```

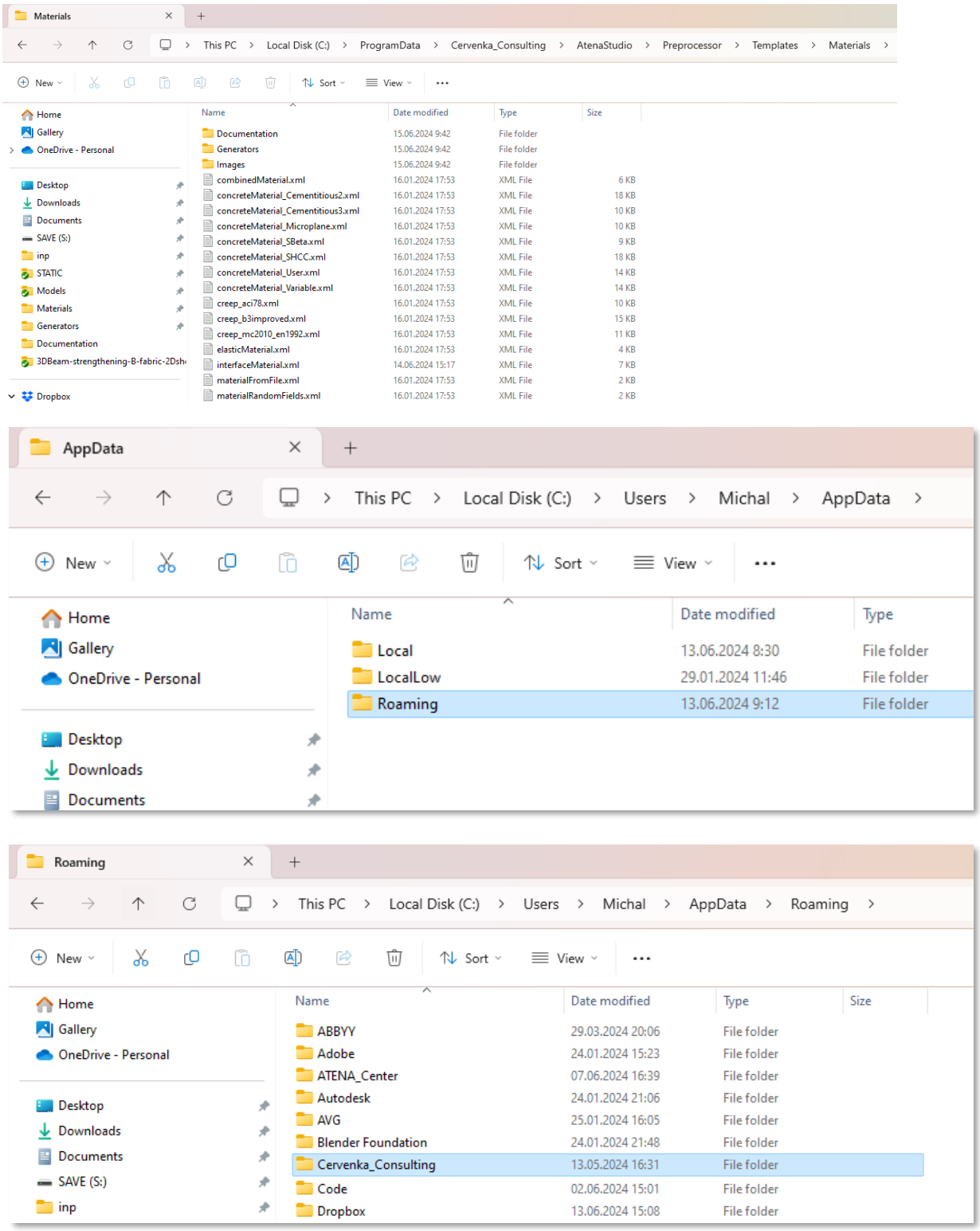
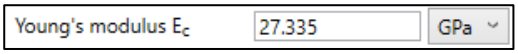


Fig. 27: Location of material library scripts on the computer where ATENA is installed.

The following figures define the key input fields in the ATENA material scripts that can be used for the definition of the new materials inside ATENA software.

- Label with text box for real number



- Real parameter with "Key" attribute (frequently used)

```
<RealParameter TabSection="Basic" Key="Young's modulus" />
```

resourceMaterial.xml defines properties of "Key" attribute

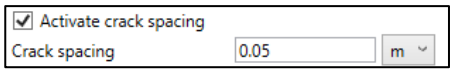
```
<RealParameter Key="Young's modulus" Label="Young's modulus E_c"
 ValueInUnit="27" Quantity="STRESS" Unit="GPa" RangeUnit="MPa"
 StandardRange="(0.005; 5e+5)" LimitRange="(0; 1e+20)" />
```

- Real parameter without „Key" attribute

```
<RealParameter TabSection="Compressive" Name="Plastic potential parameter A"
 Value="5.436344" StandardRange="[-1e6; 1e6]" LimitRange="[-1e9; 1e9]"
 ToolTip="First coefficient of the plastic potential function" />
```

Fig. 28: Definition of real number input field in material dialogs.

- Label with check box, values True / False



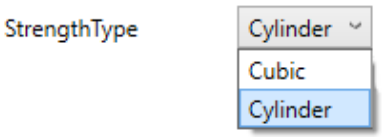
- It is used to activate / deactivate another parameter

```
<BoolParameter TabSection="Tensile" Key="Activate crack spacing„
 OnChange="onChange_Activate" />
<RealParameter TabSection="Tensile" Key="Crack spacing" />
```

- Function „OnChange" is defined in Python

```
def onChange_Activate():
   Activate_Crack_Spacing = materialProxy.Parameters("Activate crack spacing").GetValue()
   if Activate_Crack_Spacing == 1:
      materialProxy.Parameters("Crack spacing").SetVisibility("normal")
   else:
      materialProxy.Parameters("Crack spacing").SetVisibility("hide")
```

Fig. 29: Definition of boolean input field in material dialogs.

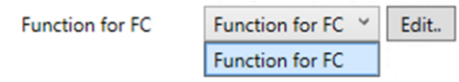


- Label with combo box
- It is used to select multiple options

```
<EnumParameter TabSection="Basic" Name="StrengthType" Value="Cylinder„
 ValueRange="Cubic, Cylinder" />
```

Fig. 30: Definition of combo box input field in material dialogs.

- Label with combo box with functions




```
<FunctionParameter TabSection="Temp dep params" Name="Function for FC" >
  <HorizontalSeries Name="T" Unit="°C" MustBeMonotone="True" >
    20; 100; 300; 400; 800; 900; 1000; 1100; 1200
  </HorizontalSeries>
  <VerticalSeries Name="Fc">
    1; 0.95; 0.85; 0.75; 0.15; 0.08; 0.04; 0.01; 0.001
  </VerticalSeries>
</FunctionParameter>
```

Fig. 31: Definition of combo box input field for material user functions in material dialogs.

- Parameter which invokes subgenerator dialog
- Can be used only inside the material


```
<GeneratorParameter TabSection="Temp dep params" Name="Generate
parameters" GeneratorSubName="ConcreteTempDep" />
```

Fig. 32: Definition of combo box input field for material user functions in material dialogs.

Tab. 2: Source of the new developed CeSTaR-3 material scripts for the new materials with time dependent material functions.

```
<?xml version="1.0" encoding="utf-8" ?>
<Materials>
  <ConcreteMaterial Prototype="Cementitious2_CeSTaR3" AnalysisType="Static, Dynamic, Creep" >
    <!-- Basic -->
    <EnumParameter TabSection="Basic" Name="Material_Prototype" Label="Material prototype"
Value="CC3DNonLinCementitious2CeSTaR3" ValueRange="CC3DNonLinCementitious2CeSTaR3"
Visibility="read" />
    <RealParameter TabSection="Basic" Key="Poisson's ratio" />
    <FunctionParameter TabSection="Basic" Name="Function for E" Label="Function for E_c"
ToolTip="Elastic modulus. Acceptable range: (0; maximal real number>, In Static analysis Time
= Step." >
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="E" Unit="MPa" />
    </FunctionParameter>
    <FunctionParameter TabSection="Basic" Name="Function for Ft" Label="Function for F_t"
ToolTip="Tensile strength Format: FT  x  Units: F/(l^2) Acceptable range: (0; maximal real
number> Default value: 3 , In Static analysis Time = Step." >
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="Tension_Strength-Ft" Unit="MPa" />
    </FunctionParameter>
    <FunctionParameter TabSection="Basic" Name="Function for Fc" Label="Function for F_c"
Image="cc2_variable_fc.gif" ToolTip="Compressive strength Format:   FC  x  Units: F/(l^2)
Acceptable range: minimal real number; 0. Default value: -30, In Static analysis Time =
Step." >
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" >
        0; 1; 3; 7; 14; 28; 60; 90; 180; 360
      </HorizontalSeries>
      <VerticalSeries Name="Compressive_Strength_Fc" Unit="MPa" >
        0; 11.3; 22.4; 31.3; 36.7; 40.2; 42.4; 43.1; 43.8; 44.1
      </VerticalSeries>
    </FunctionParameter>
    <!-- Tensile -->
    <FunctionParameter TabSection="Tensile" Name="Function for Gf" Label="Function for G_f"
Image="cc2_variable_tensile.gif" ToolTip="Specific fracture energy Units: F/l Acceptable
range: 0; maximal real number, In Static analysis Time = Step." >
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="Fracture energy Gf" Unit="MN/m" />
    </FunctionParameter>
```

```
    <RealParameter TabSection="Tensile" Key="Fixed crack" />
    <RealParameter TabSection="Tensile" Key="Radial cracks" />
    <BoolParameter TabSection="Tensile" Key="Activate crack spacing"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Tensile" Key="Crack spacing" />
    <BoolParameter TabSection="Tensile" Key="Activate crack spacing min"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Tensile" Key="Crack spacing min" />
    <BoolParameter TabSection="Tensile" Key="Activate tension stiffening"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Tensile" Key="Tension stiffening" />
    <BoolParameter TabSection="Tensile" Key="Activate aggregate interlock"
OnChange="onChange_Activate" Value="1" />
    <RealParameter TabSection="Tensile" Key="Agg size" />
    <BoolParameter TabSection="Tensile" Key="Activate shear factor"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Tensile" Key="Shear factor" />
    <BoolParameter TabSection="Tensile" Key="Activate unloading factor"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Tensile" Key="Unloading factor" />
    <!-- Compressive -->
    <RealParameter TabSection="Compressive" Key="Plastic strain EPS CP" />
    <BoolParameter TabSection="Compressive" Name="Activate function for EPS CP"
Label="Activate function for ϵ_cp" Value="1" OnChange="onChange_Activate" />
    <FunctionParameter TabSection="Compressive" Name="Function for EPS CP" Label="Function for
ϵ_cp" ToolTip="Plastic strain at compressive strength. Units: none Acceptable range: minimal
real number;0 Default value: -0.001 Generation formula: FC/E , In Static analysis Time =
Step.">
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="Plastic strain EPS CP" />
    </FunctionParameter>
    <RealParameter TabSection="Compressive" Key="Critical comp disp Wd" />
    <BoolParameter TabSection="Compressive" Name="Activate function for Wd" Label="Activate
function for w_d" Value="0" OnChange="onChange_Activate" />
    <FunctionParameter TabSection="Compressive" Name="Function for Wd" Label="Function for
w_d" Image="compressive.gif" ToolTip="Critical compressive displacement Units: l Acceptable
range: 0; maximal real number Default value: -0.0005 , In Static analysis Time = Step.">
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="Critical comp disp WD" Unit="m" />
    </FunctionParameter>
    <BoolParameter TabSection="Compressive" Key="Activate crush band min"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Compressive" Key="Crush band min" />
    <!-- Miscellaneous -->
    <RealParameter TabSection="Miscellaneous" Key="Excentricity EXC" />
    <FunctionParameter TabSection="Miscellaneous" Name="Function for FcO" Label="Function for
F_cO" ToolTip="Onset of non-linear plasticity behavior in compression. Units: F/(l^2)
Acceptable range: minimal real number,-FT*2 Default value: -20  Generation formula: FT*2.1 \n
In Static analysis Time = Step.">
      <HorizontalSeries Name="Time" Unit="day" MustBeMonotone="True" />
      <VerticalSeries Name="Onset of crushing FcO" Unit="MPa" />
    </FunctionParameter>
    <RealParameter TabSection="Miscellaneous" Key="Dir of pl flow BETA" />
    <RealParameter TabSection="Miscellaneous" Key="Rho density" />
    <RealParameter TabSection="Miscellaneous" Key="Thermal expansion" />
    <BoolParameter TabSection="Miscellaneous" Key="Activate substepping"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Miscellaneous" Key="Substeps per Ft" />
    <RealParameter TabSection="Miscellaneous" Key="Max substeps" />
    <BoolParameter TabSection="Miscellaneous" Key="Limit tau crack" />
    <BoolParameter TabSection="Miscellaneous" Key="Activate particular damping"
OnChange="onChange_Activate" />
    <RealParameter TabSection="Miscellaneous" Key="Damping mass coeff" />
    <RealParameter TabSection="Miscellaneous" Key="Damping stiffnes coeff" />
    <OnChangePython>

def onChange_Activate():
  Activate_Crack_Spacing = materialProxy.Parameters("Activate crack spacing").GetValue()
  if Activate_Crack_Spacing == 1:
    materialProxy.Parameters("Crack spacing").SetVisibility("normal")
  else:
    materialProxy.Parameters("Crack spacing").SetVisibility("hide")
  Activate_Crack_Spacing_Min = materialProxy.Parameters("Activate crack spacing
min").GetValue()
  if Activate_Crack_Spacing_Min == 1:
    materialProxy.Parameters("Crack spacing min").SetVisibility("normal")
  else:
    materialProxy.Parameters("Crack spacing min").SetVisibility("hide")

  Activate_Tension_Stiffening = materialProxy.Parameters("Activate tension
stiffening").GetValue()
  if Activate_Tension_Stiffening == 1:
    materialProxy.Parameters("Tension stiffening").SetVisibility("normal")
  else:
    materialProxy.Parameters("Tension stiffening").SetVisibility("hide")
  Activate_Aggregate_Interlock = materialProxy.Parameters("Activate aggregate
interlock").GetValue()
```

```
  if Activate_Aggregate_Interlock == 1:
    materialProxy.Parameters("Agg size").SetVisibility("normal")
  else:
    materialProxy.Parameters("Agg size").SetVisibility("hide")
  Activate_Shear_Factor = materialProxy.Parameters("Activate shear factor").GetValue()
  if Activate_Shear_Factor == 1:
    materialProxy.Parameters("Shear factor").SetVisibility("normal")
  else:
    materialProxy.Parameters("Shear factor").SetVisibility("hide")
  Activate_Unloading_factor = materialProxy.Parameters("Activate unloading factor").GetValue()
  if Activate_Unloading_factor == 1:
    materialProxy.Parameters("Unloading factor").SetVisibility("normal")
  else:
    materialProxy.Parameters("Unloading factor").SetVisibility("hide")
  Activate_Function_for_EPS_CP = materialProxy.Parameters("Activate function for EPS
CP").GetValue()
  if Activate_Function_for_EPS_CP == 1:
    materialProxy.Parameters("Function for EPS CP").SetVisibility("normal")
    materialProxy.Parameters("Plastic strain EPS CP").SetVisibility("hide")
  else:
    materialProxy.Parameters("Function for EPS CP").SetVisibility("hide")
    materialProxy.Parameters("Plastic strain EPS CP").SetVisibility("normal")
  Activate_Crush_Band_Min = materialProxy.Parameters("Activate crush band min").GetValue()
  if Activate_Crush_Band_Min == 1:
    materialProxy.Parameters("Crush band min").SetVisibility("normal")
  else:
    materialProxy.Parameters("Crush band min").SetVisibility("hide")
  Activate_Substepping = materialProxy.Parameters("Activate substepping").GetValue()
  if Activate_Substepping == 1:
    materialProxy.Parameters("Substeps per Ft").SetVisibility("normal")
    materialProxy.Parameters("Max substeps").SetVisibility("normal")
    materialProxy.Parameters("Limit tau crack").SetVisibility("normal")
  else:
    materialProxy.Parameters("Substeps per Ft").SetVisibility("hide")
    materialProxy.Parameters("Max substeps").SetVisibility("hide")
    materialProxy.Parameters("Limit tau crack").SetVisibility("hide")
  Activate_Function_for_Wd = materialProxy.Parameters("Activate function for Wd").GetValue()
  if Activate_Function_for_Wd == 1:
    materialProxy.Parameters("Function for Wd").SetVisibility("normal")
    materialProxy.Parameters("Critical comp disp Wd").SetVisibility("hide")
  else:
    materialProxy.Parameters("Function for Wd").SetVisibility("hide")
    materialProxy.Parameters("Critical comp disp Wd").SetVisibility("normal")
  Activate_Damping = materialProxy.Parameters("Activate particular damping").GetValue()
  if Activate_Damping == 1:
    materialProxy.Parameters("Damping mass coeff").SetVisibility("normal")
    materialProxy.Parameters("Damping stiffnes coeff").SetVisibility("normal")
  else:
    materialProxy.Parameters("Damping mass coeff").SetVisibility("hide")
    materialProxy.Parameters("Damping stiffnes coeff").SetVisibility("hide")
  return
    </OnChangePython>
    <InputPython>

# Important help functions to write material into input
# ----------------------------------------------------------
# header_inp = header_to_inp() - returns MATERIAL ID .. NAME .. TYPE ..
# param_inp = param_to_inp(param_name, input_name)
# diagram_inp = function_param_to_inp(param_name, id)
# diagram_ref_inp = function_param_ref_to_inp(id)
# save_inp(input_string)
#from cStringIO import StringIO
result = StringBuilder()

mat_id = materialProxy.Id
Function_for_E_id = 1
Function_for_FT_id = 2
Function_for_FC_id = 3
Function_for_GF_id = 4
Function_for_EPS_CP_id = 5
Function_for_WD_id = 6
Function_for_FC0_id = 7
result.write(function_param_to_inp("Function for E", Function_for_E_id))
result.write(function_param_to_inp("Function for Ft", Function_for_FT_id))
result.write(function_param_to_inp("Function for Fc", Function_for_FC_id))
result.write(function_param_to_inp("Function for Gf", Function_for_GF_id))
result.write(function_param_to_inp("Function for EPS CP", Function_for_EPS_CP_id))
result.write(function_param_to_inp("Function for Wd", Function_for_WD_id))
result.write(function_param_to_inp("Function for Fc0", Function_for_FC0_id))
result.write(header_to_inp("CC3DNonLinCementitious2Variable",1))
result.write(param_to_inp("Thermal expansion", "ALPHA"))
result.write(param_to_inp("Rho density", "RHO"))
result.write(param_to_inp("Poisson's ratio", "MU"))
result.write(param_to_inp("Dir of pl flow BETA", "BETA"))

if (materialProxy.Parameters("Activate function for Wd").GetValue()):
  result.write("   PARAM \"WD\"  F " + str(get_function_id(mat_id,Function_for_WD_id)) + "\n")
```

```
else:
  result.write(param_to_inp("Critical comp disp Wd", "WD"))

if (materialProxy.Parameters("Activate function for EPS CP").GetValue()):
  result.write("   PARAM \"EPS_CP\"  F " + str(get_function_id(mat_id,Function_for_EPS_CP_id))
+ "\n")
else:
  result.write(param_to_inp("Plastic strain EPS CP", "EPS_CP"))

result.write(param_to_inp("Excentricity EXC", "EXC"))
result.write(param_to_inp("Fixed crack", "FIXED"))

result.write("   PARAM \"E\"  F " + str(get_function_id(mat_id,Function_for_E_id)) + "\n")
result.write("   PARAM \"FT\"  F " + str(get_function_id(mat_id,Function_for_FT_id)) + "\n")
result.write("   PARAM \"FC\"  F " + str(get_function_id(mat_id,Function_for_FC_id)) + "\n")
result.write("   PARAM \"GF\"  F " + str(get_function_id(mat_id,Function_for_GF_id)) + "\n")
result.write("   PARAM \"FC0\"  F " + str(get_function_id(mat_id,Function_for_FC0_id)) + "\n")

if materialProxy.Parameters("Activate crack spacing").GetValue():
  if materialProxy.Parameters("Activate crack spacing").GetValue() and
materialProxy.Parameters("Activate crack spacing min").GetValue():
    if materialProxy.Parameters("Crack spacing min").GetValue() >
materialProxy.Parameters("Crack spacing").GetValue():
      result.write(param_to_inp("Crack spacing min", "CRACK_SPACING"))
    else:
      result.write(param_to_inp("Crack spacing", "CRACK_SPACING"))
  else:
    result.write(param_to_inp("Crack spacing", "CRACK_SPACING"))
if materialProxy.Parameters("Activate crack spacing min").GetValue():
  result.write(param_to_inp("Crack spacing min", "CRACK_SPACING_MIN"))
if (materialProxy.Parameters("Activate tension stiffening").GetValue()):
  result.write(param_to_inp("Tension stiffening", "TENSION_STIFFENING"))
if (materialProxy.Parameters("Activate aggregate interlock").GetValue()):
  result.write(param_to_inp("Agg size", "AGG_SIZE"))
if (materialProxy.Parameters("Activate shear factor").GetValue()):
  result.write(param_to_inp("Shear factor", "SHEAR_FACTOR"))
if (materialProxy.Parameters("Activate unloading factor").GetValue()):
  result.write(param_to_inp("Unloading factor", "UNLOADING"))

if (materialProxy.Parameters("Activate crush band min").GetValue()):
  result.write(param_to_inp("Crush band min", "CRUSH_BAND_MIN"))

if (materialProxy.Parameters("Activate substepping").GetValue()):
  result.write(param_to_inp("Substeps per Ft", "SUBSTEPS_PER_FT"))
  result.write(param_to_inp("Max substeps", "MAX_SUBSTEPS"))
  if (materialProxy.Parameters("Limit tau crack").GetValue()):
    result.write(param_to_inp("Limit tau crack", "LIMIT_TAU_CRACK"))

if projectProxy.Analysis == "Dynamic":
  if materialProxy.Parameters("Activate particular damping").GetValue():
    result.write(param_to_inp("Damping mass coeff", "DAMPING_MASS"))
    result.write(param_to_inp("Damping stiffnes coeff", "DAMPING_STIFF"))

if projectProxy.Space == "Axisymmetric":
  result.write(param_to_inp("Radial cracks", "RADIAL_CRACKS"))

result.write("   IDEALISATION " + materialProxy.GetIdealisation() + "\n")
result.write("   ;\n")

save_inp(result.getvalue())
    </InputPython>
  </ConcreteMaterial>
</Materials>
```

5.4 Python part

User input processing takes place using a python functions written in the same file, but outside the xml part.

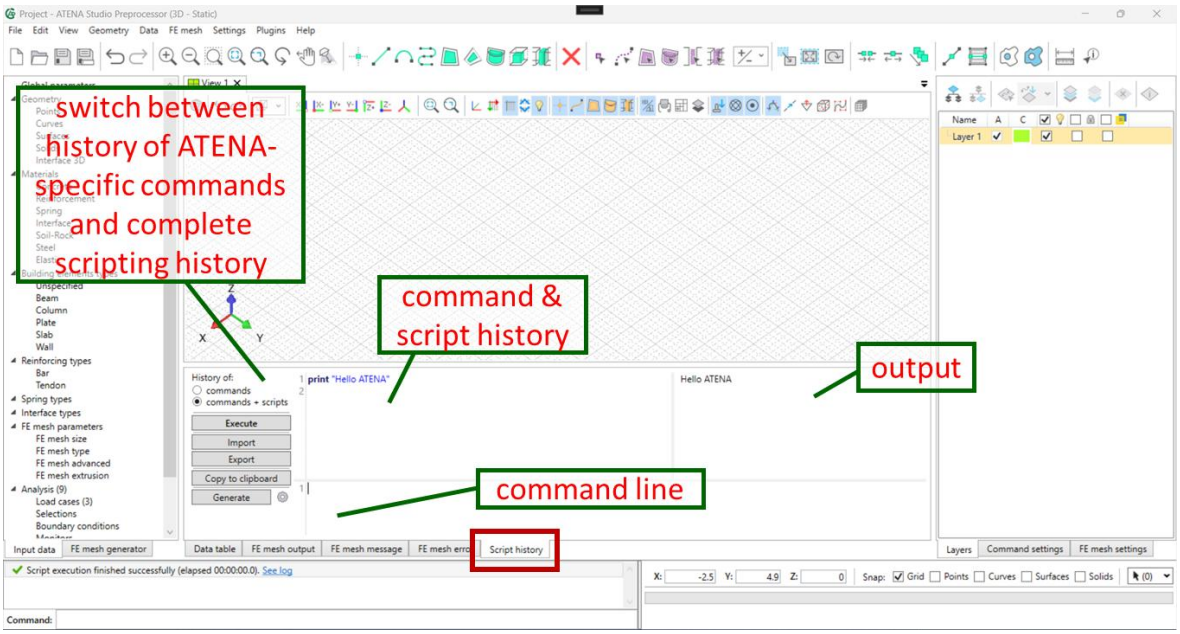
ATENA PRE commands:

- ATENA-specific commands
- automatically created after every GUI operation
- history available through “Script history” tab

2) Python scripting

- allows implementation of user-defined functions

- currently only Iron Python supported



Returning defined items:

- created items (geometrical entities, materials, element types, etc.) are returned as objects/lists after command execution
- if the new item contains lower entities, they are return as a list sorted in decreasing order (e.g., surface/lines/points)
- object's properties can be accessed (.Id, .X, etc.)

Example:

```
in:   L1 = line(startPoint = [0,0,0], endPoint = [1,2,0])   #create line
      print len(L1)      #print the length of the L1 list
➔ out: 3

in:    print L1[0]       #printing first item in the list, i.e., line
➔ out: Line segment ID = 1  (StartID = 1, EndID = 2)

in:   print L1[1]        #printing second item in the list, i.e., point
➔ out: Point ID = 2  (1, 2, 0)

in:   print L1[2]        #printing third item in the list, i.e., point
➔ out: Point ID = 1  (0, 0, 0)

in:   P1 = L1[2]
      print "Point " + str(P1.Id) + ", x-cor = " + str(P1.X)
➔ out: Point 1, x-cor = 0.0
```

Fig. 33: Example of Python scripts for definition of geometrical points.

Tab. 3: An example of Python script for programming ATENA user specific plugins.

```
def CreateBeam(originX, originY, originZ, heigth, width, length, fc, nRebar, dRebar):
	P1 = point(location = [originX, originY, originZ]) # origin of the beam
	L1 = translate(startPoint= [0,0,0], endPoint = [0,width,0], copies = 1, collapse =
"True", extrude = "True", keepParametric = "True", createSurfaces = "True", createSolids =
"True", respectLayers = "True", points = P1.Id)[0] #extruding width
	S1 = translate(startPoint = [0,0,0], endPoint = [0,0,heigth], copies = 1, collapse =
"True", extrude = "True", keepParametric = "True", createSurfaces = "True", createSolids =
"False", respectLayers = "True", curves = L1.Id)[0]  #extruding height
	V1 = translate(startPoint = [0,0,0], endPoint = [length,0,0], copies = 1, collapse =
"True", extrude = "True", keepParametric = "True", createSurfaces = "True", createSolids =
"True", respectLayers = "True", surfaces = S1.Id)[0] #extruding length

	concreteName ="Concrete_" + str(fc) + "MPa_" + str(V1.Id)
	material(name = concreteName, prototype = "CC3DNonLinCementitious2", generator =
"General")
	materialGeneratorEdit(material = concreteName, parameter = "StrengthValue", value =
fc, unit = "MPa")
	materialGeneratorRun(material = concreteName)
	elementTypeName ="Beam_" + str(V1.Id)
	el1 = elementType(name = elementTypeName, type = "BeamType", material = concreteName)
	elementTypeToGeometryAssign(name =elementTypeName, solids = V1.Id)

	if nRebar > 0:
		rebarName="Steel_" +  str(V1.Id)
		material(name = rebarName, prototype = "CCReinforcement", generator =
"General")
		materialGeneratorRun(material = rebarName)
		elementRebarName="Rebar_" +  str(V1.Id)
		elementType(name = elementRebarName, type = "BarType", material = rebarName)
		elementTypeEdit(name = elementRebarName , parameter = "ReBondType", value =
"PerfectBond")
		elementTypeEdit(name = elementRebarName, parameter = "ProfileDiameter", value =
dRebar)

		conCover = 0.03
		pointsToExtrude = []
		if nRebar > 1:
			rebarSpacing = (width - 2*conCover)/(nRebar-1)
			newPoint = point(location = [originX +conCover , originY + conCover,
originZ + conCover])
			pointsToExtrude.append(newPoint)
			for i in range(nRebar-1):
				newPoint = point(location = [originX +conCover , originY +
conCover + (i+1)*rebarSpacing, originZ + conCover])
				pointsToExtrude.append(newPoint)
		else:
			newPoint = point(location = [originX +conCover , originY + width/2,
originZ + conCover])
			pointsToExtrude.append(newPoint)

		rebarLinesIds=[]
		for i in range(len(pointsToExtrude)):
			newLine = translate(startPoint= [0,0,0], endPoint = [length-
conCover,0,0], copies = 1, collapse = "False", extrude = "True", keepParametric = "True",
createSurfaces = "False", createSolids = "False", respectLayers = "True", points =
pointsToExtrude[i].Id)
			rebarLinesIds.append(newLine[0].Id)

		elementTypeToGeometryAssign(name = elementRebarName, curves = rebarLinesIds)


CreateBeam(originX=0, originY=0, originZ=0, heigth=0.5, width=0.25, length=4, fc=50, nRebar=3,
dRebar=0.016)
```

The described method allows to definition of user plugins. They are loaded into the preprocessor automatically if you place them in the "Plugins" directory in the AppData Roaming folder. The complete path to the folder may look like this:

C:\Users\Mike\AppData\Roaming\Cervenka_Consulting\AtenaStudio\Preprocessor_2025\MyTemplates\Plugins

5.5 Run Time Analysis, Output and Post-Processing

Simulation execution is done via the standard ATENA run-time and post-processing Tool AtenaStudio, see ATENA Studio User's Manual [5].

5.6 Advanced Usage (Optional)

- Parametric studies with different SCM ratios
- Integration with NumPy/Pandas for optimization
- Automated generation of spiral configurations
- Scripting experimental comparison plots

6 Validation and Example Manual

This chapter presents a set of **representative examples and validation studies** demonstrating the applicability, robustness, and practical usability of the ATENA Module for Green Concrete Modelling and Design developed as the software result **TM04000013-V2** within the CeSTaR-3 project. The purpose of this chapter is to document how the implemented functionality can be applied to realistic engineering problems and to verify that the software performs consistently with established experimental evidence and previously validated ATENA benchmarks.

The presented examples cover different **levels of structural modelling**, ranging from reference benchmark problems to design-oriented analyses of structural components and systems. They illustrate typical modelling workflows, including the definition of time-dependent material properties, mapping of experimental material data, construction-sequence modelling, and nonlinear analysis under monotonic, cyclic, and combined loading scenarios. Wherever applicable, material parameters are derived from or calibrated using data from the experimental database of green concrete materials (**project result TM04000013-V6**), ensuring traceability between experimental research and numerical simulation.

The validation approach adopted in this chapter combines:

- reference ATENA benchmark studies confirming consistency with established nonlinear analysis performance,
- problem-specific examples demonstrating the added value of the green concrete modelling extensions,
- step-by-step modelling procedures illustrating practical engineering workflows.

The examples are not intended to represent an exhaustive verification of all possible applications, but rather to demonstrate **typical and representative use cases** relevant to engineering practice. The presented results confirm that the software enables stable nonlinear simulations, realistic representation of damage and cracking mechanisms, and meaningful interpretation of structural response for design verification and comparative studies.

By documenting both the modelling process and the resulting structural response, this chapter serves as an **example manual** for users and as a **validation record** for the software result. It provides evidence that the ATENA Module for Green Concrete Modelling and Design is suitable for applied research and engineering use, supporting the objectives of the CeSTaR-3 project to facilitate the practical adoption of sustainable concrete technologies through reliable numerical modelling.

6.1 Summary of General ATENA Validation Benchmarks

This section summarizes additional blind competition results of ATENA software. Verification of simulation models for concrete structures is typically conducted through comparison with experimental data. Interesting insight can be obtained from blind predictions in international competitions, when the experimental results are not known at the time of the analysis. Fig. 34 to Fig. 39 summarizes several such contests and benchmarks in which the authors participated, for more details see Cervenka et. al. (2024) [23]. The overall summary is prodived in Fig. 39, where the predicted strength is normalized by the ratio F_{sim}/F_{exp} with 22 cases from seven benchmark contests displayed on the horizontal axis. The vertical bars represent the prediction scatter, while the author's results are marked in green.

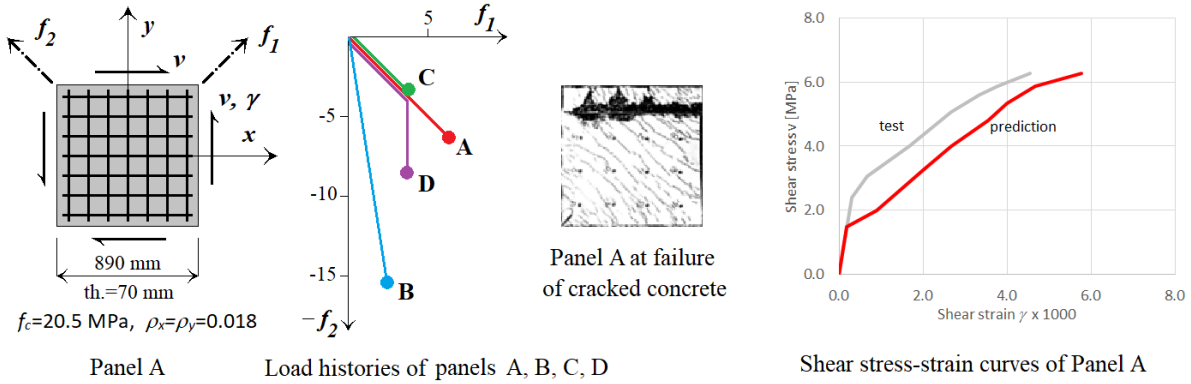


Fig. 34. Reinforced concrete panels, Toronto 1985.

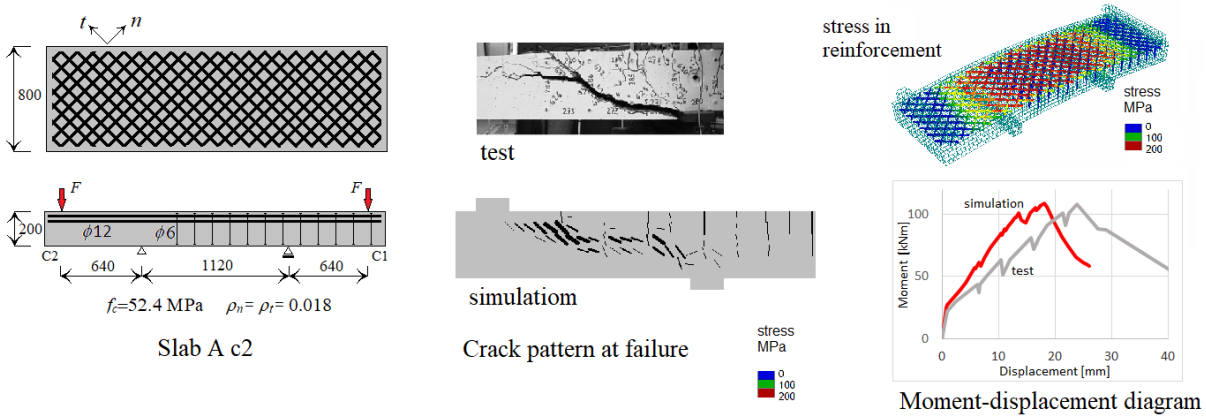


Fig. 35. Reinforced concrete slab, ETH 2005.

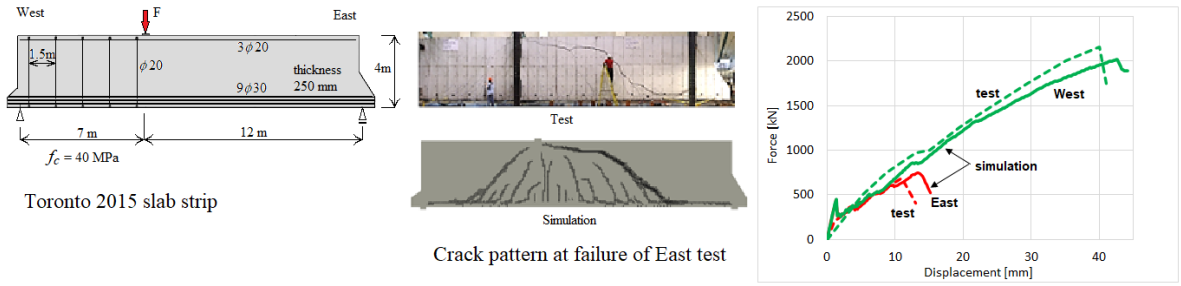


Fig. 36 Shear strength of very thick slab strip, Toronto 2015.

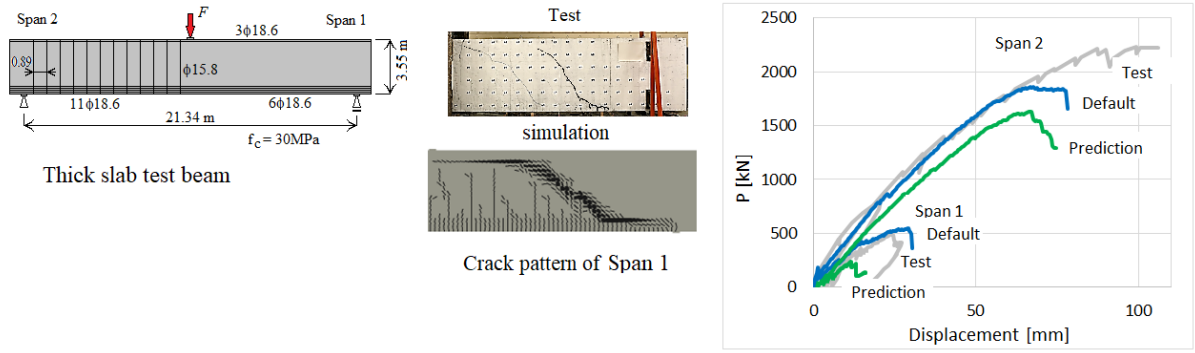


Fig. 37 Shear strength of reinforced concrete foundation, Berkeley 2021.

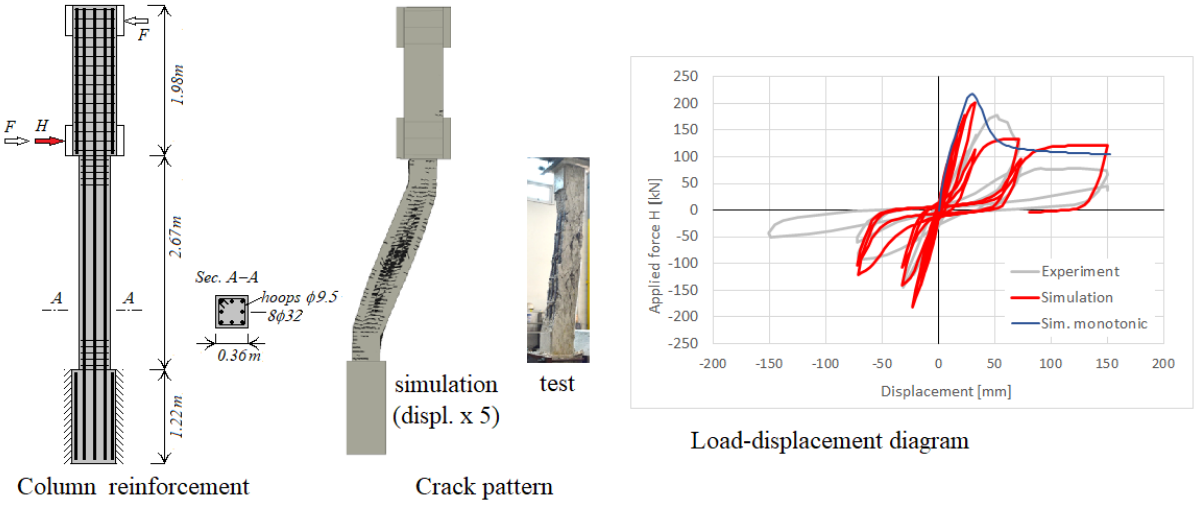


Fig. 38 Cyclic loaded reinforced concrete column.

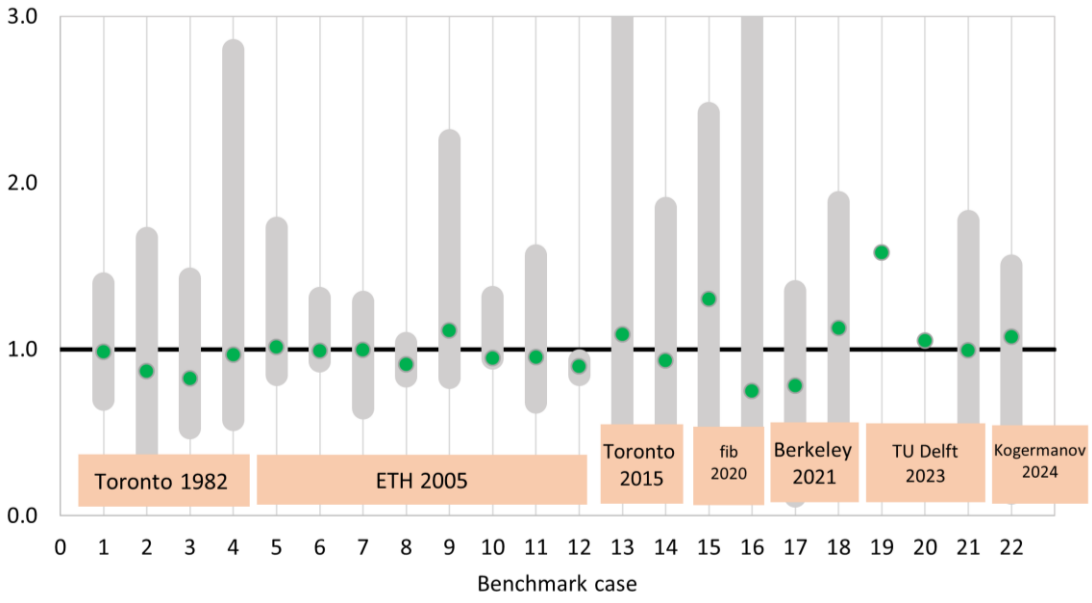


Fig. 39. Benchmark summary.

Over the past 40 years, the engineering community has shown sustained interest in improving simulation tools; however, no clear trend toward reduced uncertainty has emerged. The benchmarks primarily focus on shear or bending strength, and the wide prediction scatter reflects a limited understanding of shear failure. In addition, strength, stiffness, deformations, and crack patterns were also considered in the evaluation.

6.2 Example of Column-Beam Connection Modelling and Design

This example illustrates the application of the ATENA Module for Green Concrete Modelling and Design to the **nonlinear analysis and design verification of a reinforced concrete column–beam connection**. The example represents a typical structural detail encountered in building structures and demonstrates the practical use of the software developed for **engineering assessment and design-oriented evaluation**.

The analyzed detail (see Fig. 40) represents a typical composite steel-concrete connection with multi-spiral reinforcement, as investigated in this project and developed by the Taiwanese partners. Such a connection has also been extensively modelled and tested during this project. The strength of the detail as well as its behavior during cycling loading has been extensively studied and validated by other workpackages and developed software tools. Namely the project software outcome V1 [24] and paper [51] describes this aspect of the project in more detail. The model of the analyzed column-beam connection design can be parametrically generated using the other developed ATENA module for MRCS parametric modelling and design [24].

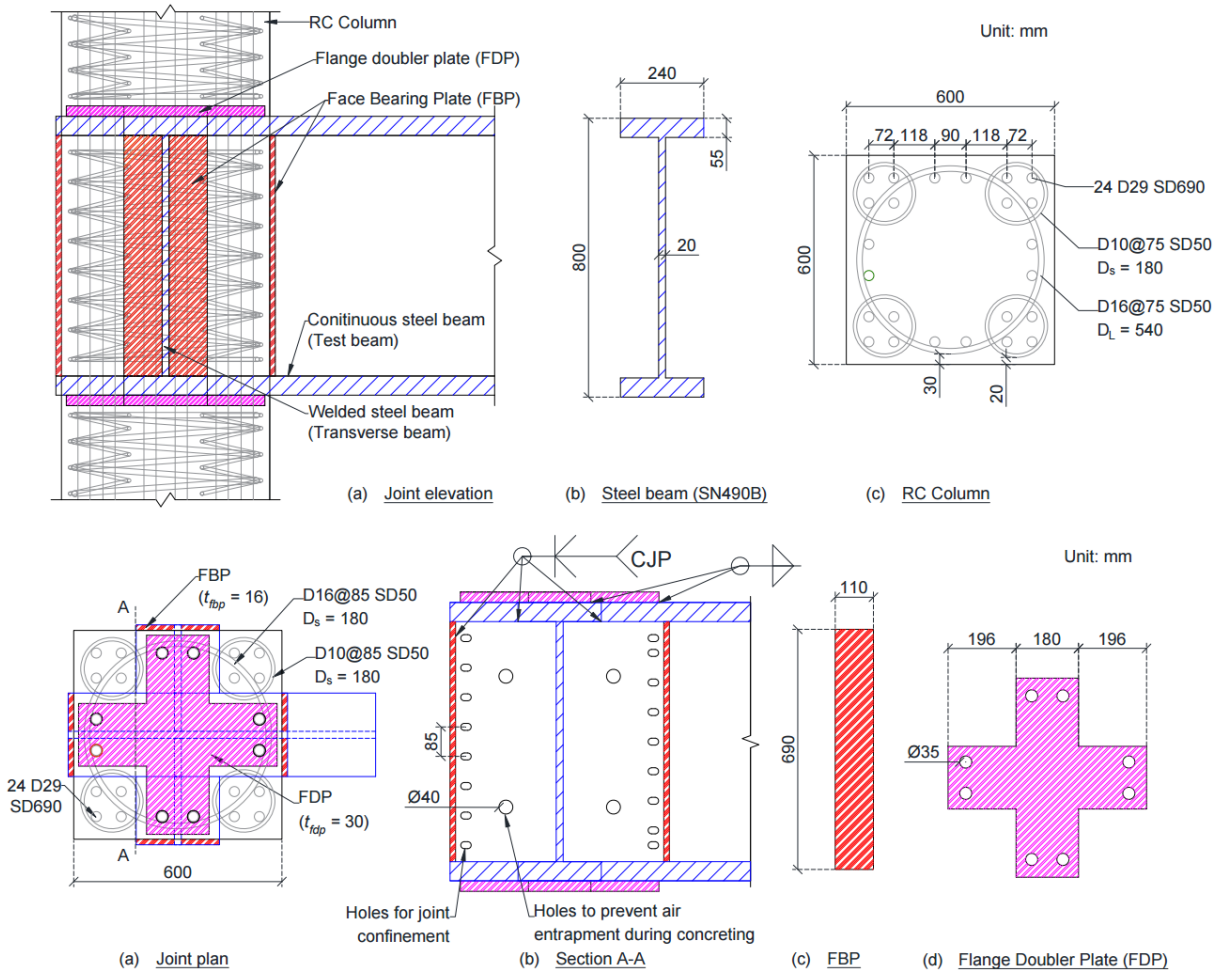


Fig. 40: The detail of the steel-concrete MRCS connection used as an example for Green Concrete modelling and simulation.

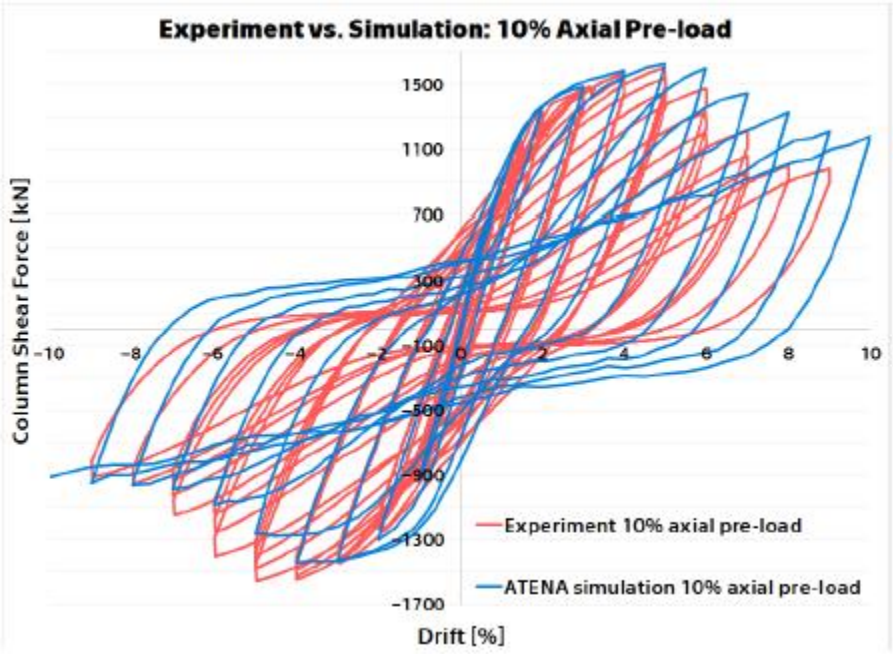


Fig. 41: Validation of the column-beam MRCS connection in software module [24] [51].

The analyzed column–beam connection is modelled using a three-dimensional nonlinear finite element representation, including concrete, reinforcement, and bond interaction. The behavior of concrete is described using the fracture–plastic constitutive model implemented in ATENA and extended within the CeSTaR-3 project to account for **green concrete with supplementary cementitious materials (SCMs)**. Material parameters may be derived from experimentally validated data, including values available in the experimental database **TM04000013-V6**, ensuring consistency between experimental observations and numerical modelling.

The numerical model captures the key mechanisms governing the response of the connection, including:

- cracking and crushing of concrete in the joint region,
- nonlinear behavior of longitudinal and transverse reinforcement,
- stress redistribution between the beam and column under increasing load,
- interaction between bending, shear, and axial forces.

The connection is subjected to a loading scheme representative of design verification scenarios, such as monotonic or cyclic loading applied to the beam end while maintaining appropriate boundary conditions for the column. The analysis is performed using incremental–iterative nonlinear solution procedures, allowing detailed observation of stiffness degradation, damage development, and ultimate capacity.

The results demonstrate that the software is capable of:

- predicting the load–deformation response of the column–beam connection,
- identifying critical regions of cracking and damage concentration,
- supporting evaluation of reinforcement detailing and joint performance,
- providing output suitable for design verification and comparative studies.

This example confirms that the ATENA Module for Green Concrete Modelling and Design can be effectively used for **modelling, analysis, and design-oriented assessment of structural connections**, which are critical components of reinforced concrete structures. The

demonstrated workflow highlights the applicability of the software for practical engineering tasks involving green concrete materials, bridging experimental research results and everyday design practice.

In this example, the detail of the connection is modelled as being part of a bigger structure, and being located at the 22th floor of the analyzed building, i.e. 4th level from the top (see Fig. 42).

This example problem will model the connection detail. The numerical model is shown in Fig. 43. The loads on this connection detail will be gradually applied to simulate the loads coming from the top floors, and they will simulate the gradual building construction. The application of loads is slightly simplified such that the focus is placed on the development of the material strength in relation to the gradual increase of loads rather than on the complexity of the real life construction process and the required design verifications. In this example scenario, the following load history is assumed as listed in Tab. 4.

Fig. 42: View of the whole building and the typical steel-concrete MRCS connection detail.

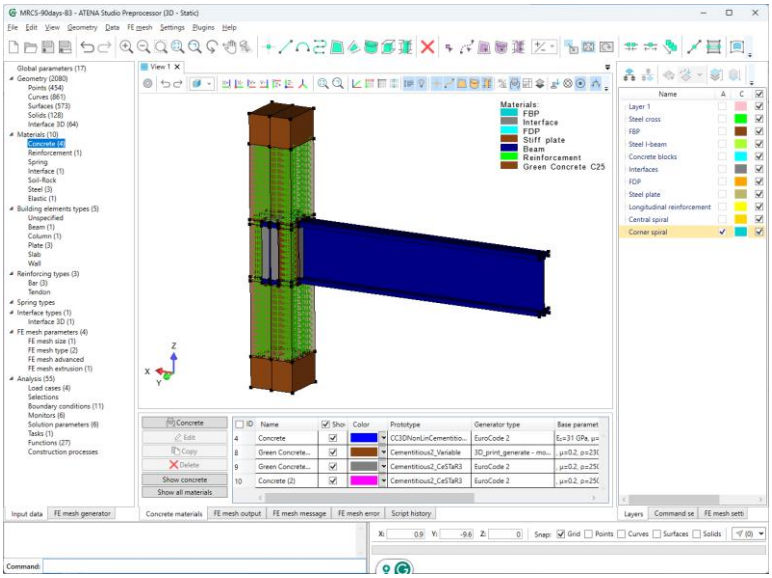


Fig. 43: View of the column-beam connection model in the software window.

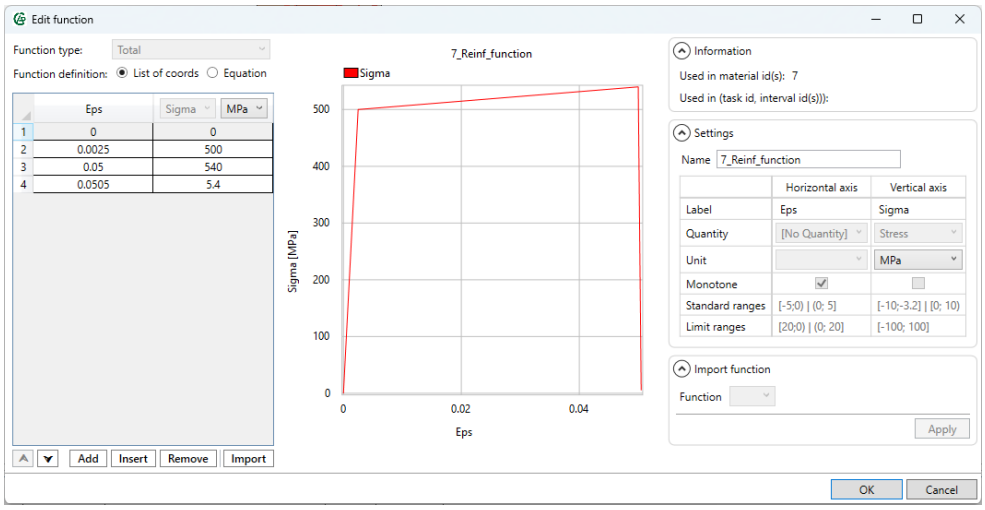


Fig. 44: Reinforcement stress-strain diagram used in the MRCS example.

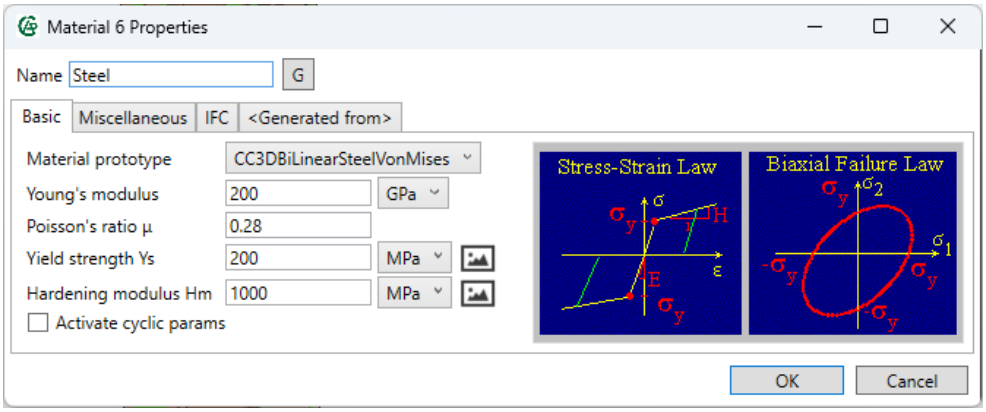


Fig. 45: Steel member material properties used in the MRCS connection example.

Tab. 4: Loading history applied in column-beam example problem

Int.	Name	# Steps	Duration [days]	Description
1	Detail dead load	1	5	Application of dead load from level I construction.
2	Beam load	5	5	Application of loads on steel beams from level I construction
3	Level II construction	1	10	Construction of Level II
4	Formwork removal	5	1	Increase of top column loads due to level II construction 2.88MN.
5	Level III construction	1	10	Construction of Level III
6	Formwork removal	5	1	Increase of top column loads due to level III construction 2.88MN.
7	Level IV construction	1	10	Construction of Level IV
8	Formwork removal	5	1	Increase of top column loads due to level IV construction 2.88MN.
9	Add. construction	5	90	Gradual increase of column forces by 0.72 MN due to additional construction

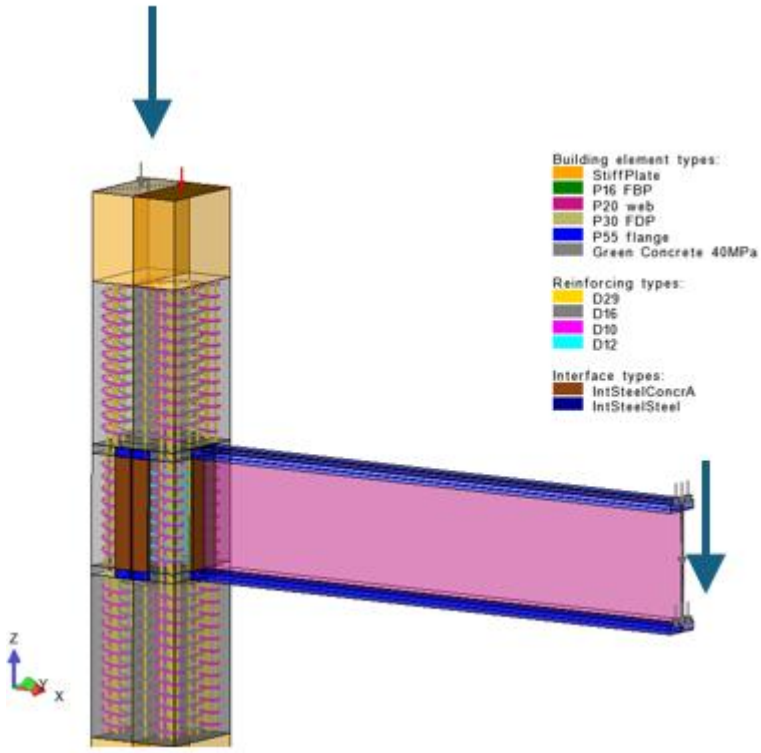


Fig. 46: Load application on the MRCS detail to simulate the loading from the rest of the structure.

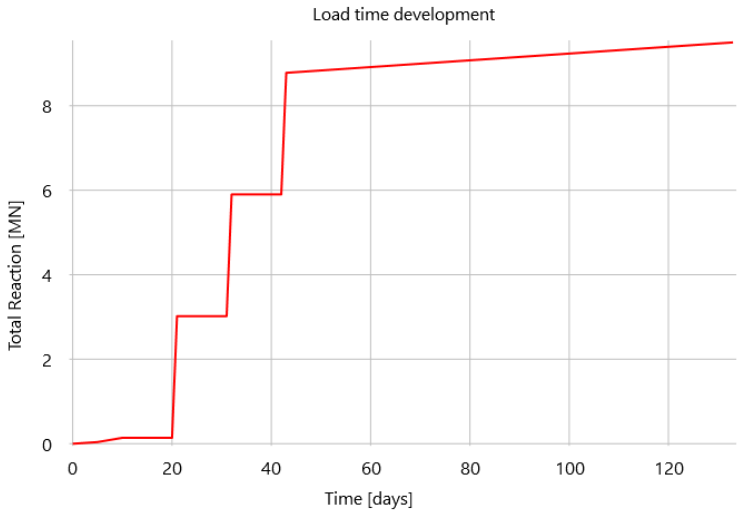


Fig. 47: The evolution of total reaction at the bottom of the column-beam MRCS detail simulating the gradual application of loads simulating the construction process.

The general approach for the optimization of construction sequence and the selection of suitable concrete type can be summarized into the following steps:

Step 1: Estimation of time history of stress development in the critical concrete elements.

Step 2: Map stress history development into the green concrete database of material maturity curves.

Step 3: Select suitable material and optimize the construction sequence.

Step 4: Verify the selected material and construction sequence by numerical simulation in ATENA-Green Concrete Module.

The proposed methodology [13] can be applied to the proposed example problem of a single column-beam MRCS connection detail as shown in Fig. 42 and Fig. 43. The application of the presented approach is described in the following subsections.

6.2.1 Step 1: Estimation of time history of stress development in the critical concrete elements.

The proposed loading history from Tab. 4 is applied to the MRCS connection detail that has been previously generated and validated using the project result V1 [24]. The numerical model is shown in Fig. 41. The average stresses in the connection are monitored and plotted in Fig. 48. The figure shows the vertical average stress development in time and is compared with the maturity curves of selected Green Concrete types from the database. The maturity curves of the green materials show the development of concrete design strength in time. The design concrete strength is determined from the material maturity curves by applying the appropriate material partial safety factors as described in the Green Concrete Modelling and Design Methodology developed in the project result V5 [13].

6.2.2 Step 2: Map stress history development into the green concrete database of material maturity curves.

From Fig. 48, it can be concluded that material labeled as PFA50 or GGBS50 are possible candidates to be used for the connection detail construction. The other possible candidate would be also the material labeled as LC15. However, the overall strength of this material might not be sufficient for subsequent live or extreme loads. The verification of live or extreme loads is not the subject of this study, which mainly focuses on the verification and optimization of the construction process and the selection of suitable material. The verification of all other required design load combinations is an essential part of the design process and may be decisive for the selection of the most suitable concrete material. In these cases, however, only the final long term concrete material strength is important, and it is therefore out of scope of this project focus.

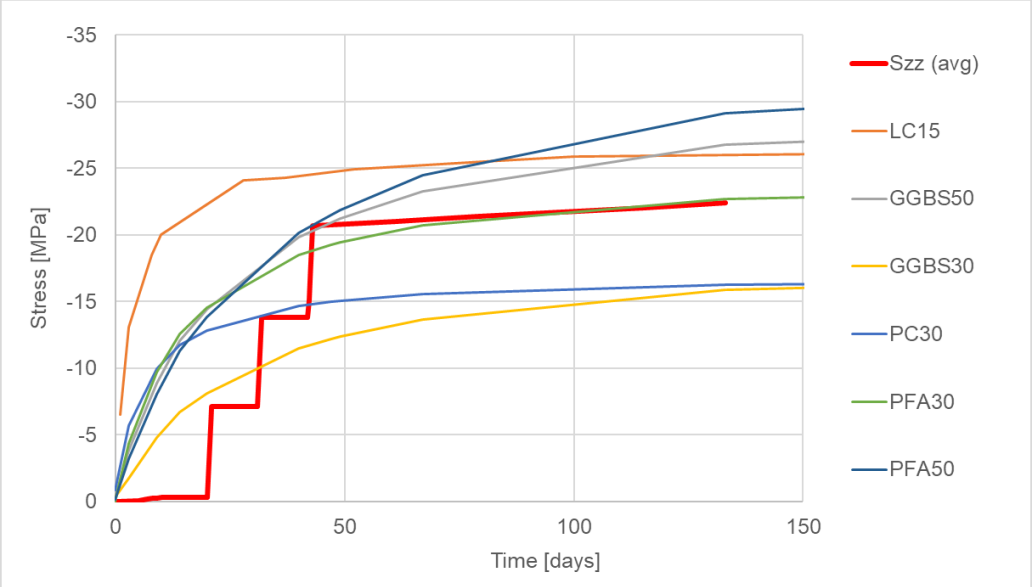


Fig. 48: Stress development in the concrete parts compared with the strength development of various Green Concrete types.

6.2.3 Step 3: Select suitable material and optimize the construction sequence.

Based on the Step 2, it was decided to choose the materials labeled as PFA30 or GGBS50 to be considered in the proposed design.

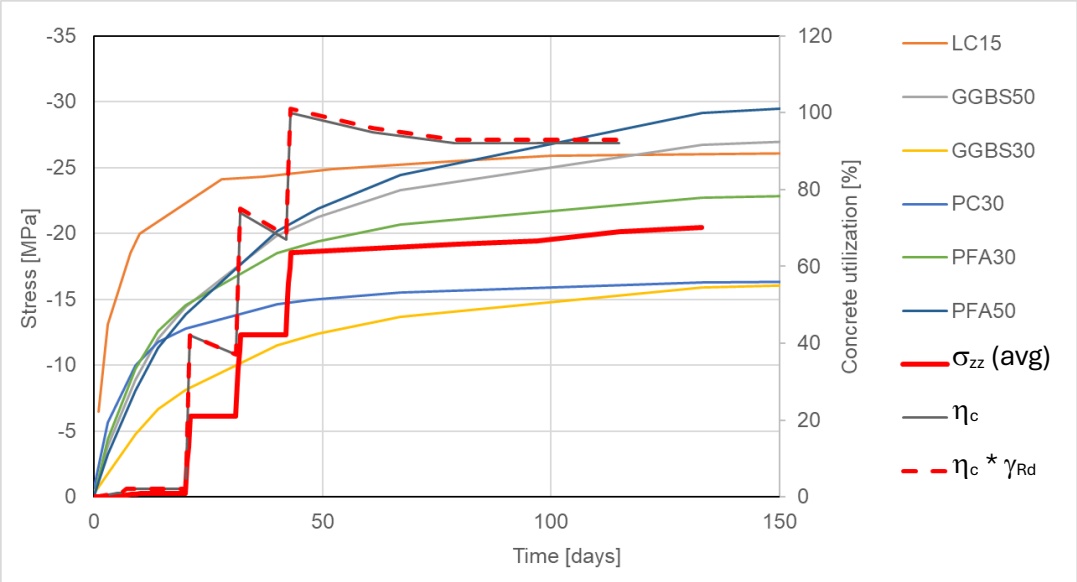


Fig. 49: Concrete stress and utilization evolution for the material PFA30

The figure shows that in terms of stress evolution the material PFA30 will closely satisfy the required stress evolution, however when the utilization factor is evaluated taking into account the required model uncertainty factor. According to the previous study [20] and the methodology [13] the smallest required model uncertainty factor γ_{Rd} = 1.01 can be used. The utilization factor in this case becomes slightly over 100%, so this material is not applicable unless the construction sequence is optimized.

This is documented on the following Fig. 50 where the average concrete stress and utilization levels are displayed for the case when the construction sequence is optimized (see Tab. 5).

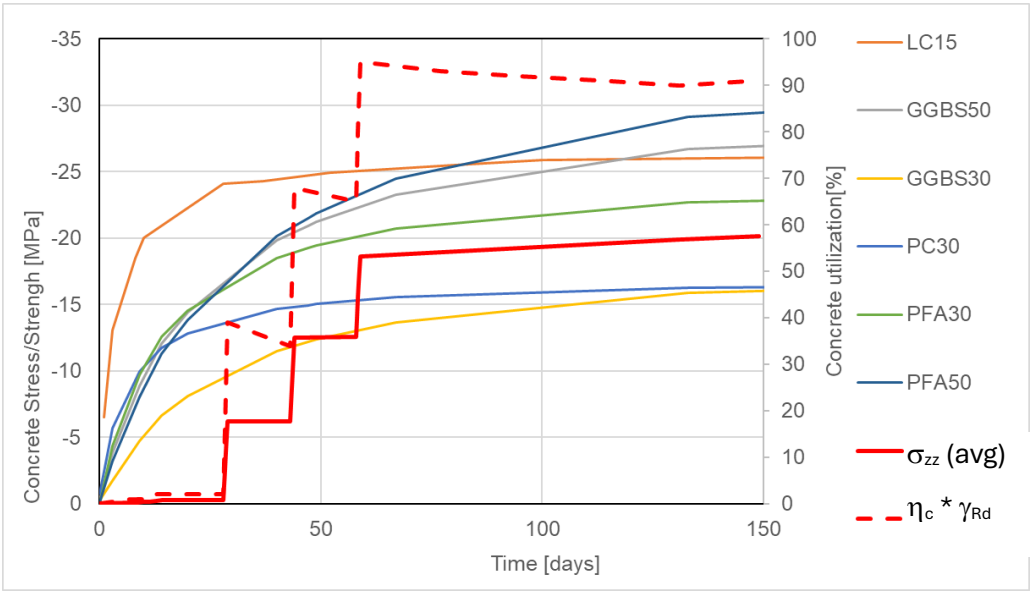


Fig. 50: Concrete stress and utilization evolution for the material PFA30 with optimized construction sequence.

Tab. 5: Optimized loading history applied in column-beam example for material PFA30

Int.	Name	# Steps	Duration [days]	Description
1	Detail dead load	1	5	Application of dead load from level I construction.
2	Beam load	5	5	Application of loads on steel beams from level I construction
3	Level II construction	1	10	Construction of Level II
4	Formwork removal	5	1	Increase of top column loads due to level II construction 2.88MN.
5	Level III construction	1	10	Construction of Level III
6	Formwork removal	5	1	Increase of top column loads due to level III construction 2.88MN.
7	Level IV construction	1	10	Construction of Level IV
8	Formwork removal	5	1	Increase of top column loads due to level IV construction 2.88MN.
9	Add. construction	5	90	Gradual increase of column forces by 0.72 MN due to additional construction

The other possibility would be to use the material labeled as GGBS50. The results for this case using the original loading history Tab. 4 is shown in Fig. 51.

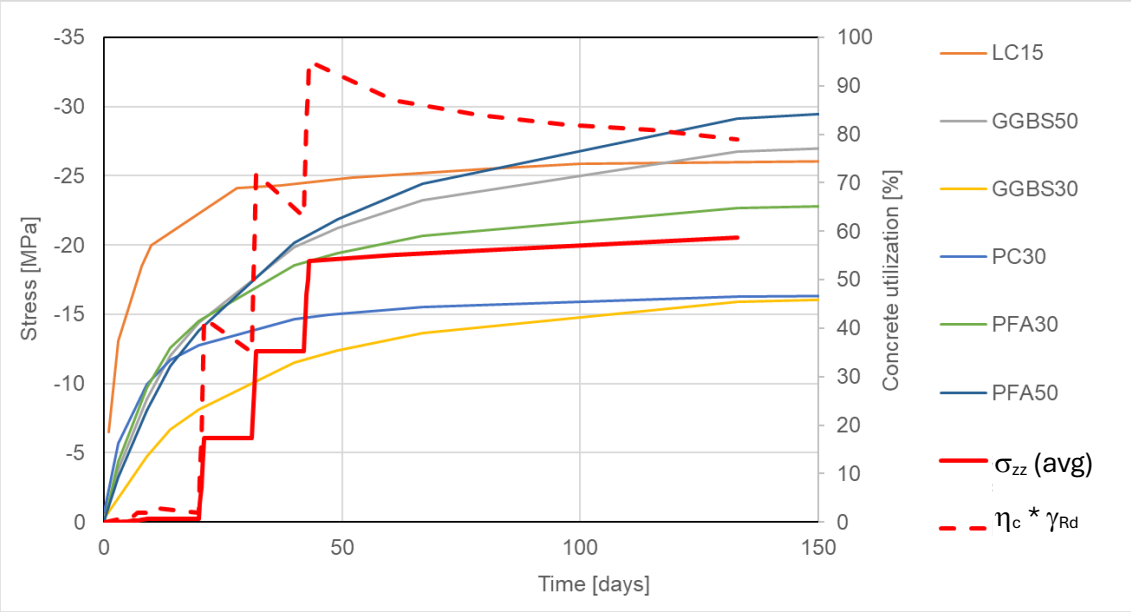


Fig. 51: Concrete stress and utilization evolution for the material GGBS50 with the original construction sequence.

6.2.4 Step 4: Verify the selected material and construction sequence by numerical simulation in ATENA-Green Concrete Module.

In this case, it can be decided to use the material GGBS50 since it shows reasonable concrete utilization levels. It is interesting to note that even though the average stress level in the connection is increasing in time as shown in Fig. 51 in times from 50 – 150 days, the overall utilization level is decreasing. This is due to the fact that the green concrete material is slowly maturing while stresses are still slightly increasing due to the continuing construction.

The final checks of the structural behavior at the early stages should involve checking the level of concrete stresses (see Fig. 54). It should be noted that higher stress than the current concrete strength can be observed due to stress localization namely in the sharp corner between concrete

and steel elements. However, the concrete should not reach the crushing state, which in ATENA software can be documented by the softening flag in the Yield/Crush Info – Softening flag as shown in Fig. 53.

Another important quantity to check is the cracking in concrete during the construction. The crack widths should be limited to micro-cracks that are barely visible. Crack widths and cracking pattern for the selected material GGBS50 are shown in Fig. 54. It demonstrates that the crack widths are below 0.03 mm. The visibility crack limit is 0.05 mm and the typical crack limit in concrete design is 0.3 mm. It is clear that the concrete cracking is satisfying both these limits.

The other quantities that should be checked are stresses in the steel members of the MRCS connection as well as the stresses in the reinforcement. In this case, they are clearly below the yielding strength of the steel σ_y = 200 MPa and f_{sy} = 500 MPa for reinforcement.

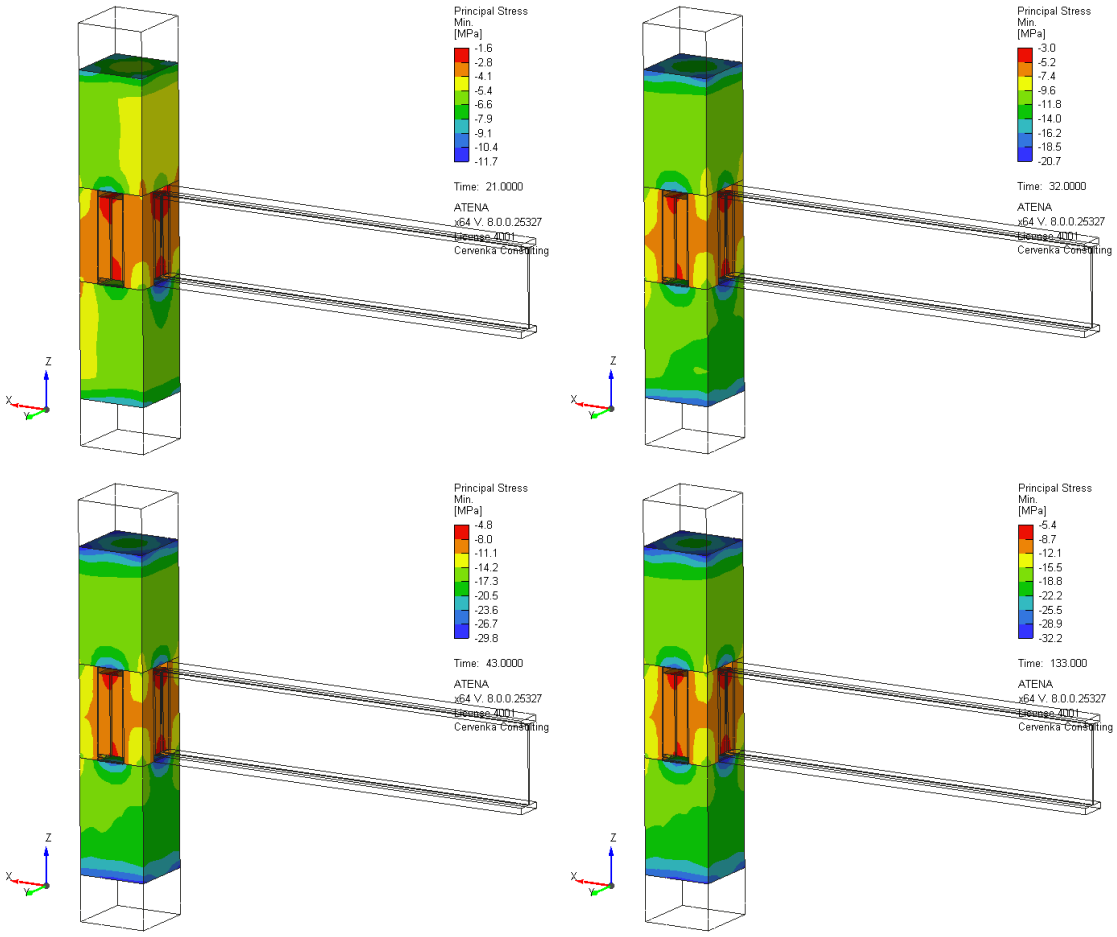


Fig. 52: Evolution of maximal concrete compressive stresses in the MRCS connection.

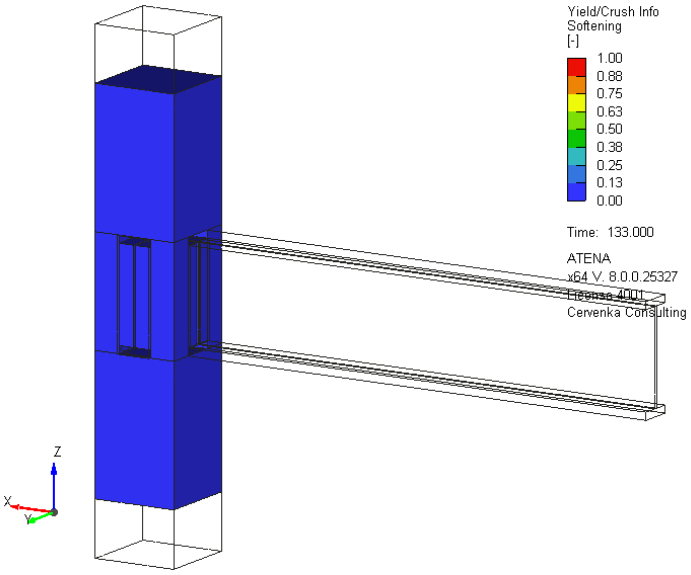


Fig. 53: Yield/Crush Info – Softening glag equal to 1 would indicate concrete crushing.

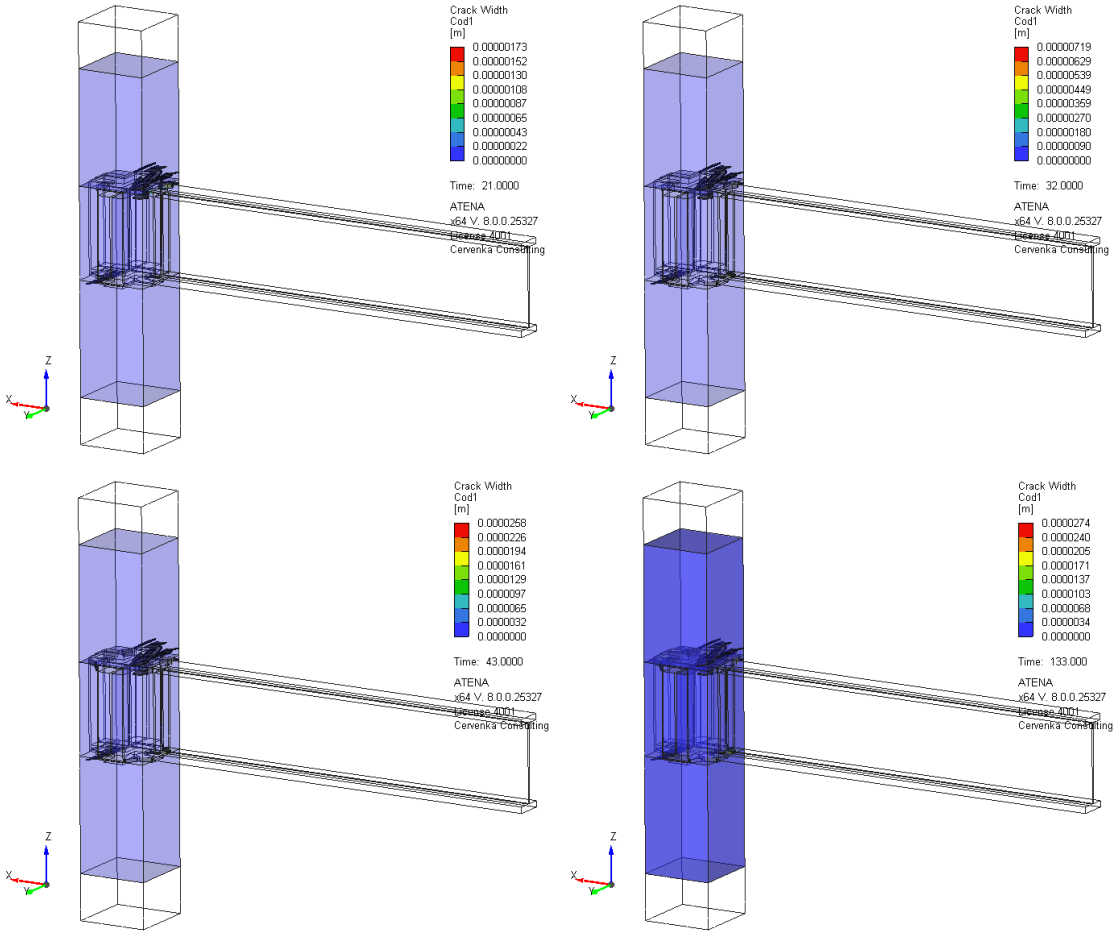


Fig. 54: Time evolution of concrete cracking in the MRCS connection during the construction process.

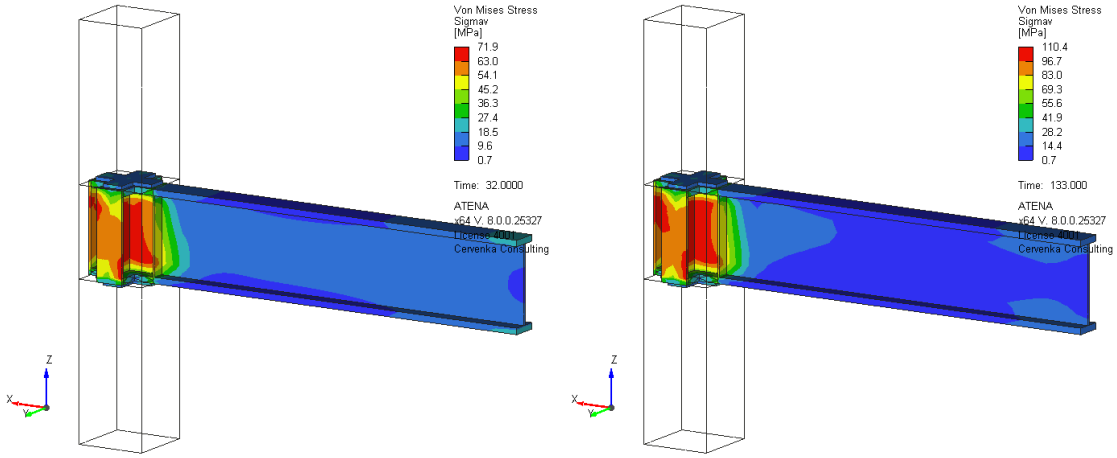


Fig. 55: Evolution of steel stresses in the MRCS connection for the material GGBS50.

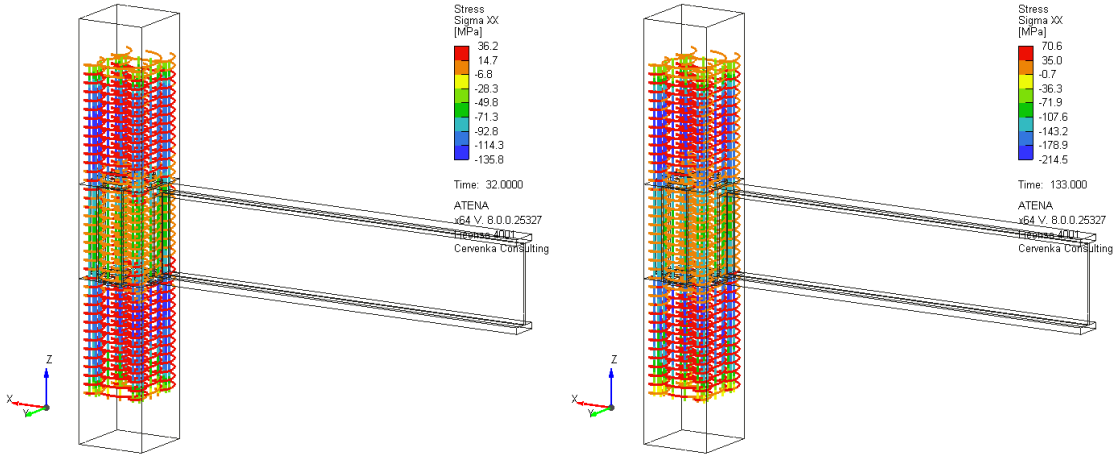


Fig. 56: Evolution of reinforcement stress in the MRCS connection for the material GGBS50.

6.3 Example of Composite Frame Modelling and Design

The next example uses the same MRCS connection detail, but this time it demonstrates and validates its application in a larger example, where the MRCS connections are part of a small composite steel concrete frame, which represents a selected part of the typical pilot building as shown in Fig. 42. The numerical model is shown in Fig. 57. It involves a MRCS connection detail as described in Section 6.2. The frame model consists of two levels with the floor height of 4.3 m and span between columns of 9.82 m in y direction and 10 m in x direction.

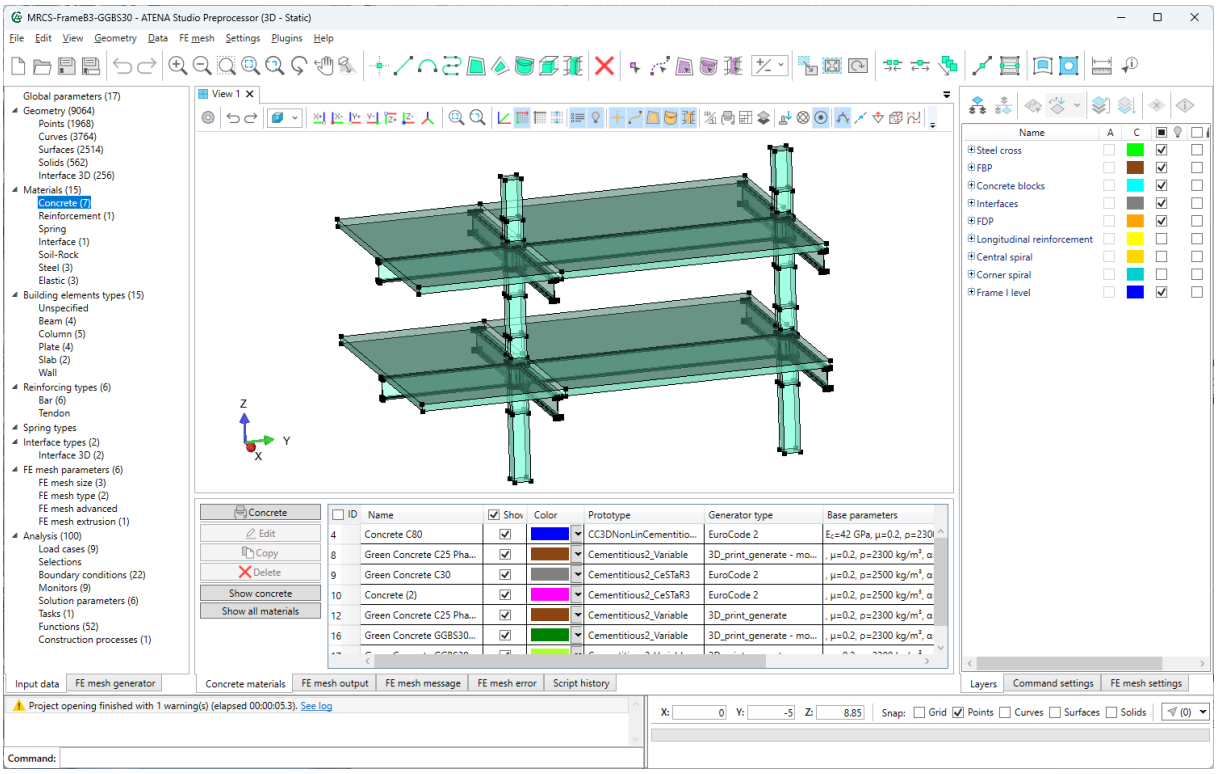


Fig. 57: Model of a small frame using MRCS connections in ATENA software with Green-Concrete construction process module.

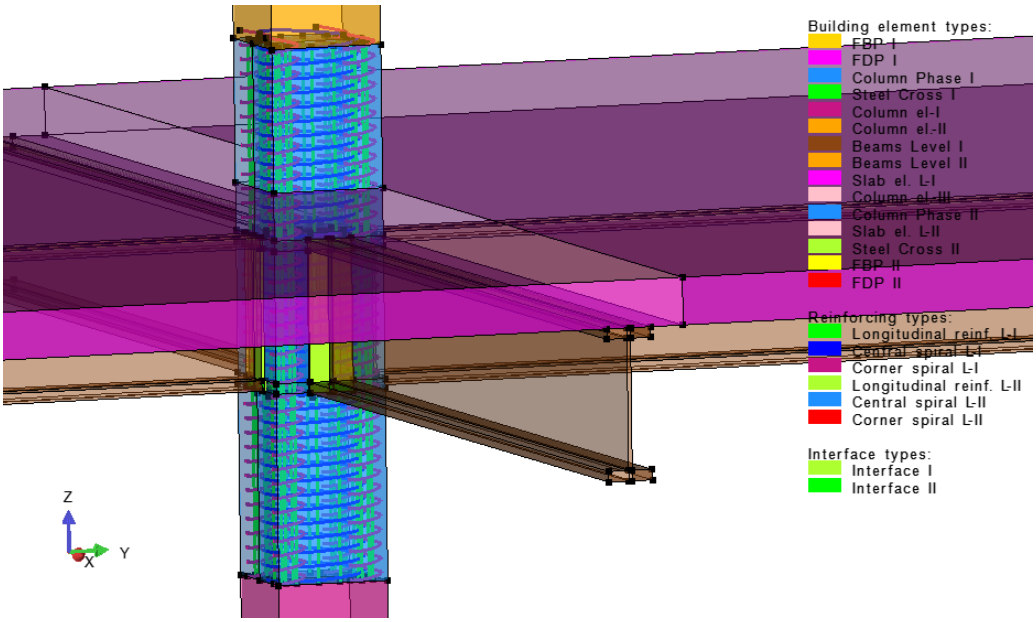


Fig. 58: Detailed reinforcement model at each MRCS connection.

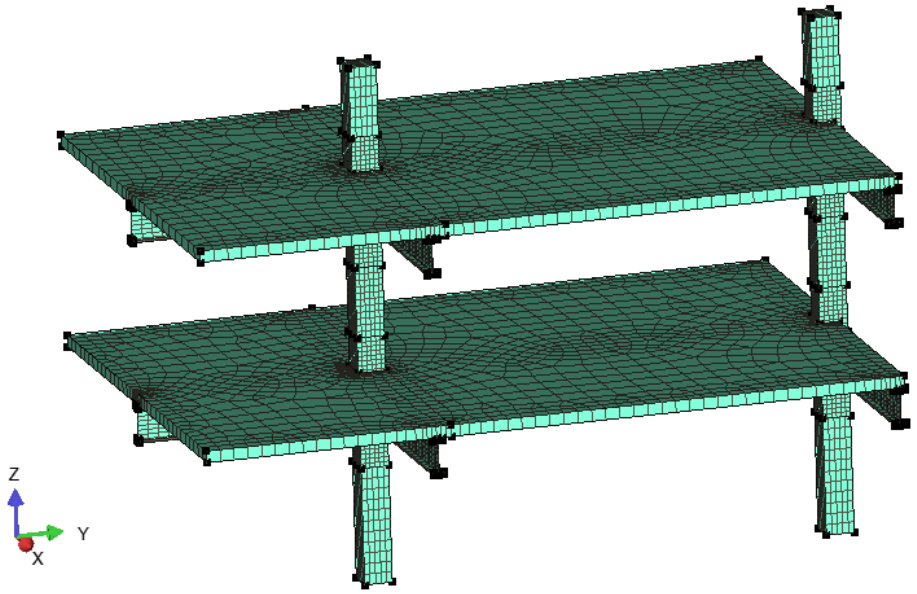


Fig. 59: Finite element model for the frame segment.

Tab. 6: Loading history applied in the frame segment example problem

Int.	Name	# Steps	Duration [days]	Description
1	Level I – Construction	1	7	Application of part of dead load for level I construction.
2	Level I – DL application, temporary supports removed	5	1	Application of remaining dead loads for level I construction
3	Level II – Construction	5	7	Applications of Level II construction loads to Level I.
4	Level II – DL application, temporary supports removed	5	1	Application of remaining dead loads for level II construction, load activation to Level – II elements.
5	Level III – Construction	5	7	Applications of Level III construction loads to Level I + II.
6	Level III – DL application, temporary supports removed	5	1	Application of remaining dead loads for level III construction, load activation to Level – III elements.
7	Additional construction	5	100	Remaining construction loads

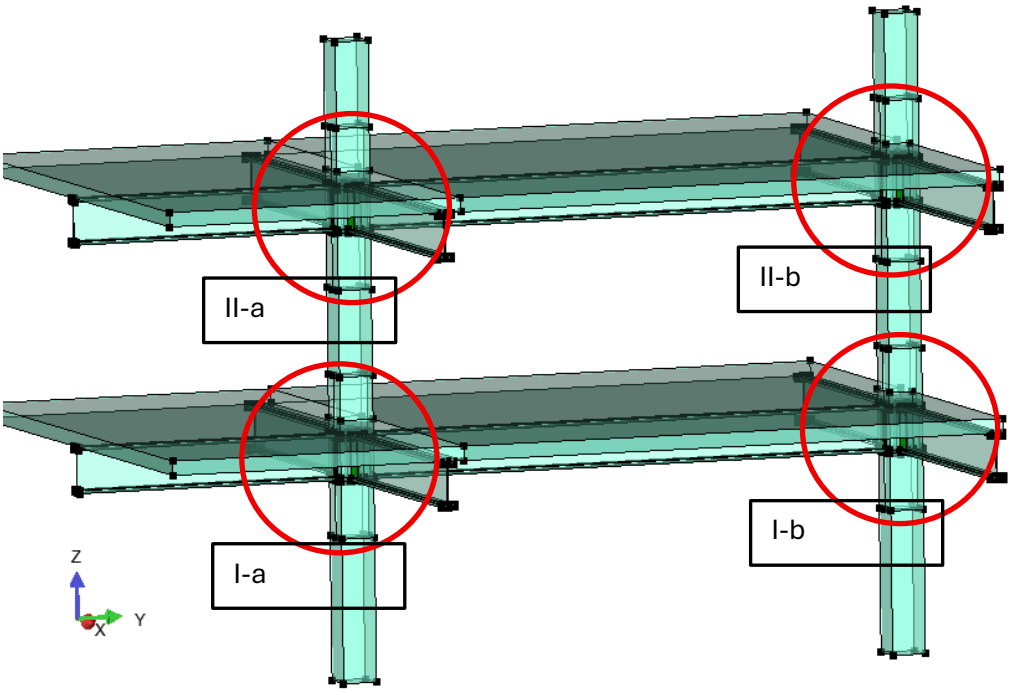


Fig. 60: Location and labeling of critical sections to be evaluated during the construction sequence and Green Concrete material optimization.

The general approach for the optimization of construction sequence and the selection of suitable concrete type according to the methodology [13] can be summarized into the following steps:

Step 1: Estimation of time history of stress development in the critical concrete elements.

Step 2: Map stress history development into the green concrete database of material maturity curves.

Step 3: Select suitable material and optimize the construction sequence.

Step 4: Verify the selected material and construction sequence by numerical simulation in ATENA-Green Concrete Module.

6.3.1 Step 1: Estimation of time history of stress development in the critical concrete elements.

The proposed loading history from Tab. 6 is applied to the frame segment with MRCS connections following the methodology [13] proposed in CeSTaR-3 project. The used numerical model is shown in Fig. 59.

In this case, two new features of the new ATENA – CeSTaR-3 Green Concrete module will be utilized:

- Modelling and simulation of material maturing and time development of material parameters.

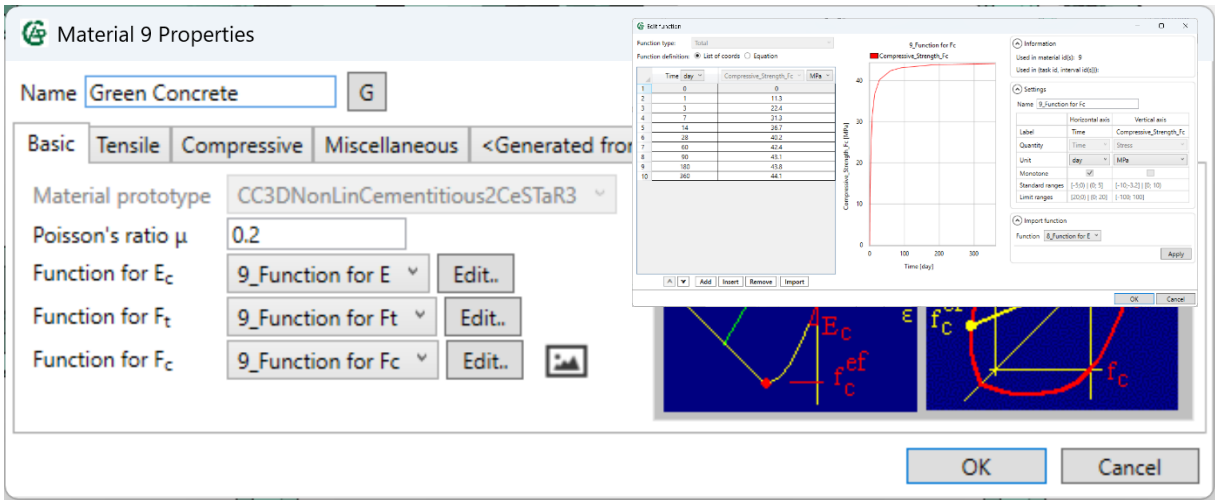


Fig. 61: Material definition dialog for the new time dependent CeSTaR3 material model.

- Modelling of construction process, i.e. gradual activation of various parts of the numerical model and their corresponding loads.

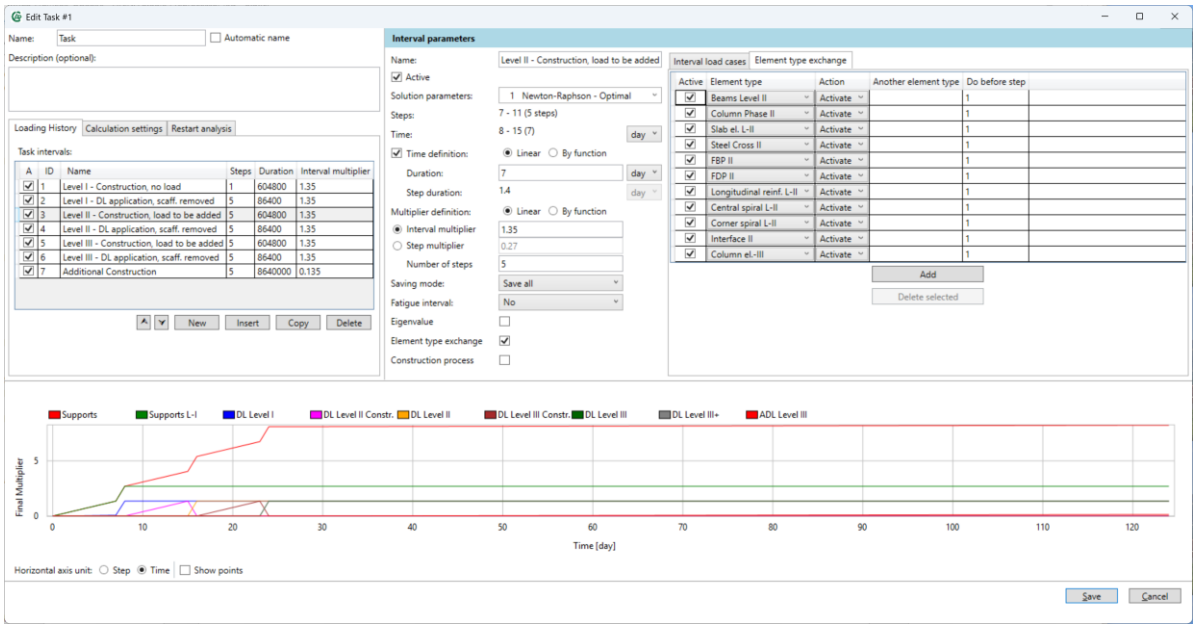


Fig. 62: Loading history and activation of individual structural elements in the ATENA – CeSTaR3 Green Concrete module.

6.3.2 Step 2: Map stress history development into the green concrete database of material maturity curves.

The average stresses in the investigated connections are monitored and plotted in Fig. 63. The figure shows the vertical average stress development at each connection depending on the age of each element and is compared with the maturity curves of selected Green Concrete types from the database.

The maturity curves of the green materials show the development of concrete design strength in time. The design concrete strength is determined from the material maturity curves by applying the appropriate material partial safety factors as described in the Green Concrete Modelling and Design Methodology developed in the project result V5 [13].

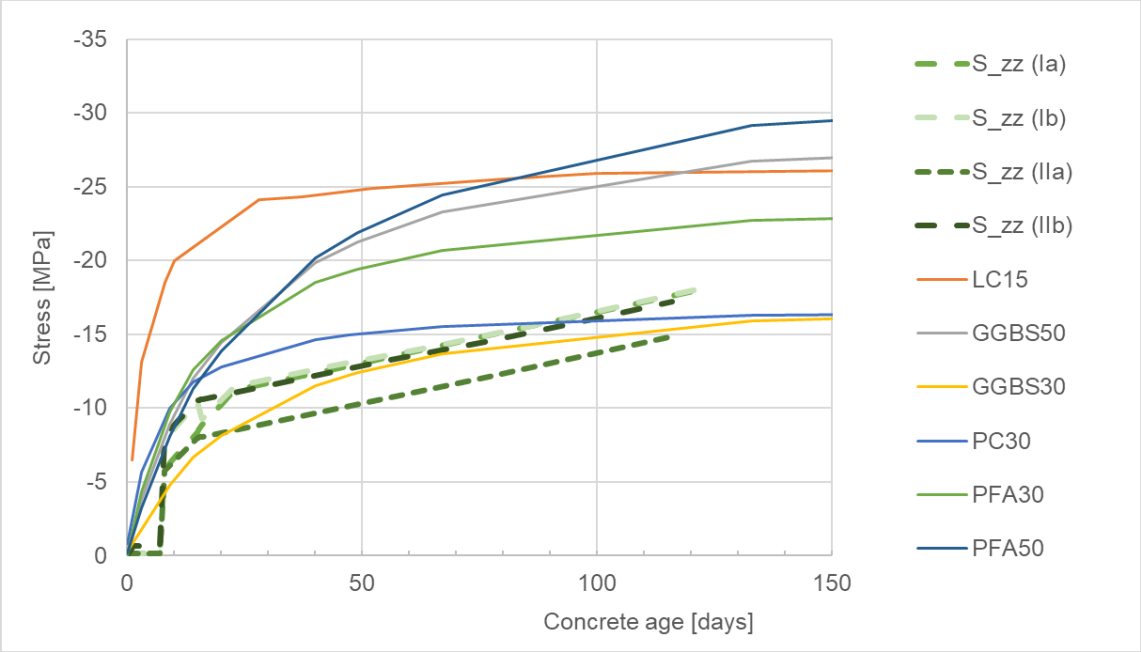


Fig. 63: Stress development in the concrete parts compared with the strength development of various Green Concrete types. The dashed lines show the average concrete stress development in the investigated MRCS connections (I-a to II-b, see Fig. 60). Horizontal axis is time in days from the beginning of each level construction.

6.3.3 Step 3: Select suitable material and optimize the construction sequence.

Based on the Step 2 and , it is possible to select suitable candidates for Green Concrete material. It is decided to choose the materials labeled as PFA30 or GGBS30 to be considered in the proposed design.

Fig. 64 shows the evolution of concrete stresses at each connection labeled Ia to IIb (see Fig. 60) based on the age of each element.

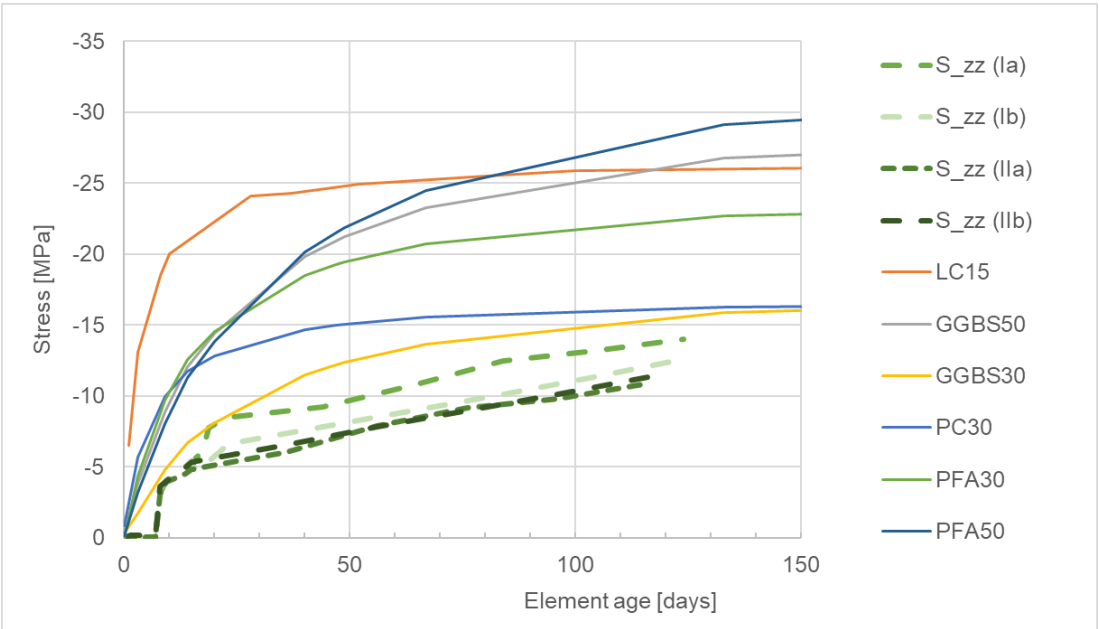


Fig. 64: Concrete stress evolution according to age at each connection detail for the material GGBS30.

The concrete utilization at each connection is shown in Fig. 65. It shows that the connection I-a has a utilization level higher than 100% at day 20 during the construction. This could addressed by modifying the construction sequence, i.e. increase the construction time of level II or by using another Green Concrete material.

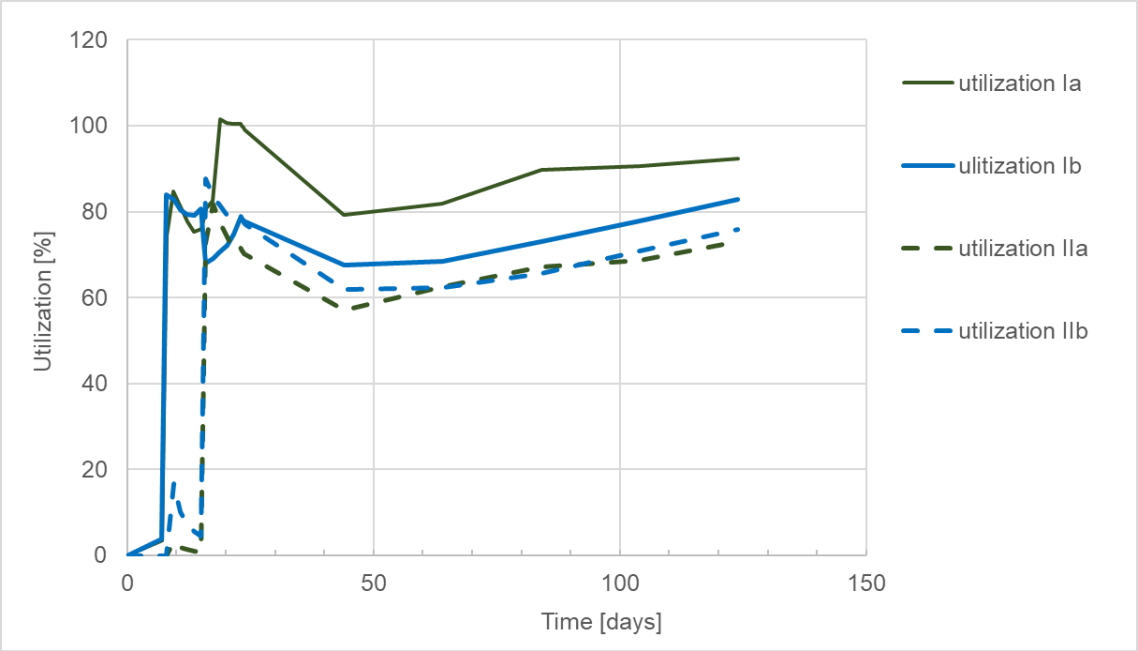


Fig. 65: Utilization of concrete at investigated connections during the construction process for the material GGBS30.

The results for different material PFA30 is shown in

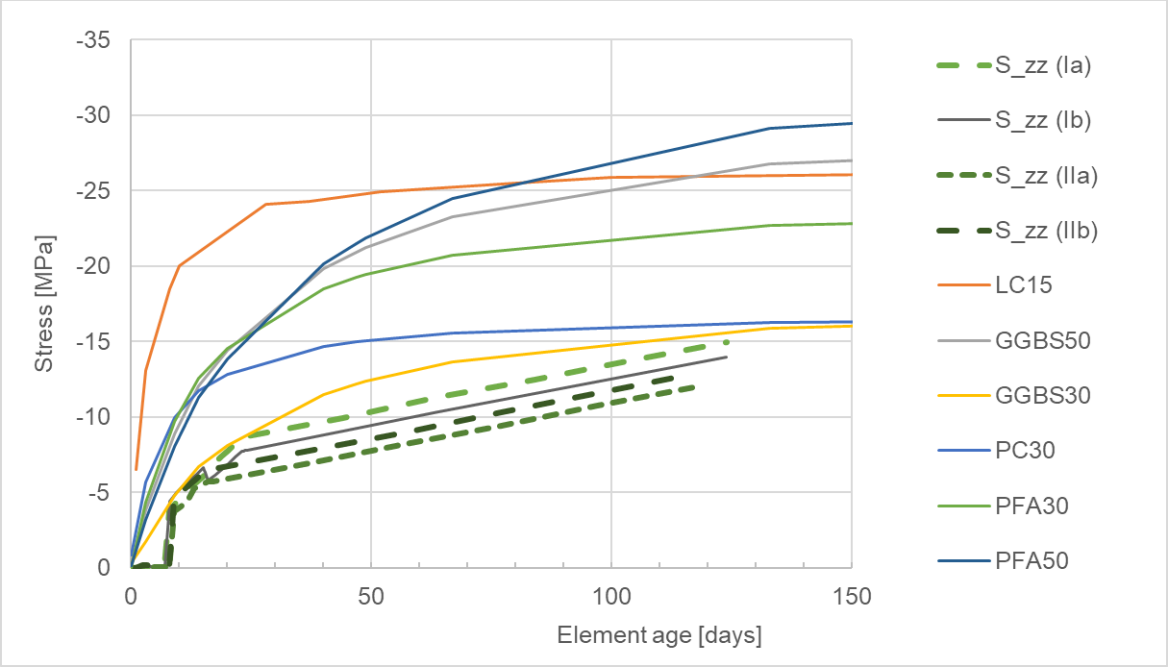


Fig. 66: Concrete stress evolution according to age at each connection detail for the material PFA30.

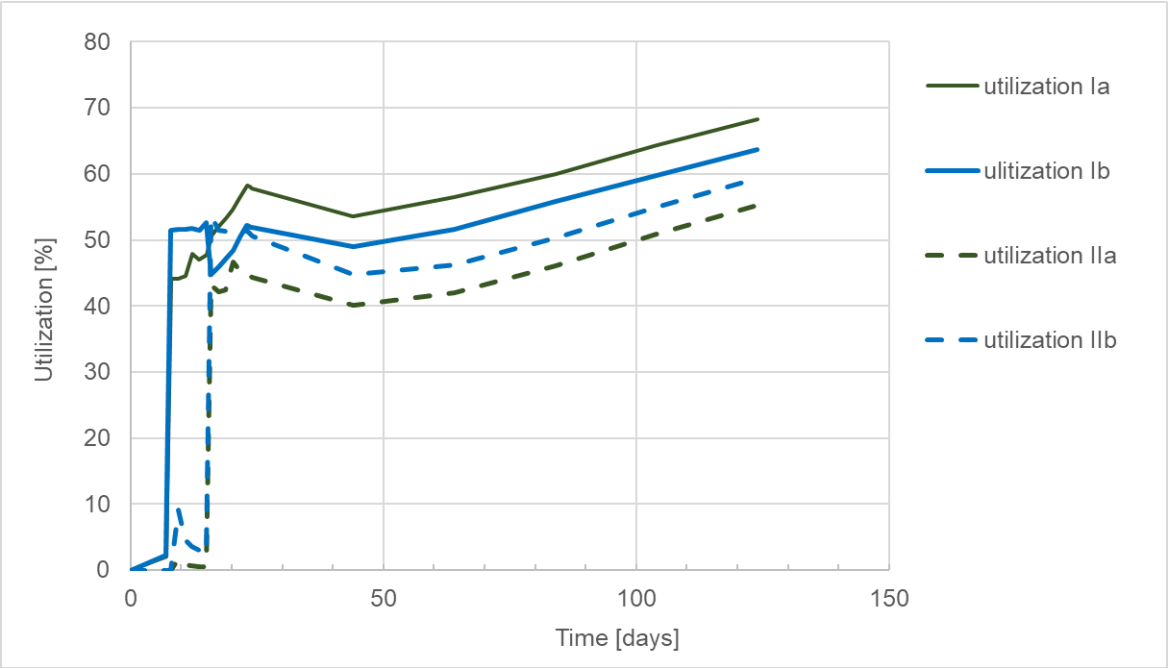


Fig. 67: Utilization of concrete at investigated connections during the construction process for the material PFA30.

6.3.4 Step 4: Verify the selected material and construction sequence by numerical simulation in ATENA-Green Concrete Module.

In this case, it can be decided to use the material PFA30 since it shows reasonable concrete utilization levels. It is interesting to note that even though the average stress level in the connection is increasing in time as shown in Fig. 66 in times from 20 – 150 days, the overall utilization level is not always increasing. It is possible to observe a partial decrease between day 20 to 50. This is since the green concrete material is slowly maturing while stresses are still slightly increasing due to the continuing construction.

Fig. 68 shows the evolution of deflections and the activation of various parts of the model in the construction sequence modelling.

The final checks of the structural behavior at the early stages should involve checking the level of concrete stresses (see Fig. 69). It should be noted that higher stress than the current concrete strength can be observed due to stress localization namely in the sharp corner between concrete and steel elements. However, the concrete should not reach the crushing state, which in ATENA software can be documented by the softening flag in the Yield/Crush Info – Softening flag as shown in Fig. 70.

Another important quantity to check is the cracking in concrete during the construction. The crack widths should be limited to micro-cracks that are barely visible. Crack widths and cracking pattern for the selected material PFA30 are shown in Fig. 54. It demonstrates that the crack widths are around 0.02 mm. The visibility crack limit is 0.05 mm and the typical crack limit in concrete design is 0.3 mm. The concrete cracking is satisfying both these limits.

The other quantities that should be checked are stresses in the steel members of the MRCS connection as well as the stresses in the reinforcement. In this case, they are clearly below the yielding strength of the steel σ_y = 200 MPa and f_{sy} = 500 MPa for reinforcement.

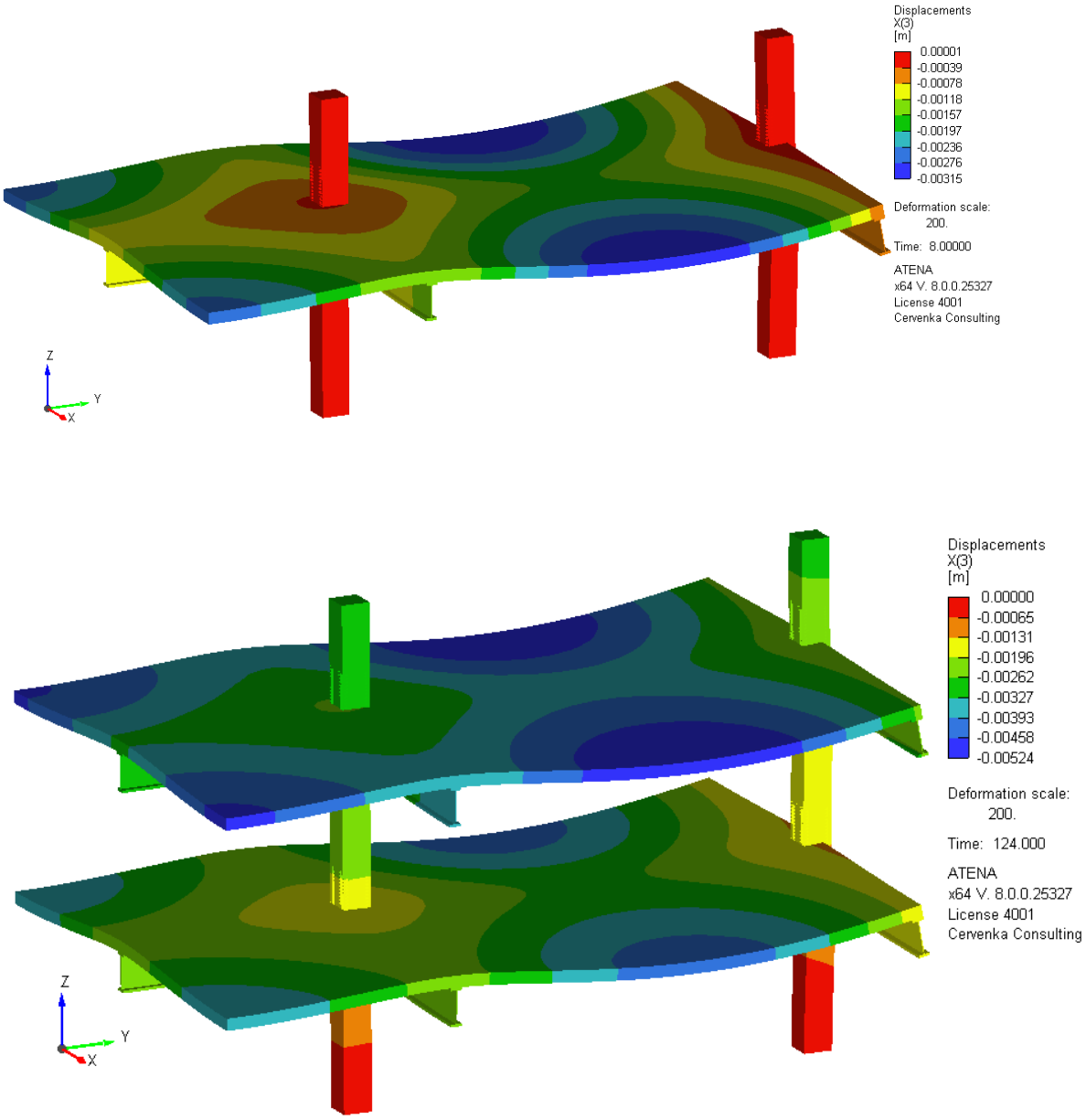


Fig. 68: Vertical deflections and modelling of the construction process in ATENA – CeSTaR-3 Green Concrete module.

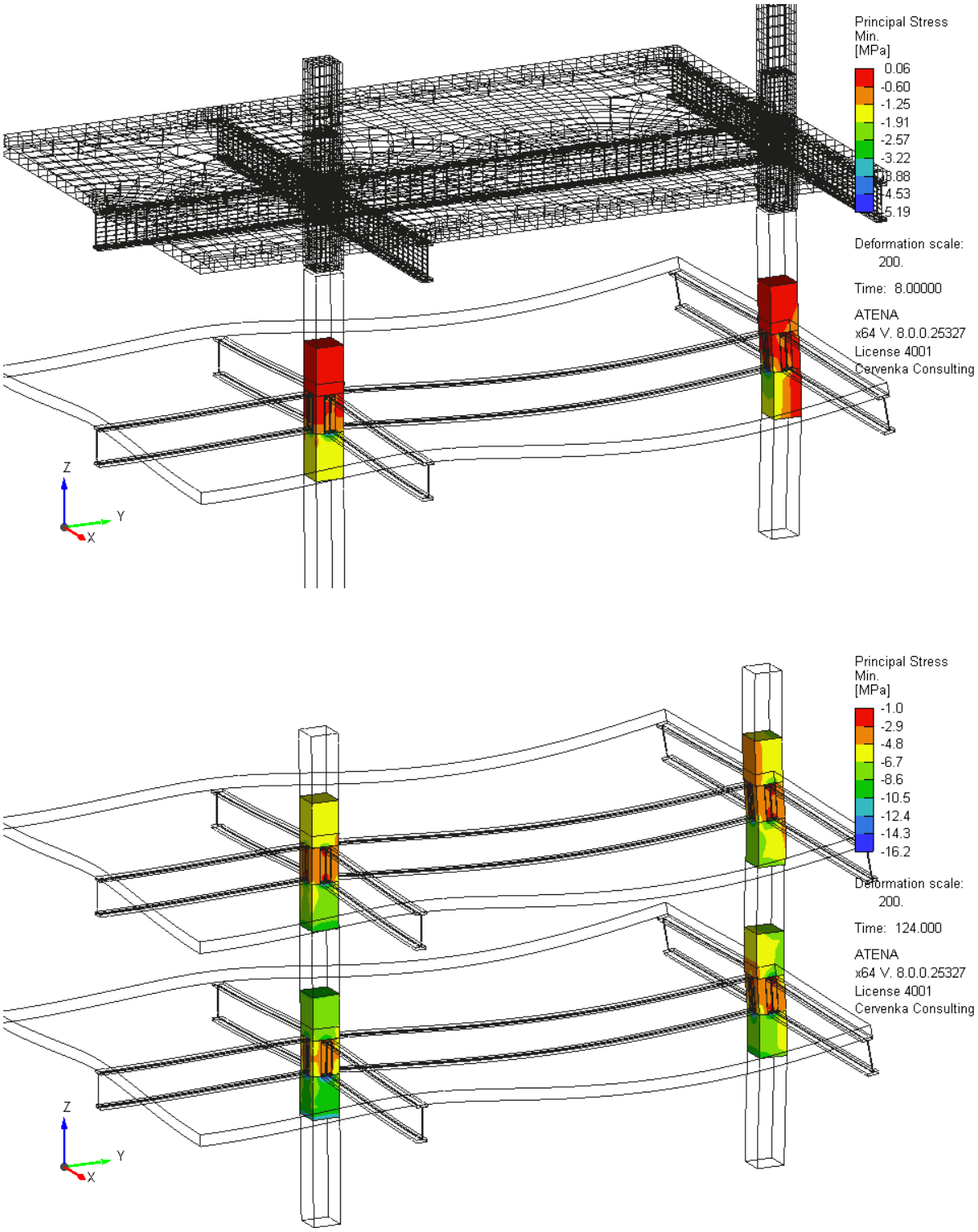


Fig. 69: Evolution of maximal concrete compressive stresses in the MRCS connection for frame example with PFA30 material. The wire frame in the top figure indicates the part of the model, which is not yet activated/constructed.

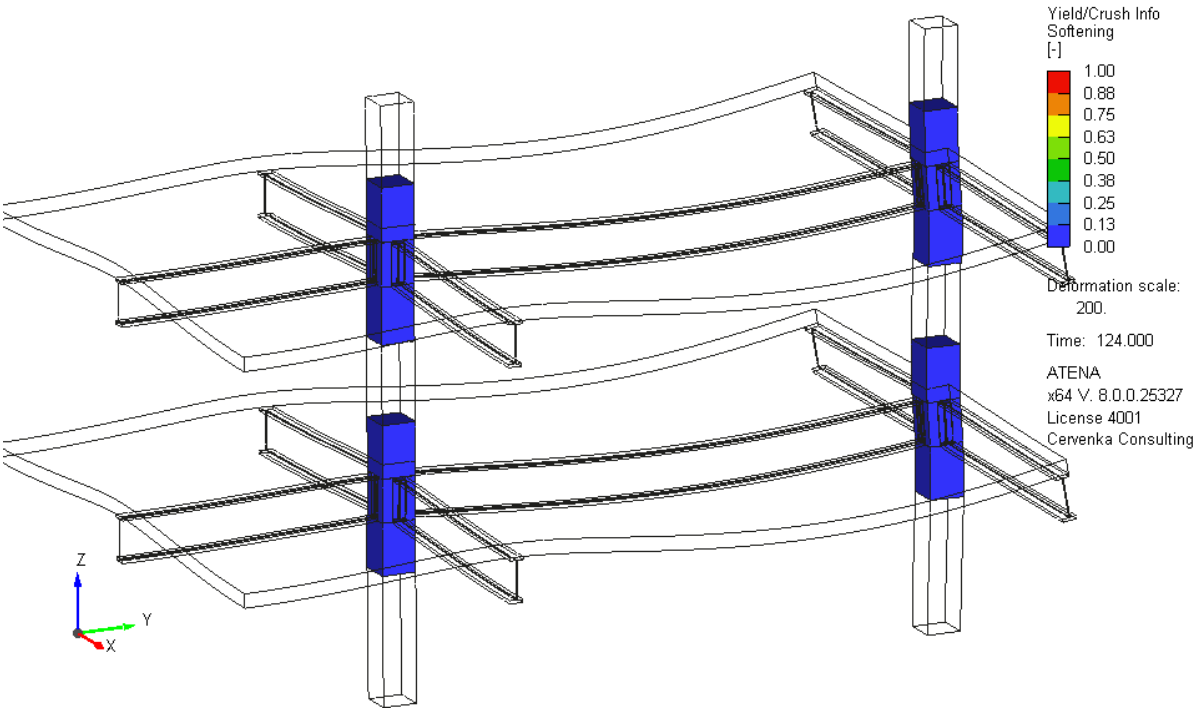


Fig. 70: Proof of no concrete crushing during the construction of MRCS frame example with PFA30 material.

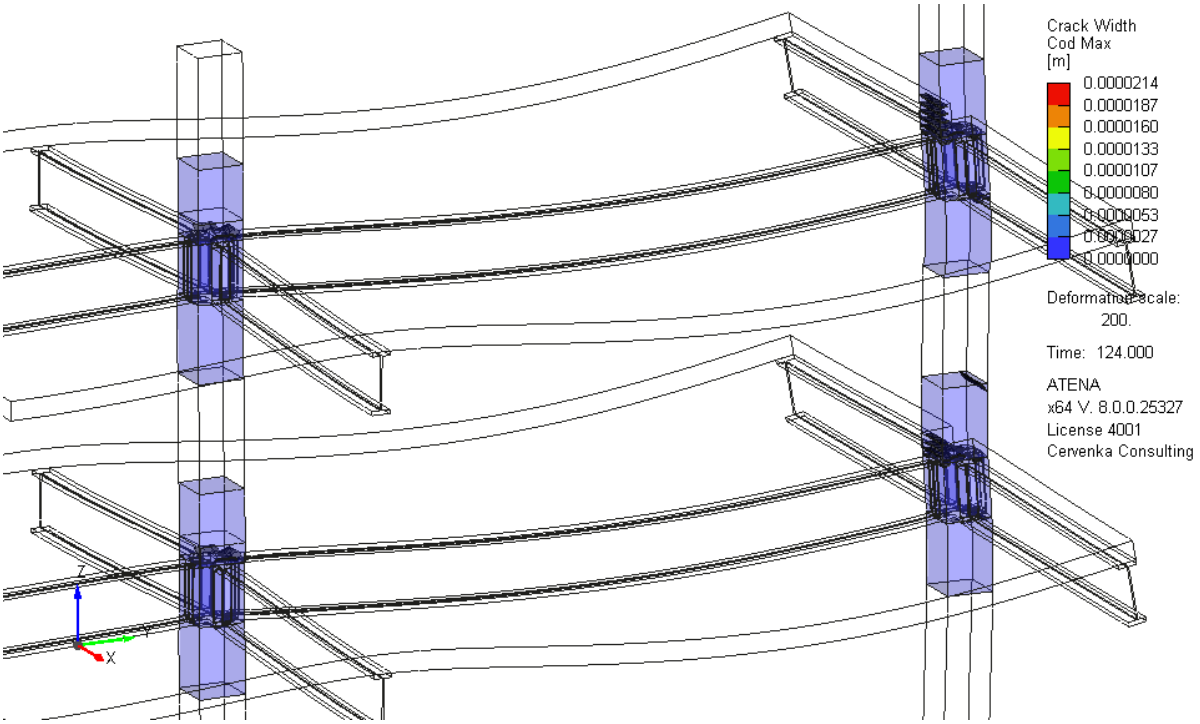


Fig. 71: Cracking at the end of the simulated construction sequence for MRCS frame example for material PFA30.

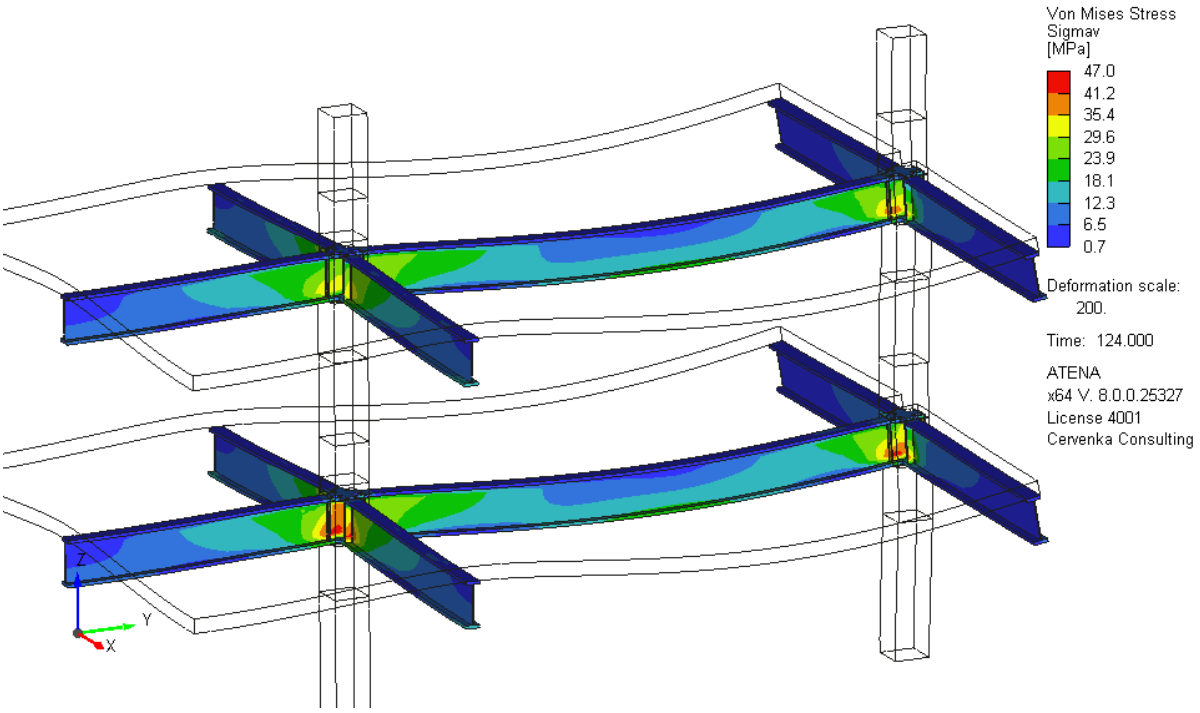


Fig. 72: Steel member stresses at the end of the simulated construction sequence for MRCS frame example for material PFA30.

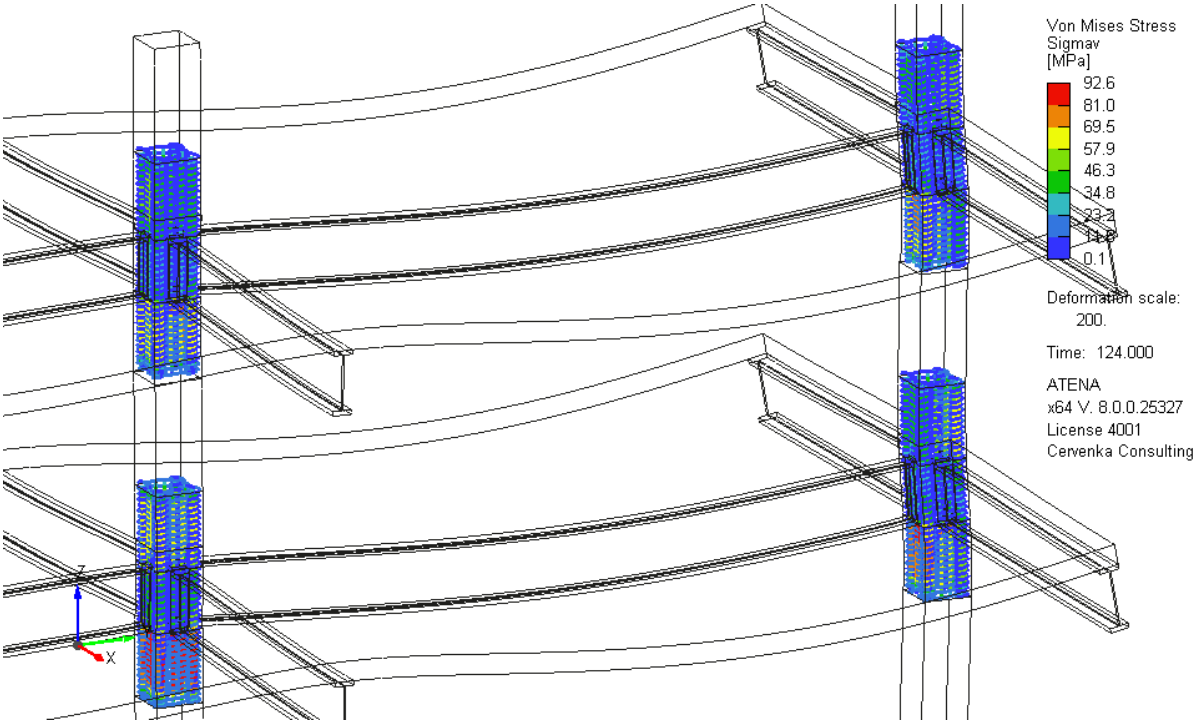


Fig. 73: Fig. 74: Steel member stresses at the end of the simulated construction sequence for MRCS frame example for material PFA30.

7 Conclusion

This document has presented the ATENA Module for Green Concrete Modelling and Design developed as the software result **TM04000013-V2** within the CeSTaR-3 bilateral research project funded by the Technology Agency of the Czech Republic under the DELTA 2 Programme. The software represents an extension of the ATENA nonlinear finite element system aimed at enabling reliable numerical simulation and design-oriented assessment of concrete structures incorporating green concrete with supplementary cementitious materials (SCMs).

The module integrates experimentally informed material models, time-dependent property evolution, and construction-sequence-aware modelling workflows into a unified nonlinear analysis environment. These capabilities address key challenges associated with the use of green concrete, including delayed strength development, modified fracture behavior, and the need for realistic assessment of structural response beyond simplified design assumptions. The linkage to the experimental database of green concrete materials (**TM04000013-V6**) ensures consistency between experimental research outcomes and numerical simulations.

The functionality of the software has been documented through a comprehensive User Manual, Technical and Theoretical Manual, and Programming Manual, covering both graphical and scripted workflows. The applicability and correctness of the implemented models have been demonstrated through a series of validation examples and benchmark studies, including comparisons with experimental results and established ATENA validation cases. These examples confirm that the software can be applied to both structural elements and structural systems under monotonic, cyclic, and time-dependent loading scenarios.

The presented software result enables structural engineers and researchers to perform advanced nonlinear analyses of green concrete structures within a robust and well-established computational framework. By transferring research-based material models and modelling concepts into an engineering-grade software implementation, the ATENA Module for Green Concrete Modelling and Design supports the practical adoption of low-carbon concrete technologies in structural design and assessment. As such, the software constitutes a significant applied outcome of the CeSTaR-3 project and contributes to more sustainable construction practices through reliable numerical simulation.

References

[1] Argyris JH. 1954 Energy Theorems and Structural Analysis. Aircraft Engineering and Aerospace Technology. Emerald, 1954.

[2] Bažant, Z.P. 1976. Instability, Ductility and Size Effect in Strain Softening Concrete, J. Engrg. Mech., ASCE, Vol. 102, No. 2, pp. 331-344.

[3] Bažant, Z.P. & Oh, B.H., 1983. Crack band theory for fracture of concrete. *Materials and Structures, RILEM* 16 (3), 155–177.

[4] Bažant, Z.P., Pijaudier-Cabot, G. 1987. Nonlocal continuum damage, localization instability and convergence. Journal of Applied Mechanics, ASME 55 (2), 287-293.

[5] Benes S., Mikolášková J., Altman T., 2025. ATENA Program Documentation, User's Manual for ATENA Studio, Cervenka Consulting s.r.o., www.cervenka.cz

[6] Bentz, E.C., Vecchio, F.J., Collins, M.P. 2006. Simplified Modified Compression Field Theory for Calculating Shear Strength of Reinforced Concrete Elements. ACI Material Journal, Jul/Aug 2006.

[7] Calatrava S. 2020. The UAE Pavilion at EXPO 2020. https://aasarchitecture.com/2021/09/the-uae-pavilion-at-expo-2020-by-santiago-calatrava/

[8] Castaldo, P., Gino, D., Bertagnoli, G., Mancini, G. 2018. Partial safety factor for resistance model uncertainties in 2D non-linear analysis of reinforced concrete structures, Engineering Structures, 176, 746-762. https://doi.org/10.1016/j.engstruct.2018.09.041.

[9] Castaldo, P., Gino, D., Bertagnoli, G., Mancini, G. 2020. Resistance model uncertainty in non-linear finite element analyses of cyclically loaded reinforced concrete systems, Engineering Structures, 211(2020), 110496, https://doi.org/10.1016/j.engstruct.2020.110496

[10] Cervenka, V., Gerstle, K., 1971. Inelastic analysis of reinforced concrete panels. Part I , Theory. Part II, Experimental verification and Application. Publication IABSE 31 (11), 32-45.

[11] Cervenka, V., 1985, Constitutive Model for Cracked Reinforced Concrete. ACI Journal Nov.-Dec. 1985, 877-882.

[12] Cervenka, V., Margoldova, J. 1995, Tension Stiffening Effect in Smeared Crack Model, Engineering Mechanics, Stain Sture (Eds), Proc. 10th Conf., ASCE EMD, May 21-24, 1995, Boulder, Colorado, pp. 655-658

[13] Cervenka, J, Bittner, Z, Havlasek, P, Rehounek, L, GREEN CONCRETE PRE-CAST ELEMENTS DESIGN AND OPTIMIZATION, Project Result TM04000013-V5, Cervenka Consulting s.r.o., 2025

[14] Cervenka J, Cervenka V, Eligehausen R, 1998 Fracture-plastic material model for concrete. Application to analysis of powder actuated anchors, FraMCoS 3, Gifu, Japan, eds. H. Mihashi and K. Rokugo, Aedificatio Publishers, Freiburg, Germany, Vol. 2, pp. 1107-1116.

[15] Cervenka J. & Papanikolaou V. 2008. Three Dimensional Combined Fracture-Plastic Material Model for Concrete. *Int Journal of Plasticity*. 24:2192-2220. doi:10.1016/j.ijplas.2008.01.004

[16] Cervenka, J., Cervenka, V., Laserna S., 2014, On finite element modelling of compressive failure in brittle materials, Computational Modeling of Concrete Structures. Bicanic et al.(Eds),Euro-C 2014, St. Anton

[17] Cervenka, J., Cervenka, V., Laserna. S., 2018a. On crack band model in finite element analysis of concrete fracture in engineering practice, Engineering Fracture Mechanics, Volume 197, 2018, Pages 27-47, ISSN 0013-7944, https://doi.org/10.1016/j.engfracmech.2018.04.010.

[18] Cervenka V. Global safety format for nonlinear calculation of reinforced concrete. Beton- und Stahlbetonbau, 103, special edition. Berlin: Ernst & Sohn; 2008. p. 37–42.

[19] Cervenka V. Reliability-based non-linear analysis according to model code 2010. J fib Struct Concr. 2013;14(1):19–28.

[20] Cervenka V, Cervenka J, Kadlec L. 2018b. Model uncertainties in numerical simulations of reinforced concrete structures. Structural Concrete; 2018: 2004–2016. https://doi.org/10.1002/suco.201700287
[21] Cervenka, V., Cervenka, J., Jendele, L., 2025. ATENA Program Documentation, Part 1: Theory, Cervenka Consulting s.r.o., www.cervenka.cz

[22] Cervenka, V., Rimkus, A., Gribniak, V., Cervenka, J., 2022. Simulation of the Crack Width in Reinforced Concrete Beams Based on Concrete Fracture. Theoretical and Applied Fracture Mechanics, 121 (2022) 103428. https://doi.org/10.1016/j.tafmec.2022.103428
[23] Cervenka, V., Cervenka, J., Rymes, J., Numerical simulation of concrete structures-From research to engineering application, Structural Concrete. 2024;1–22., 2024 fib. International Federation for Structural Concrete, DOI: 10.1002/suco.202300016
[24] Cervenka, J., Zenisek, M., Altman, T., ATENA MODULE FOR NEW MULTI-SPIRAL REINFORCED CONCRETE AND STEEL COLUMN-BEAM CONNECTIONS, Project Result TM04000013-V1, Cervenka Consulting s.r.o., 2025
[25] Collins, M.P., Vecchio, F.J., Mehlhorn, G., 1985. An international competition to predict the response of reinforced concrete panels. Canadian Journal of Civil Engineering, Vol.12, 624-644, 1985.
[26] Collins, M.P., Bentz, E.C., Quach, P., Proestos, G.T., 2015. The Challenge of Predicting the Shear Strength of Very Thick Slabs. Concrete International, V.37,No.11, Nov. 2015, pp 29-37.
[27] Costa, D.D., Cervenka, V., Costa, R.G., 2018. Model uncertainty in discrete and smeared crack prediction in RC beams under flexural loads. Engineering Fracture Mechanics 199, Elsevier, (2018) 532–543.
[28] Engen M, Hendriks M., Köhler J, Øverli JA, Åldtstedt E. 2017. A quantification of modelling uncertainty for non-linear finite element analysis of large concrete structures. Structural Safety 2017; 64: 1-8.
[29] Etse G. Theoretische und numerische untersuchung zum diffusen und lokalisierten versagen in beton. Ph.D. Thesis. University of Karlsruhe. 1998
[30] EN 1990, 2002. Eurocode 0: Basis of structural design. European Committee for Standardization (CEN). Brussels.
[31] EN 1992-1-1, 2004. Eurocode 2: Design of Concrete Structures - Part 1-1: General ruels and rules for buildings. European Committee for Standardization (CEN). Brussels.
[32] *fib* MC 2010 . International Federation for Structural Concrete. Model Code for Concrete Structures 2010. Berlin: Wilhelm Ernst & Sohn; 2013.
[33] Gino D, Castaldo P, Giordano L, Mancini G. 2021. Model uncertainty in nonlinear numerical analyses of slender reinforced concrete members. Structural Concrete. 1–26. https://doi.org/10.1002/suco.202000600
[34] de Borst, R. 1986. Non-linear analysis of frictional materials. PhD Thesis, Delft University of Technology, The Netherlands.
[35] de Borst, R., Benallal, A., and Heeres, O.M. 1996. A gradient-enhanced damage approach to fracture. J. de Physique IV, C6, pp. 491-502.
[36] de Borst R, Mühlhaus HB. Gradient dependant plasticity: formulation and algorithmic aspects. Int J Numer Meth Eng. 1992;35(3):521–39.
[37] de Borst R, Rots JG. Occurence of spurious mechanisms incomputations of strain-softening solids. Eng Comput. 1989;6:272–80.
[38] Hordijk, D.A. 1991. Local approach to fatigue of concrete. PhD Thesis, Delft University of Technology, The Netherlands.
[39] JCSS Probabilistic Model Code, 2001.https://www.jcss-lc.org/jcss-probabilistic-model-code/
[40] Jeager, T., Marti, P., 2009. Reinforced Concrete Slab Shear Prediction: Entries and Discussion. ACI Journal May-June 2009. Pp. 309-318.

[41] Jendele L, Phillips DV, 1992. Finite Element Software for Creep and Shrinkage in Concrete, Computer and Structures, 45 (1), 113-126.
[42] Jirásek, M., Bažant, Z.P. 2001. Inelastic Analysis of Structures, John Willey & Sons, LTD, Baffins Lane, Chichester, England, ISBN 0-471-98716-6
[43] Lee J, Fenves, GL. Plastic-damage model for cyclic loading of concrete structures. J. Eng. Mech, ASCE. 1998;124(8):892–900.
[44] Lin CS., Scordelis A. Nonlinear Analysis of RC Shells of General Form, ASCE, J. of Struct. Eng., Vol. 101, No. 3, 1975, pp. 152–63.
[45] Ngo, D., Scordelis, A.C. 1967. Finite element analysis of reinforced concrete beams, J. Amer. Concr. Inst. 64, pp. 152-163.
[46] Menétrey, P., Willam, K.J. 1995. Triaxial failure criterion for concrete and its generalization. ACI Structural Journal 92 (3), 311-318.
[47] Moehle, J.P., Zhai, J. 2021. Blind Prediction Competition of Shear Strength for Thick Concrete Foundation Elements With and Without Shear Reinforcement. Private communication. 2021.
[48] Ngo, D., Scordelis, A.C. 1967. Finite element analysis of reinforced concrete beams, J. Amer. Concr. Inst. 64, pp. 152-163.
[49] Pramono, E., Willam, K.J. 1989. Fracture energy-based plasticity formulation of plain concrete. Journal of Engineering Mechanics, ASCE 115 (6), 1183-1204.
[50] Rashid, Y.R. 1968. Analysis of prestressed concrete pressure vessels. Nuclear Engineering and Design 7 (4), 334-344.
[51] Rehounek, L., Zenisek, M., Numerical Simulation and Model Validation of Multispiral-Reinforced Concrete Columns' Response to Cyclic Loading, Buildings 2025, 15(21), 3855; https://doi.org/10.3390/buildings15213855
[52] Rots, J.G., Blaauwendraad, J. 1989. Crack models for concrete : Discrete or smeared ? Fixed, multi-directional or rotating ? Heron 34 (1).
[53] Schlangen, E. 1993. Experimental and Numerical Analysis of Fracture Processes in Concrete, Ph.D. dissertation, Delf University of Technology. 1993.
[54] Suidan, M., Schnobrich, W.C. 1973. Finite Element Analysis of Reinforced Concrete, ASCE, J. of Struct. Div., Vol. 99, No. ST10, pp. 2108-2121
[55] Swiler L.P., Giunta A.A., 2007. Aleatory and Epistemic Uncertainty Quantification for Engineering Applications. Sandia National Laboratories. American Statistical Association, July 30 - August 2, 2007 in Salt Lake, UT. https://www.osti.gov/servlets/purl/1147526
[56] Terzic, V. et al, 2021. Quasi-static cyclic test of an RC column. Private communication. Nov. 2021.
[57] Turner M., Clough RW. 1956 Stiffness and deflection analysis of complex structures. Journal of the Aeronautical Science. Vol.23, September 1956, No. 9.
[58] Vrouwenvelder A.C.W.M., 2002. Developments towards full probabilistic design codes. Structural Safety, Volume 24, Issues 2–4, April–October 2002, Pages 417-432.
[59] Zienkiewicz O.C., Cheung Y.K. 1967. The Finite Element Method in Structural and Continuum Mechanics. McGraw-Hill, London, 1967. p 274.
[60] Zienkiewicz O.C., Valliappan S., King I.P. 1969. Elasto-plastic solutions of engineering problems 'initial stress' finite element approach. Int. Journal for Numerical Methods in Engineering. Volume1, Issue1, January/March 1969, pp. 75-100 https://doi.org/10.1002/nme.1620010107